

On the Channel-Sensitive Delay Behavior of LIFO-Backpressure

Wei Si and David Starobinski

Dept. of Electrical and Computer Engineering, Boston University, USA
 {weisi, staro}@bu.edu

Abstract—In this paper, we study the delay performance of backpressure routing algorithms using LIFO schedulers (LIFO-backpressure). We uncover a surprising behavior in which, under certain channel conditions, the average delay of packets decreases as the traffic load in the network increases. We propose and analyze a queueing-theoretic model under which the scheduler can transmit packets only if the queue length (i.e., the number of packets in the queue) meets or exceeds a threshold, and we show that the model analytically bears out the observed phenomenon. Using matrix geometric methods, we derive a numerical solution for the average packet delay in the general case, and, using z -transform techniques, we further provide closed-form solutions for the average delay in special cases. Our analysis indicates that when the threshold is fixed (as may happen under lossless channel conditions), the average delay increases with increasing traffic load, as expected. On the other hand, when the threshold fluctuates (as may happen under changing, lossy channel conditions), the average delay may decrease, sometimes substantially, with the traffic load. We corroborate these findings with TOSSIM simulations using real channel traces and run on different types of networks.

Keywords—Backpressure algorithms, queueing theory, data collection protocols, wireless sensor networks.

I. INTRODUCTION

Backpressure routing algorithms promise throughput-optimal performance and provide elegant cross-layer solutions for a wide range of networking problems [1]. Yet, they also notoriously suffer from high end-to-end packet delays. This problem is exacerbated at low traffic load due to the lack of sufficient pressure to drive packets toward their destinations.

The work in [2] proposes an elegant solution to the delay problem by replacing the standard first-in-first-out (FIFO) queueing schedulers at routing nodes by last-in-first-out (LIFO) schedulers. LIFO-backpressure traps a few packets at each queue to establish a routing gradient and ensures fast delivery of most other packets. This joint routing-scheduling policy has been analytically demonstrated to achieve an optimal utility-delay tradeoff [3].

LIFO-backpressure has been implemented in the form of a data collection protocol for wireless sensor networks, called the Backpressure Collection Protocol (BCP) [2]. Unlike minimum-cost tree routing algorithms (e.g., [4]), BCP makes routing and forwarding decisions based on local information and does not need to explicitly compute paths. Extensive simulations and testbed experiments show that LIFO-BCP drastically improves delay performance over the FIFO-based version of BCP.

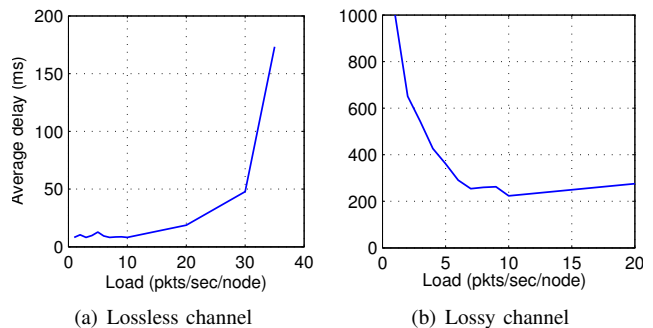


Fig. 1. Average end-to-end packet delay of LIFO-backpressure in a five-node wireless sensor network simulation under lossless and lossy channels.

Nevertheless, our own TOSSIM simulations show that LIFO-backpressure can exhibit intriguing delay behavior in certain conditions, as illustrated in Fig. 1 (the simulation set-up, which uses real RSSI traces, is described in detail in Section VI). Under lossless channel conditions as shown in Fig. 1(a), the average delay of packets increases with the traffic load, in a manner that is consistent with standard queueing models, such as $M/M/1$. On the other hand, under lossy channel conditions, wherein a non-negligible fraction of packets get lost and require re-transmissions, we observe an opposite trend: *the end-to-end average delay of delivered packets decreases with the traffic load*, at least initially. Thus, Fig. 1(b) indicates that the average delay of packets when packets are generated at a rate of one per second at each node is four times higher than that when packets are generated at rate of seven per second at each node (i.e., 1000 ms in the former case versus 250 ms in the latter case).

The goal of this paper is to explain this strange behavior within the context of understanding the impact of channel and traffic conditions on the delay behavior of LIFO-backpressure schedulers. We introduce and analyze a queueing-theoretic model that *qualitatively* captures the behavior of LIFO-backpressure. Specifically, we focus on a two-node network consisting of one source node and one destination node. This simple network turns out to be sufficient to reproduce the observed effects.

The behavior of the LIFO-backpressure scheduler at the source node is modelled using a single-queue system with threshold. The threshold may change over time, depending on channel conditions. The scheduler can transmit packets only if the queue length (i.e., the number of packets in the queue) meets or exceeds the threshold. Under appropriate

statistical assumptions on the traffic and channel dynamics, the evolution of such a system can be described using a multi-dimensional continuous-time Markov chain (CTMC). We derive a numerical solution for the general case using matrix geometric methods [5]. Furthermore, using z -transform techniques [6], we provide closed-form solutions for the special cases where the threshold oscillates between 0 and 1 and between 0 and ∞ .

Our analysis indicates that the counter-intuitive delay behavior, whereby the average delay initially decreases with the traffic load, occurs due to threshold changes. The analysis further reveals that this effect gets more pronounced as the rate of threshold changes becomes slower. On the other hand, if the threshold is fixed (e.g., if the channel is lossless), then the average delay increases with the traffic load as expected.

The rest of this paper is organized as follows. In Section II, we review related work on backpressure routing algorithms. In Section III, we detail the BCP protocol, upon which our analytical model is based. In Section IV, we formulate our CTMC model and provide a matrix geometric method for numerically solving the general model. Next, in Section V, we derive closed-form expressions of the average delay in some special cases. Section VI presents simulation results for larger networks to support our analytical findings. Section VII concludes the paper.

II. RELATED WORK

The origin of backpressure algorithms lies in the seminal work of Tassiulas and Ephremides [7]. A backpressure algorithm is mathematically constructed by minimizing the Lyapunov drift that represents the difference between the values of the Lyapunov function at the current time slot and at the next time slot. This leads to a problem, known as MaxWeight, of maximizing the weighted sum of link rates, in which the weights are represented by backlog differentials. Intuitively, data packets are sent over links with high rates and to neighbors with low backlog, thus achieving a load balancing effect.

