Channel Sensitivity of LIFO-Backpressure: Quirks and Improvements

Wei Si, David Starobinski, Morteza Hashemi, Moshe Laifenfeld, and Ari Trachtenberg

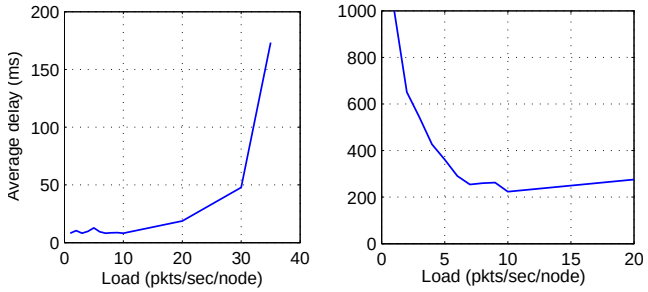
***Abstract*—We study the delay performance of backpressure routing algorithms using LIFO schedulers (LIFO-backpressure). We uncover a surprising behavior in which, under certain channel conditions, the average delay of packets is high at low traffic load and decreases as the load in the network increases. We propose and analyze a queueing-theoretic model under which the scheduler can transmit packets only if the queue length 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 a special case. Our analysis indicates that when the threshold is fixed (as may happen under lossless channel conditions), the average delay is small at low traffic load and increases with increasing load, as expected. On the other hand, when the threshold fluctuates (as may happen under changing, lossy channel conditions), the average delay *may* be high at low load and decrease, sometimes substantially, with the traffic load. We corroborate these findings with TOSSIM simulations on different types of networks, using measured channel traces. Further, we propose a replication-based LIFO-backpressure algorithm (RBL) to improve the delay performance of LIFO-backpressure. Analytical and simulation results show that RBL dramatically reduces the delay of LIFO-backpressure at low load, while maintaining high throughput performance at high load.**

***Index Terms*—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 [2]. 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 [3] 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[1] tradeoff [4].



(a) Lossless channel (b) Lossy channel

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

LIFO-backpressure has been implemented in the form of a data collection protocol for wireless sensor networks, called the Backpressure Collection Protocol (BCP) [3], which routes packets toward a single destination (sink). Unlike minimum-cost tree routing algorithms (e.g., [5]), 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.

Nevertheless, our own TOSSIM simulations (using the source code of BCP in TinyOS [3]) 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 is small at low traffic and increases with the 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 is high at low traffic load and decreases with the 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., 1,000 ms in the former case versus 250 ms in the latter case).

The first goal of this paper is to explain this unexpected behavior within the context of understanding the impact of channel and traffic conditions on the delay behavior of

An earlier and shorter version of this paper appeared in the proceedings of the IEEE 2013 Annual Allerton Conference [1]. This work was supported in part by the US National Science Foundation (NSF) under grants CCF-0916892, CCF-0964652, CNS-1012910, and CNS-1409053.

W. Si and D. Starobinski are with the Division of Systems Engineering, Boston University; M. Hashemi and A. Trachtenberg are with the Department of Electrical and Computer Engineering, Boston University, Boston, MA 02215 USA (e-mail: weisi@bu.edu; staro@bu.edu; mhashemi@bu.edu; trachten@bu.edu).

M. Laifenfeld is with Wireless Enablers Group of GM Advanced Technical Center, 7 Hamada St, Herzliya 46733, Israel (e-mail: moshe.laifenfeld@gm.com).

[1]The delay of a packet is defined as the time elapsing from its generation at the source node till its delivery at the destination node. The computation of the average delay applies only to delivered packets, thus excluding those packets which have not reached the destination.

LIFO-backpressure schedulers. We introduce and analyze a queueing-theoretic model that *qualitatively* captures the behavior of LIFO-backpressure.

Specifically, we initially 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 is related to the expected number of transmissions (ETX) needed for a successful packet reception on a given channel. Thus 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 [6]. Furthermore, using z-transform techniques [7], we provide closed-form solutions for the special case where the threshold oscillates between 0 and 1.

Next, we conduct a delay analysis of LIFO-backpressure for a chain network in a low load regime. Our analysis indicates that the high delay of LIFO-backpressure at low load occurs due to slow variations of the threshold. On the other hand, if the threshold is fixed (e.g., if the channel is lossless), then the average delay is small at low load and increases with the traffic load as expected.

The second main contribution of this work is to propose a novel lightweight mechanism, called *replication-based LIFO-backpressure (RBL)*, to remedy the above problem. Through the analysis of an approximate CTMC, that is asymptotically exact at low load, we show that RBL improves delay performance over LIFO-backpressure at low load. We implement the replication mechanism into BCP, and refer to the new implementation as RBL-BCP. Our simulations of RBL-BCP demonstrate the delay performance improvement over BCP. The simulations also show that RBL does not compromise throughput performance of BCP at high load.

In summary, our contributions are the following:

- We propose and analyze a queueing-theoretic model to elucidate the high delay problem of LIFO-backpressure at low load;
- We propose and analyze a replication-based LIFO-backpressure algorithm that reduces the large delay of LIFO-backpressure at low load;
- Through extensive simulations, we demonstrate the existence of the high delay problem of LIFO-backpressure in large networks and show that RBL significantly mitigates this issue.