The chief advantages of backpressure algorithms are to avoid explicit path computations and achieve throughput-optimal performance. However, backpressure algorithms suffer from high end-to-end packet delays, due to lack of backpressure to push packets toward their destinations, sometimes leading to packet looping. These problems are more severe at light load. An extreme case is of a packet entering an empty network and engaging into some kind of random walk until reaching its destination.

Several approaches have been proposed to solve the delay problem of backpressure algorithms [8–11]. Instead of using queue differentials as weights of the MaxWeight problem, [9] proposes representing weights with delay information of packets in the queues. The idea is that packets that have already experienced high delays are more likely to be scheduled for transmission in the next time slot, whereas the original backpressure algorithm would give longer queues higher priority irrespective of the delay experienced by packets. The authors in [10] describe a novel backpressure-based per-packet randomized routing framework. It leverages a shadow queue structure that lowers complexity of maintaining queues. By minimizing the number of hops by packets, their routing algorithms reduce delay drastically.

Based on the original backpressure algorithms, Neely *et al.* developed so-called quadratic Lyapunov function based algorithms (QLA) for general stochastic network utility optimization problems [1]. Instead of purely minimizing the Lyapunov drift, QLA is constructed by minimizing the Lyapunov drift plus a penalty (or the negative of a utility), in which the penalty is weighted by a parameter V . As V gets larger, the algorithm puts more emphasis on the resulting penalty and less on network stability. The performance results of QLA are given in the following $[O(1/V), O(V)]$ utility-delay tradeoff form: backpressure is able to achieve a utility that is within $O(1/V)$ of the optimal utility for any scalar $V \geq 1$, while guaranteeing an average network delay that is $O(V)$. Although QLA does not emerge specifically as a solution for the delay problem of the original backpressure algorithms, QLA can inherently prevent packet looping when the penalty function is related to the number of transmissions since looping adds transmissions. However, a large delay may still prevail at low load due to the lack of backpressure to push packets toward their destinations.

Much effort has been spent to reduce the large $O(V)$ delay of QLA. The authors in [12] prove that under QLA, the network backlog stays close to a fixed value (called attractor), which is the dual optimal solution of a deterministic optimization problem. While the attractor has order of $O(V)$, the fluctuation of the network backlog around the attractor is bounded by $O(\log^2(V))$ with high probability. The authors, therefore, propose an algorithm that pre-fills queues with null packets that play the role of attractor. Hence, the real packets arrive into a queue whose length is bounded by $O(\log^2(V))$, and the algorithm achieves an optimal $[O(1/V), O(\log^2(V))]$ utility-delay tradeoff.

Motivated by practical implementations of backpressure routing algorithms, the authors in [3] prove that LIFO-backpressure achieves the optimal $[O(1/V), O(\log^2(V))]$ utility-delay tradeoff. Note that FIFO-backpressure would achieve a $[O(1/V), O(V)]$ utility-delay tradeoff since packets need to traverse a whole queue in order to get transmitted. The idea behind LIFO-backpressure is straightforward: packets constituting the attractor are trapped in the queue forever and serve the same role as that of null packets in the algorithm described above. The delay improvement of LIFO over FIFO is shown both through real experiments [2] and theoretical studies [3].

Most of the above studies focus on the optimal utility-delay tradeoff in terms of the scalar parameter V (or when the parameter V becomes large). Little study has been conducted on the effects of other network parameters on the delay performance of backpressure routing algorithms, including channel dynamics and traffic load in the network. As we have shown in Section I, even though LIFO-backpressure achieves an optimal utility-delay tradeoff performance, its delay at low load may be very high. The observed delay behavior can hardly be explained by the previous theoretical studies. This work serves the goal of better understanding the behavior of LIFO-backpressure and shedding light on the effects of network parameters (channel conditions and network traffic) on its delay performance.

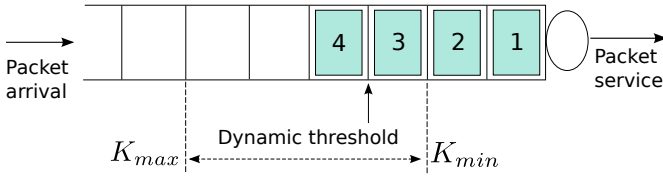


Fig. 2. Illustration of queueing system with dynamic threshold. In this example, the threshold has the dynamic range [2,6] and the current threshold is 3. Due to LIFO policy and threshold range, packet 1 and packet 2 will never have chance to be served and stay in the queue forever.

III. BCP EXPLAINED

In this section, we describe the design of the BCP protocol [2], since it serves as the basis of our queueing-theoretic model. BCP is a practical, distributed QLA implementation, where nodes independently make routing decisions based on local information. The routing decisions are made per packet instead of routing all packets through the same computed path.

Next, we explain how BCP make routing decisions. Let Q_i represent the backlog at node i . Then $\Delta Q_{i,j} = Q_i - Q_j$ is the queue differential (backpressure) between node i and its neighbor node j . Let $\bar{R}_{i \rightarrow j}$ denote the estimated link rate from i to j and $\overline{ETX}_{i \rightarrow j}$ be the average number of transmissions for a packet to be successfully sent over the link. In the routing policy of BCP, node i calculates the following backpressure weight for each neighbor j :

$$w_{i,j} = (\Delta Q_{i,j} - V \cdot \overline{ETX}_{i \rightarrow j}) \cdot \bar{R}_{i \rightarrow j}.$$

The routing decision (next hop of the packet) is determined by finding the neighbor j^* with the highest weight. Then the node needs to make the forwarding decision: if $w_{i,j^*} > 0$, the packet is forwarded to node j^* , else the packet is held until the metric is recomputed. In other words, if the weights for all neighbour nodes are zero or negative, the node will do nothing but wait till the next recomputation.

As a QLA algorithm, BCP aims to minimize the number of packet transmissions (ETX) while guaranteeing network stability. The parameter V represents the weight of the penalty (ETX) in the optimization problem. When $V = 0$, the algorithm reverts to the original backpressure algorithm.

Due to the routing policy of BCP, the queue dynamics at a node is subject to the queue dynamics and link dynamics to all neighbors. Instead of studying large networks, we focus our efforts on a simple two-node network. In this network, packets are injected into the source node s and forwarded to the destination node t . Under BCP, the source node simply calculates the weight:

$$\begin{aligned} w_{s,t} &= (\Delta Q_{s,t} - V \cdot \overline{ETX}_{s \rightarrow t}) \cdot \bar{R}_{s \rightarrow t} \\ &= (Q_s - V \cdot \overline{ETX}_{s \rightarrow t}) \cdot \bar{R}_{s \rightarrow t}. \end{aligned}$$

The second equation comes from the fact that $Q_t = 0$. Since s only has t as its neighbor node, s does not need to choose the next hop and only needs to make forwarding decisions. Furthermore, we can drop $\bar{R}_{s \rightarrow t}$ because it does not affect the sign of the backpressure weight. For ease of discussion, we discard the subscripts in the formula. Based on BCP and the form of backpressure weight, the source node forwards a packet only when $Q > V \cdot \overline{ETX}$. When $Q \leq V \cdot \overline{ETX}$, the source node is waiting either for the queue Q to grow

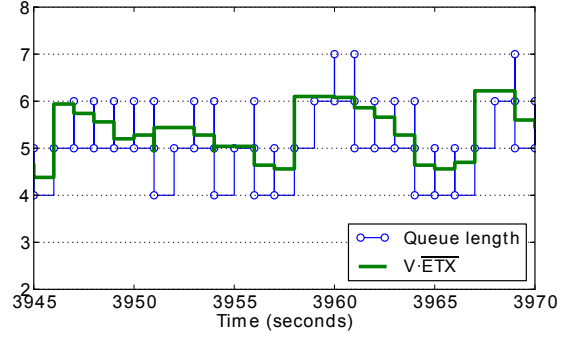


Fig. 3. Simulated evolution of queue length and ETX of BCP over time with $V = 2$. This illustrates the role of $V \cdot \overline{ETX}$ as the threshold on the queue.

or \overline{ETX} to become smaller. Thus, the value of $V \cdot \overline{ETX}$ serves as a threshold on the queue.

First, let's take a look at the scenario of a lossless channel or more generally a static channel with fixed \overline{ETX} . In this case, the threshold is static with value $V \cdot \overline{ETX}$. Due to the forwarding policy of BCP, Q will be lower bounded by $V \cdot \overline{ETX}$. Under FIFO, the average delay is $D = \frac{Q}{\lambda} \geq V \cdot \frac{\overline{ETX}}{\lambda}$ by Little's Law, where λ denotes the packet arrival rate. This lower bound is consistent with the $O(V)$ delay result in theoretical analysis. As the load λ increases, the lower bound decreases. Under LIFO, a fraction of packets, the number of which is equal to the threshold $V \cdot \overline{ETX}$, is trapped in the queue forever. Ignoring these packets, the rest of the queue is equivalent to an $M/M/1$ queue, for which the average delay increases as load increases.

Next, let's suppose the channel has a dynamic \overline{ETX} in the range of $[\overline{ETX}_{min}, \overline{ETX}_{max}]$. Correspondingly, the threshold is dynamic within range $[V \cdot \overline{ETX}_{min}, V \cdot \overline{ETX}_{max}]$, which we further simply denote by $[K_{min}, K_{max}]$. Then Q will be lower bounded by K_{min} due to the threshold range. Under FIFO, the average delay is $D = \frac{Q}{\lambda} \geq \frac{K_{min}}{\lambda} = V \cdot \frac{\overline{ETX}_{min}}{\lambda}$. However, under LIFO, the bottom K_{min} packets are trapped in the queue forever and the rest of the queue will be equivalent of a queue with dynamic threshold in the range of $[0, K_{max} - K_{min}]$. For example, in Fig. 2, the threshold range is [2, 6]. Under FIFO, the packet needs to go through all the queue to get served and the queue length is at least 2 due to the range of threshold. However, under LIFO, packet 1 and packet 2 are in the queue forever. Thus the rest of the queue is equivalent to a queue with threshold range $[0, 4]$. Fig. 3 illustrates the threshold effect of ETX on the queue length.

IV. GENERAL MODEL AND NUMERICAL METHOD

In this section, we construct a system-level queueing model with dynamic threshold based on the routing policy of LIFO-backpressure (LIFO-BCP) and represent it with a CTMC. Meanwhile, we provide a matrix geometric method to numerically solve the CTMC and obtain the average delay of packets in the queueing system.

A. Queueing model

Assume that the arrival process of packets is Poisson with rate λ . The channel is represented by the Gilbert model [13], a Markov chain that transits between two states, namely,

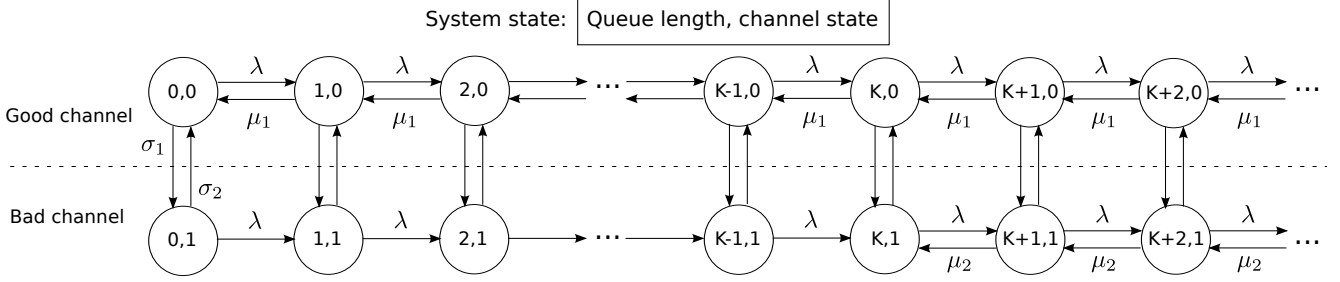


Fig. 4. Markov chain of the single-queue system.

good state and bad state. The transition rate from good state to bad state is σ_1 and the transition rate from bad state to good state is σ_2 . Under the good channel, the threshold is 0 and service time is exponentially distributed with rate μ_1 while under the bad channel, the threshold is K and the service time is exponentially distributed with rate μ_2 (usually, $\mu_1 \geq \mu_2$). Thus in association with the channel model, the threshold dynamic can also be represented by a two-state Markov chain. Let $(n, 0)$ and $(n, 1)$ represent the system states of n packets in the queue under good and bad channels, respectively. Then Fig. 4 depicts the whole Markov chain for the queueing system.

Although the simplistic Gilbert channel model assumes that the channel can only be in two different states, it is sufficient for qualitatively capturing the temporal dynamics and correlation of more complex channel models. We also note that under lossless channel with fixed threshold, the system state can have transitions restricted to the half of the Markov chain under good channel, which is the same as $M/M/1$.

B. Probability generating function

The steady state probability of $(n, 0)$ and $(n, 1)$ are denoted by $P_{n,0}$ and $P_{n,1}$. Define $P_n \triangleq [P_{n,0}, P_{n,1}]^T$. Then the steady state distribution of the queue length is

$$\pi_n = P_{n,0} + P_{n,1} = \mathbf{e}^T P_n. \quad (1)$$

We define the following probability generating functions using z -transform: $G_0(z) = \sum_{n=0}^{\infty} z^n P_{n,0}$, $G_1(z) = \sum_{n=0}^{\infty} z^n P_{n,1}$, and

$$\mathcal{G}(z) = \begin{bmatrix} G_0(z) \\ G_1(z) \end{bmatrix} = \sum_{n=0}^{\infty} z^n \begin{bmatrix} P_{n,0} \\ P_{n,1} \end{bmatrix} = \sum_{n=0}^{\infty} z^n P_n. \quad (2)$$

The probability generating function of the steady state distribution of queue length N is

$$\mathcal{F}_N(z) = \sum_{n=0}^{\infty} z^n \pi_n = \sum_{n=0}^{\infty} z^n \mathbf{e}^T P_n = \mathbf{e}^T \mathcal{G}(z). \quad (3)$$

Then the average number of packets in the queue can be obtained from $\mathcal{F}_N(z)$ and the average delay can be calculated by Little's Law:

$$\mathbb{E}[N] = \sum_{n=0}^{\infty} n \pi_n \quad (4)$$

$$= \left. \frac{d}{dz} \mathcal{F}_N(z) \right|_{z=1}, \quad (5)$$

$$\mathbb{E}[T] = \mathbb{E}[N] / \lambda. \quad (6)$$

Next we develop a matrix geometric method [5] for solving the steady state distribution of the CTMC and calculating the average delay by (4) and (6). In Section V, we will derive closed-form solutions for the probability generating functions, (2) and (3), and compute the average delay based on (5) and (6) for two special cases.

C. Matrix geometric method

Based on the balance equations and the normalization condition, we aim to obtain the steady state distribution, P_n .

We first derive the balance equations at each state of the CTMC. The balance equations at states $(0, 0)$ and $(0, 1)$ are:

$$\begin{aligned} (\sigma_1 + \lambda)P_{0,0} &= \sigma_2 P_{0,1} + \mu_1 P_{1,0}, \\ (\sigma_2 + \lambda)P_{0,1} &= \sigma_1 P_{0,0}. \end{aligned}$$

Define $Q \triangleq \begin{bmatrix} \sigma_1 & -\sigma_2 \\ -\sigma_1 & \sigma_2 \end{bmatrix}$, $\Lambda \triangleq \begin{bmatrix} \lambda & 0 \\ 0 & \lambda \end{bmatrix}$, $M_1 \triangleq \begin{bmatrix} \mu_1 & 0 \\ 0 & 0 \end{bmatrix}$, then the above balance equations can be simplified as:

$$(Q + \Lambda)P_0 = M_1 P_1.$$

The balance equations at states $(n, 0)$ and $(n, 1)$ ($1 \leq n < K$) are:

$$\begin{aligned} (\lambda + \mu_1 + \sigma_1)P_{n,0} &= \lambda P_{n-1,0} + \mu_1 P_{n+1,0} + \sigma_2 P_{n,1}, \\ (\lambda + \sigma_2)P_{n,1} &= \lambda P_{n-1,1} + \sigma_1 P_{n,0}, \\ \Rightarrow (\Lambda + M_1 + Q)P_n &= \Lambda P_{n-1} + M_1 P_{n+1}. \end{aligned}$$

The balance equations at states $(K, 0)$ and $(K, 1)$ are:

$$\begin{aligned} (\lambda + \mu_1 + \sigma_1)P_{K,0} &= \lambda P_{K-1,0} + \mu_1 P_{K+1,0} + \sigma_2 P_{K,1}, \\ (\lambda + \sigma_2)P_{K,1} &= \lambda P_{K-1,1} + \mu_2 P_{K+1,1} + \sigma_1 P_{K,0}. \end{aligned}$$

Define $M_2 \triangleq \begin{bmatrix} \mu_1 & 0 \\ 0 & \mu_2 \end{bmatrix}$, then

$$(\Lambda + M_1 + Q)P_K = \Lambda P_{K-1} + M_2 P_{K+1}.$$

The balance equations at states $(n, 0)$ and $(n, 1)$ ($n > K$) are:

$$\begin{aligned} (\lambda + \mu_1 + \sigma_1)P_{n,0} &= \lambda P_{n-1,0} + \mu_1 P_{n+1,0} + \sigma_2 P_{n,1}, \\ (\lambda + \mu_2 + \sigma_2)P_{n,1} &= \lambda P_{n-1,1} + \mu_2 P_{n+1,1} + \sigma_1 P_{n,0}, \\ \Rightarrow (\Lambda + M_2 + Q)P_n &= \Lambda P_{n-1} + M_2 P_{n+1}. \end{aligned}$$

In summary, the balance equations are the following:

$$\begin{cases} (\Lambda + Q)P_0 = M_1 P_1, & (7) \\ (\Lambda + M_1 + Q)P_n = \Lambda P_{n-1} + M_1 P_{n+1}, 0 < n < K & (8) \\ (\Lambda + M_1 + Q)P_K = \Lambda P_{K-1} + M_2 P_{K+1}, & (9) \\ (\Lambda + M_2 + Q)P_n = \Lambda P_{n-1} + M_2 P_{n+1}, n > K. & (10) \end{cases}$$

Now we choose P_1 as an unknown vector and express P_n as a linear transform of P_1 . By (7), we have

$$P_0 = (Q + \Lambda)^{-1} M_1 P_1 \triangleq T_1 P_1.$$

We express P_n in the matrix geometric form:

$$P_n = \begin{cases} \mathcal{R}_1^{n-1} P_1, & \text{if } 1 \leq n < K, \\ \mathcal{R}_2^{n-K} P_K, & \text{if } n \geq K. \end{cases}$$

Then by taking $P_{K+1} = \mathcal{R}_2 P_K$ into (9), we have:

$$\begin{aligned} P_K &= (\Lambda + M_1 + Q - M_2 \mathcal{R}_2)^{-1} \Lambda P_{K-1} \\ &= (\Lambda + M_1 + Q - M_2 \mathcal{R}_2)^{-1} \Lambda \mathcal{R}_1^{K-2} P_1 \\ &\triangleq T_2 P_1. \end{aligned}$$

The steady state distribution can now be expressed as follows:

$$P_n = \begin{cases} T_1 P_1, & \text{if } n = 0, \\ \mathcal{R}_1^{n-1} P_1, & \text{if } 1 \leq n < K, \\ \mathcal{R}_2^{n-K} T_2 P_1, & \text{if } n \geq K. \end{cases} \quad (11)$$

The sum of P_n from 0 to ∞ is

$$\left[\sum_{n=0}^{\infty} P_n \right] = (T_1 + \sum_{n=1}^{K-1} \mathcal{R}_1^{n-1} + \sum_{n=K}^{\infty} \mathcal{R}_2^{n-K} T_2) P_1.$$

Based on the normalization condition and the two-state channel model, we have the following equation, from which P_1 can be solved:

$$\begin{bmatrix} 1 & 1 \\ \sigma_1 & -\sigma_2 \end{bmatrix} \begin{bmatrix} \sum_{n=0}^{\infty} P_{n,0} \\ \sum_{n=0}^{\infty} P_{n,1} \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}. \quad (12)$$

Before solving (12), we need to determine \mathcal{R}_1 and \mathcal{R}_2 . They can be solved numerically as follows. By (8), we have

$$(\Lambda + M_1 + Q) \mathcal{R}_1 P_1 = \Lambda P_1 + M_1 \mathcal{R}_1^2 P_1. \quad (13)$$

A sufficient condition to satisfy (13) is

$$(\Lambda + M_1 + Q) \mathcal{R}_1 = \Lambda + M_1 \mathcal{R}_1^2. \quad (14)$$

To find \mathcal{R}_1 , we can iteratively calculate the following until convergence:

$$\mathcal{R}_{1(j)} = (\Lambda + M_1 + Q)^{-1} (\Lambda + M_1 \mathcal{R}_{1(j-1)}^2),$$

where $\mathcal{R}_{1(j)}$ is the approximation to \mathcal{R}_1 at the j -th step.

[5] has shown that by starting with $\mathcal{R}_{1(0)} = 0$, the sequence $\{\mathcal{R}_{1(0)}, \mathcal{R}_{1(1)}, \mathcal{R}_{1(2)}, \dots\}$ is a monotonically increasing sequence that converges to the minimal nonnegative solution to (14).

Similarly, \mathcal{R}_2 can also be found through iteratively calculating

$$\mathcal{R}_{2(j)} = (\Lambda + M_2 + Q)^{-1} (\Lambda + M_2 \mathcal{R}_{2(j-1)}^2).$$

With \mathcal{R}_1 , \mathcal{R}_2 and P_1 known and by (1), (4), (6), and (11), the average delay of packets in the queueing system is

$$\mathbb{E}(T) = e^T \left(\sum_{n=1}^{K-1} n \mathcal{R}_1^{n-1} + \sum_{n=K}^{\infty} n \mathcal{R}_2^{n-K} T_2 \right) P_1 / \lambda. \quad (15)$$

The computation of the geometric sum in (15) can be conveniently carried out through diagonalization and eigen-decomposition of \mathcal{R}_1 and \mathcal{R}_2 . The method we describe here applies directly for the case of $K \geq 2$. The average packet

delay when $K = 1$ can also be numerically computed using the matrix geometric method with minor change. However, we will instead provide a closed-form analysis in the next section. Numerical results obtained by the matrix geometric method will be presented in Section VI.

V. ANALYSIS OF SPECIAL CASES

In this section, we provide analytical results of two special cases of the general model. First, we analyze the case where the threshold varies between 0 and 1. This represents our main result as it explains the counter-intuitive delay behavior of LIFO-backpressure under lossy channels. Then, we analyze the case where the threshold varies between 0 and ∞ , which turns out to have a similar delay behavior as $M/M/1$.

A. $K = 1$

When $K = 1$, the balance equations are:

$$(\Lambda + Q) P_0 = M_1 P_1, \quad (16)$$

$$(\Lambda + M_1 + Q) P_1 = \Lambda P_0 + M_2 P_2, \quad (17)$$

$$(\Lambda + M_2 + Q) P_n = \Lambda P_{n-1} + M_2 P_{n+1}, \text{ for } n \geq 2 \quad (18)$$

Multiplying both sides of (17) and (18) with z^n and summing from $n = 1$ to ∞ , we can get

$$\begin{aligned} (\Lambda + M_2 + Q) \sum_{n=1}^{\infty} z^n P_n &= z(M_2 - M_1) P_1 \\ &+ \Lambda z \sum_{n=1}^{\infty} z^{n-1} P_{n-1} + M_2 \frac{1}{z} \sum_{n=1}^{\infty} z^{n+1} P_{n+1}, \end{aligned}$$

According to definition of $\mathcal{G}(z)$ in (2),

$$\begin{aligned} (\Lambda + M_2 + Q)[\mathcal{G}(z) - P_0] &= z(M_2 - M_1) P_1 + \Lambda z \mathcal{G}(z) \\ &+ M_2 \frac{1}{z} [\mathcal{G}(z) - P_0 - z P_1]. \end{aligned}$$

We then replace P_0 by $(Q + \Lambda)^{-1} M_1 P_1$ from (16):

$$\begin{aligned} [z^2 \Lambda - z(\Lambda + M_2 + Q) + M_2] \mathcal{G}(z) &= \\ (1 - z)[M_2(Q + \Lambda)^{-1} M_1 + z(M_2 - M_1)] P_1. \end{aligned} \quad (19)$$

To simplify, we rewrite (19) as:

$$\mathcal{A}(z) \mathcal{G}(z) = (1 - z) \mathcal{B}(z) P_1,$$

where

$$\begin{aligned} \mathcal{A}(z) &= z^2 \Lambda - z(\Lambda + M_2 + Q) + M_2, \\ \mathcal{B}(z) &= M_2(Q + \Lambda)^{-1} M_1 + z(M_2 - M_1). \end{aligned}$$

Then

$$\mathcal{G}(z) = \frac{\text{adj} \mathcal{A}(z)}{\det \mathcal{A}(z)/(1 - z)} \mathcal{B}(z) P_1. \quad (20)$$

In order to obtain $\mathcal{G}(z)$, we need to solve P_1 . Since it is a two-dimension vector, we need to find two equations. The first equation is the normalization condition, i.e., $\mathcal{F}_N(z)|_{z=1} = 1$, and by (3), we have

$$e^T \frac{\text{adj} \mathcal{A}(z)|_{z=1}}{[\det \mathcal{A}(z)/(1 - z)]|_{z=1}} \mathcal{B}(z)|_{z=1} P_1 = 1. \quad (21)$$

The second equation is obtained by finding a root of $\det \mathcal{A}(z) = 0$ such that the root z_0 satisfies $0 < z_0 < 1$. Then the second equation is

$$\text{adj} \mathcal{A}(z_0) \mathcal{B}(z_0) P_1 = 0. \quad (22)$$

Assuming $\sigma_1 = \sigma_2 = \sigma$, $\mu_1 = \mu_2 = \mu$, we have

$$\begin{aligned} \text{adj} \mathcal{A}(z) &= \begin{bmatrix} \mu - (\lambda + \mu + \sigma)z + \lambda z^2 & -\sigma z \\ -\sigma z & \mu - (\lambda + \mu + \sigma)z + \lambda z^2 \end{bmatrix}, \\ \det \mathcal{A}(z) &= (1-z)(\mu - \lambda z)[\mu - (\lambda + \mu + 2\sigma)z + \lambda z^2], \end{aligned}$$

and

$$z_0 = (2\sigma + \lambda + \mu - \sqrt{(2\sigma + \lambda + \mu)^2 - 4\mu\lambda})/2\lambda.$$

By solving (21) and (22), we obtain

$$P_1 = \frac{\lambda(\mu - \lambda)}{\mu} \begin{bmatrix} \frac{1}{\mu} - \frac{2\lambda^2}{E_1} \\ \frac{2\lambda}{E_1} \end{bmatrix}, \quad (23)$$

where

$$\begin{aligned} E_1 &= \lambda\mu + 4\lambda\sigma + 2\mu\sigma - (\lambda + 2\sigma)E_2 + 3\lambda^2 + 4\sigma^2, \\ E_2 &= \sqrt{(\lambda + \mu + 2\sigma)^2 - 4\mu\lambda}. \end{aligned}$$

By substituting (23) into (20) and using (3), (5), and (6), the average delay of packets in the queueing system when $K = 1$ is

$$\mathbb{E}[T] = \frac{1}{\mu - \lambda} + \frac{2\lambda}{3\lambda^2 + (\mu + 4\sigma)\lambda - E_2\lambda - 2E_2\sigma + (2\mu\sigma + 4\sigma^2)}. \quad (24)$$

Expanding the Maclaurin series of (24) on λ , we obtain the following approximation of the average delay at low load:

$$\mathbb{E}[T] = \left(\frac{2}{\mu} + \frac{1}{2\sigma}\right) - \frac{(\mu + 4\sigma)(\mu + \sigma)}{2\mu\sigma^2(\mu + 2\sigma)}\lambda + o(\lambda). \quad (25)$$

By (25), the first order derivative of the average delay on the load is strictly negative. Thus the average delay decreases with load under light traffic. This is consistent with the counter-intuitive behavior of LIFO-backpressure observed in simulations. The trend of delay decreasing at low load is more apparent when σ is small, i.e., the rate at which the threshold varies between good and bad states is low. As σ grows larger, both the zeroth and first order derivative become smaller, which leads to the interesting conclusion that the delay performance of a fast-varying channel is better than that of slow-varying channel. As we will show in Section VI, the counter-intuitive delay trend indeed vanishes when σ gets large.

An intuitive explanation of the counter-intuitive delay behavior is that the packet at the bottom of queue gets stuck when the threshold is 1 and gets served when the threshold returns to 0. Thus the queueing delay of the stuck packet is mostly determined by the transition time of the threshold from 1 to 0, which could be very large. When the traffic load increases, the probability that the queue length is strictly larger than one in the bad state increases and the proportion of stuck packets decreases, thus reducing the overall average delay.

B. $K = \infty$

When $K = \infty$, there will be no packet service when the channel is in a bad state. Then the balance equations are:

$$\begin{cases} (\Lambda + Q)P_0 = M_1P_1, \\ (\Lambda + M_1 + Q)P_n = \Lambda P_{n-1} + M_1P_{n+1}, \text{ for } n \geq 1 \end{cases} \quad (26)$$

By multiplying both sides of (27) with z^n and summing up, we have

$$(\Lambda + M_1 + Q)(\mathcal{G}(z) - P_0) = z\Lambda\mathcal{G}(z) + \frac{M_1}{z}(\mathcal{G}(z) - P_0 - zP_1).$$

Using (26), we have

$$\begin{aligned} [z^2\Lambda - z(\Lambda + M_1 + Q) + M_1]\mathcal{G}(z) &= (1-z)M_1P_0 \\ &= (1-z) \begin{bmatrix} \mu_1 P_{0,0} \\ 0 \end{bmatrix}. \end{aligned}$$

and we further rewrite it as:

$$\mathcal{A}(z)\mathcal{G}(z) = (1-z) \begin{bmatrix} \mu_1 P_{0,0} \\ 0 \end{bmatrix},$$

where

$$\mathcal{A}(z) = z^2\Lambda - z(\Lambda + M_1 + Q) + M_1.$$

Then the probability generating function can be expressed as follows:

$$\mathcal{G}(z) = \frac{\text{adj} \mathcal{A}(z)}{\det \mathcal{A}(z)/(1-z)} \begin{bmatrix} \mu_1 P_{0,0} \\ 0 \end{bmatrix}, \quad (28)$$

where

$$\begin{aligned} \text{adj} \mathcal{A}(z) &= \begin{bmatrix} (\lambda + \sigma_2)z - \lambda z^2 & \sigma_2 z \\ \sigma_1 z & \mu_1 - (\lambda + \mu_1 z + \sigma_1)z + \lambda z^2 \end{bmatrix}, \\ \det \mathcal{A}(z) &= z(1-z)[\lambda\mu_1 + \mu_1\sigma_2 + \lambda^2 z^2 \\ &\quad - \lambda(\lambda + \mu_1 + \sigma_1 + \sigma_2)z]. \end{aligned}$$

In order to obtain $\mathcal{G}(z)$, we need to determine the unknown variable, $P_{0,0}$. Since there is only one variable, we only need one equation. Using the normalization condition and (3), the solution is

$$P_{0,0} = \frac{\sigma_2}{\sigma_1 + \sigma_2} - \frac{\lambda}{\mu_1}. \quad (29)$$

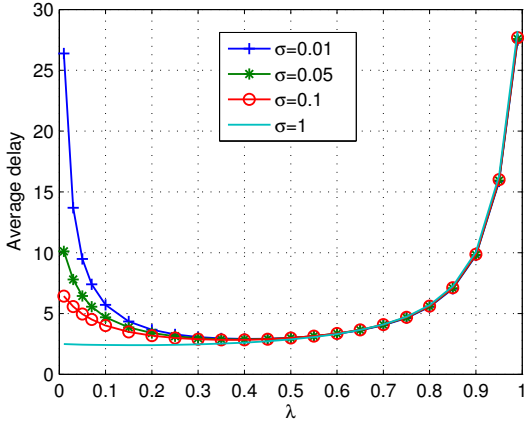
By substituting (29) into (28) and using (3), (5), and (6), the average delay of packets in the queueing system when $K = \infty$ is

$$\mathbb{E}[T] = \left[1 + \frac{\sigma_1\mu_1}{(\sigma_1 + \sigma_2)^2}\right] / \left[\frac{\sigma_2}{\sigma_1 + \sigma_2}\mu_1 - \lambda\right]. \quad (30)$$

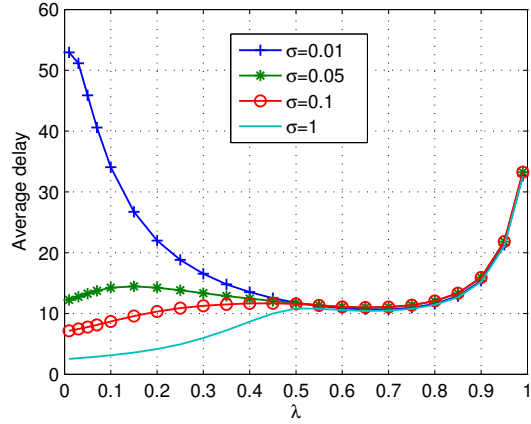
Therefore, when $K = \infty$, the average delay is a monotonically increasing function on the load, and behaves in a manner similar as the average delay in an $M/M/1$ queue. The result is not surprising since the queue can only be served during a fraction of the time, i.e., $\frac{\sigma_2}{\sigma_1 + \sigma_2}$, which does not depend on other system parameters such as the arrival rate and service rates.

VI. NUMERICAL AND SIMULATION RESULTS

In this section, we provide numerical results obtained by the matrix geometric method and z -transform method described in Sections IV and V. We also provide simulation results of LIFO-BCP to verify the existence of the counter-intuitive delay behavior of LIFO-backpressure in large networks.

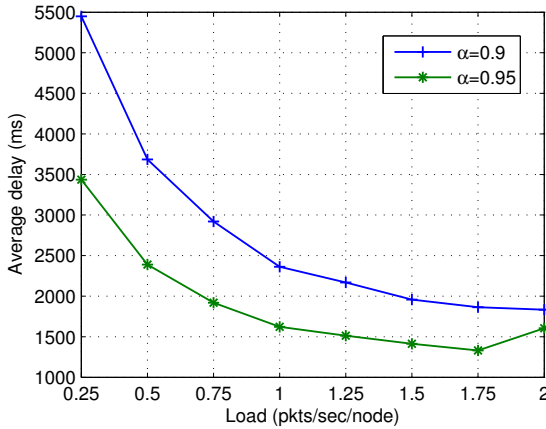


(a) $K = 1$ by z -transform method

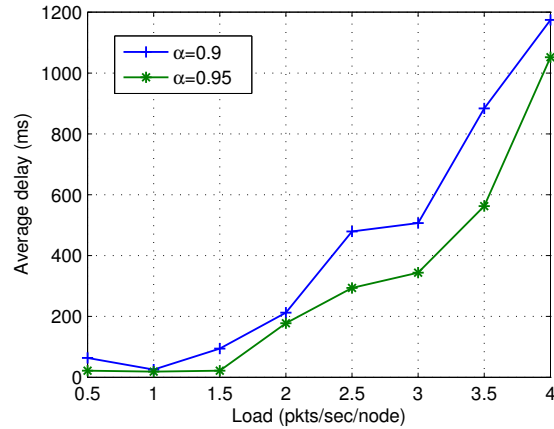


(b) $K = 10$ by matrix geometric method

Fig. 5. Average packet delay versus packet arrival rate (traffic load) λ for different threshold transition rates σ and fixed service rate $\mu = 1$.



(a) Lossy channel (Received power = -80 dBm)



(b) Lossless channel (Received power = -75 dBm)

Fig. 6. Average delay versus load with fixed noise power -85 dBm in a 25-node grid network.

A. Numerical results

Numerical results for the average packet delay in the queuing model are depicted in Fig. 5 (a) and (b), for the cases $K = 1$ and $K = 10$, respectively. The results for $K = 1$ show that the average delay decreases with traffic at low load. This phenomenon gets more pronounced as the transition rates between different channel states become slower (i.e., $\sigma \rightarrow 0$). On the other hand, for $K = 10$, the average delay generally increases with the traffic load, unless the transition rates between different channels states are very slow (e.g., $\sigma = 0.01$). This result is consistent with our analysis for $K = \infty$. For both cases $K = 1$ and $K = 10$, all the curves merge as $\lambda \rightarrow 1$. This means that temporal channel dynamics do not have as much effect at high load.

B. Simulation results

We next describe simulations of the BCP protocol. Our goal is to verify that our analysis qualitatively captures the behavior of this protocol under different channel conditions. Our simulation is run on TOSSIM [14], the standard TinyOS simulator for wireless sensor networks. The simulated network consists of a root node and some sensor nodes, both of which use the sensor model MICAz. In a simulation, the

sensor nodes are first initialized uniformly randomly within one second. After initialization, all the sensor nodes periodically generate packets and inject them into the network layer, where BCP routes the packets toward the root node. The goal of the random initialization is to reduce the amount of MAC contention and MAC delays that would occur if all the nodes generated packets at the same time.

Our first set of simulations are performed on a five-node network. The results are depicted in Fig. 1, shown in the introduction of the paper. The simulations use real RSSI (received signal strength) traces collected from a vehicular environment, where each sensor node is attached to a different wheel of a car and the root node is placed on the driver seat [15]. For the lossless channel, we configure the noise power to be -95 dBm, while for the lossy channel, we use real noise traces collected from the Meyer Library of Stanford [16]. These traces exhibit complex temporal dynamics, wherein the noise floor is at about -98 dBm and spikes are at about -86 dBm. The results are consistent with our analytical findings, that is, the initial decrease of the delay with traffic load occurs under bursty channel conditions, but not under perfect channel conditions.

Our second set of simulations are conducted for a network

consisting of 24 sensor nodes and one root node. The topology is a 5×5 grid where the root node is located at the center. In this topology, a link only exists between direct neighbors. In other words, nodes that are two hops away cannot hear each other. We fix the noise power to be -85 dBm while we test different received signal powers, namely -80 dBm and -75 dBm. The packet error probability at signal-to-noise-ratio (SNR) of 10 dB is close to zero while that at SNR of 5 dB is varying in the range between 0 and 1/2 in the simulator. Therefore, the two different received powers represent lossless and lossy channels.

Simulations are run for different values of the parameter α used by the BCP protocol in its estimation of the expected number of transmissions \overline{ETX} . This estimation is based on an exponential moving weighted average, where \overline{ETX} is updated as follows whenever a new sample of ETX is obtained: $\overline{ETX}_{new} = \alpha \overline{ETX}_{old} + (1 - \alpha)ETX$.

Fig. 6 shows results for the two different received powers. In Fig. 6(a), when the channel is lossy and the threshold is dynamic, the average delay decreases with the load. In Fig. 6(b), on the other hand, when the channel is lossless and the threshold is static, the average delay increases with the load. We note that the average delay at low load in the dynamic case is at least two orders of magnitude larger than in the static case. This phenomenon occurs even though the average number of transmissions in the dynamic case is only at most twice larger than that in the static case. These results showcase the manifestation and significance of channel-sensitive delay behaviors of LIFO-backpressure in large networks. We note that increasing the value of α somewhat helps to alleviate this problem, but does not eliminate it.

VII. CONCLUSION

We developed a queueing-theoretic model and solved it using matrix geometric numerical methods, to elucidate the channel-sensitive delay behavior of LIFO-backpressure. We also provided closed-form analytical results on the average delay in two special cases. Our results show that the counter-intuitive delay behavior is most pronounced when the channel is slowly-varying and changes in the threshold value are not large. We verified the existence and significance of the channel-sensitive delay behaviors of LIFO-backpressure in large networks through simulations.

Our analysis is tied to the Backpressure Collection Protocol (BCP), which aims to minimize the number of transmissions over a long horizon. Since BCP roots in the stochastic network utility optimization framework, the phenomenon uncovered in this paper and the analysis associated with it may apply to other network optimization problems solved using threshold-based algorithms with time-varying threshold. A possible example includes network economics where the price for each bit changes over time and acts as a dynamic threshold on the queue length [1, 17].

A natural solution to improve the delay performance of LIFO-backpressure at low load is to inject additional traffic in the network. This can be achieved by allowing nodes to transmit duplicates or encoded linear combinations of previous packets. However, redundant traffic may exacerbate congestion at high load. Thus, such a solution must be carefully engineered and is left as an interesting area for future work.

ACKNOWLEDGEMENTS

This work was supported in part by NSF under grant CCF-0916892.

REFERENCES

- [1] L. Georgiadis, M. J. Neely, and L. Tassiulas, *Resource Allocation and Cross-Layer Control in Wireless Networks*. Foundations and Trends in Networking, 2006.
- [2] S. Moeller, A. Sridharan, B. Krishnamachari, and O. Gnawali, "Routing without routes: the backpressure collection protocol," in *IPSN*, 2010.
- [3] L. Huang, S. Moeller, M. Neely, and B. Krishnamachari, "Lifo-backpressure achieves near-optimal utility-delay tradeoff," *Networking, IEEE/ACM Transactions on*, vol. 21, no. 3, pp. 831–844, 2013.
- [4] O. Gnawali, R. Fonseca, K. Jamieson, D. Moss, and P. Levis, "Collection tree protocol," in *SenSys*, 2009.
- [5] M. F. Neuts, *Matrix Geometric Solutions in Stochastic Models*. The Johns Hopkins University Press, Baltimore, 1981.
- [6] J. N. Daigle, *Queueing Theory for Telecommunications*. Addison-Wesley Publishing Company, Inc., 1992.
- [7] L. Tassiulas and A. Ephremides, "Stability properties of constrained queueing systems and scheduling policies for maximum throughput in multihop radio networks," *Automatic Control, IEEE Transactions on*, vol. 37, no. 12, pp. 1936–1948, 1992.
- [8] M. Alresaini, M. Sathiamoorthy, B. Krishnamachari, and M. Neely, "Backpressure with adaptive redundancy (bwar)," in *INFOCOM*, 2012.
- [9] B. Ji, C. Joo, and N. Shroff, "Delay-based back-pressure scheduling in multi-hop wireless networks," in *INFOCOM*, 2011.
- [10] E. Athanasopoulou, L. Bui, T. Ji, R. Srikant, and A. Stolyar, "Back-pressure-based packet-by-packet adaptive routing in communication networks," *Networking, IEEE/ACM Transactions on*, vol. 21, no. 1, pp. 244–257, 2013.
- [11] L. Ying, S. Shakkottai, A. Reddy, and S. Liu, "On combining shortest-path and back-pressure routing over multihop wireless networks," *Networking, IEEE/ACM Transactions on*, vol. 19, no. 3, pp. 841–854, Jun. 2011.
- [12] L. Huang and M. Neely, "Delay reduction via lagrange multipliers in stochastic network optimization," *Automatic Control, IEEE Transactions on*, vol. 56, no. 4, pp. 842–857, 2011.
- [13] E. N. Gilbert, "Capacity of a burst-noise channel," *Bell System Technical Journal*, vol. 39, pp. 1253–1265, Sep. 1960.
- [14] P. Levis, N. Lee, M. Welsh, and D. Culler, "Tossim: accurate and scalable simulation of entire tinyos applications," in *SenSys*, 2003.
- [15] M. Hashemi, W. Si, M. Laifelfeld, D. Starobinski, and A. Trachtenberg, "Intra-car Wireless Sensors Data Aggregation: A Multi-hop Approach," in *VTC*, 2013.
- [16] H. Lee, A. Cerpa, and P. Levis, "Improving wireless simulation through noise modeling," in *IPSN*, 2007.
- [17] P. Marbach and R. Berry, "Downlink resource allocation and pricing for wireless networks," in *INFOCOM*, 2002.