The rest of this paper is organized as follows. In Section II, we review related work on backpressure routing algorithms and describe the BCP protocol, upon which our analytical model is based. In Section III, we formulate our CTMC model and provide a matrix geometric method for numerically solving the general model. We also derive closed-form expressions of the average delay for a special case. In Section IV, we analyze the delay of LIFO-backpressure in a chain network at low load. Section V describes our proposed RBL and provides corresponding analysis results. Section VI presents simulation results for larger networks to support our analytical findings and compares performance of RBL-BCP and BCP. Finally, Section VII concludes the paper.

II. Related work

A. Backpressure algorithms

The origin of backpressure algorithms lies in the seminal work of Tassiulas and Ephremides [8]. 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 [9].

Several approaches have been proposed to solve the delay problem of backpressure algorithms [10–14]. Instead of using queue differentials as weights of the MaxWeight problem, [11] 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 [12] 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. [13] proposes a hybrid routing algorithm based on a shortest-path algorithm and the backpressure routing. By forcing a set of constraints on the number of hops that can be traversed by packets, this method prevents packets from long paths exploration. Similarly, the authors in [14] propose the use of combination of a shortest-path algorithm and the backpressure method in order to improve delay performance. Furthermore, they show that implementing per-neighbor queues instead of per-flow queues can further reduce delays, as well as system implementation complexity.

B. Quadratic Lyapunov function based algorithms

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 [2]. 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 $O(V)$. QLA can 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 [15] 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 [4] 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 [3] and theoretical studies [4].

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. 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.

The authors in [10] propose backpressure with adaptive redundancy (BWAR) in the context of delay-tolerant networks. BWAR uses an adaptive redundancy mechanism to improve the delay performance of backpressure at low load. While RBL resembles BWAR, it differs in two key aspects: (i) BWAR is based on the original backpressure algorithm while BRL is based on QLA, with a penalty function that aims to minimize the number of transmissions, and (ii) BWAR generates duplicates whenever the queue length falls below some preset threshold, whereas RBL generates replicas only when packets get stuck in the queue due to channel degradation. We detail RBL in Section V. Note that the idea of injecting redundant packets was also proposed in [16, 17] for routing in delay-tolerant networks.

We distinguish this paper from our previous work [1] by addition of our proposed replication-based LIFO-backpressure algorithm. We also provide the analysis and simulation results to show that RBL improves delay performance over LIFO-backpressure.

Algorithm 1 BCP

```
while Q_i > 0 do
    Compute the backpressure weight w_{i,j} for each neighbor j
    Find the neighbor j* such that j* = arg max_j w_{i,j}
    if w_{i,j*} > 0 then
        Transmit one packet to j*
        Update ETX_{i→j*} and R_{i→j*}
    else
        Wait for a reroute period
    end if
end while
```

C. BCP explained

BCP [3] 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.

Since all the packets are routed to the same destination, each node only needs to maintain one queue. Let $\mathcal{Q}_i$ represent the backlog at node i. Then $\Delta\mathcal{Q}_{i,j} = \mathcal{Q}_i - \mathcal{Q}_j$ is the queue differential (backpressure) between node i and its neighbor node j. Let $\overline{\mathcal{R}}_{i\to j}$ denote the estimated link rate (measured in packets per second) from i to j and $\overline{ETX}_{i\to j}$ 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:

$$w_{i,j} = (\Delta\mathcal{Q}_{i,j} - V \cdot \overline{ETX}_{i\to j}) \cdot \overline{\mathcal{R}}_{i\to j}.$$

where, V is a trade-off parameter.

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 (after a *reroute period*). A pseudo-code of BCP is given in Algorithm 1. Since routing decisions are made on a per-packet basis, at most one packet can be transmitted at each iteration of the "while" loop in the algorithm.

As a QLA algorithm, BCP aims to minimize the number of packet transmissions (ETX) while guaranteeing network stability. The parameter V ($V \geq 1$) represents the weight of the penalty (ETX) in the optimization problem.

QLA assumes that ETX is perfectly observed while BCP estimates $\overline{ETX}$ based on an exponential moving weighted average. $\overline{ETX}$ is updated as follows whenever a new sample of ETX is obtained: $\overline{ETX}_{new} = \alpha\overline{ETX}_{old} + (1-\alpha)ETX$, in which the default value of α is 0.9. In TinyOS, a node

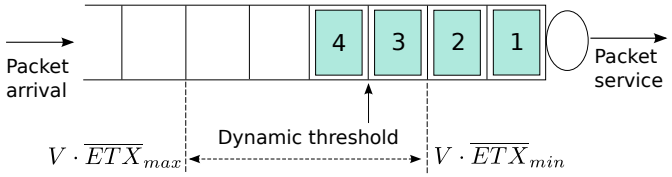


Fig. 2. Illustration of queueing system with dynamic threshold. In this example, the threshold has the dynamic range [2,6] and the current threshold is 3. Due to LIFO policy and threshold range, packet 1 and packet 2 will never have a chance to be served and stay in the queue forever.

makes at most five attempts to transmit a packet to a neighbor. Therefore, $1 \leq \overline{ETX} \leq 5$. The link rate is calculated as the reciprocal of the packet transmission time. The estimated link rate $\overline{\mathcal{R}}$ is updated based on an exponential moving weighted average.

III. Two-node analysis

In this section, we model and analyze LIFO-backpressure (LIFO-BCP) in the context of a two-node network. Based on the routing policy of LIFO-backpressure, we construct a system-level queueing model with dynamic threshold and represent it with a CTMC. Then, we provide a matrix geometric method to numerically solve the CTMC and obtain the average delay of packets in the queueing system. Meanwhile, we use z-transform technique to analyze the average delay for a special case.

A. BCP in two-node network

In the two-node 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:

$$\begin{aligned} w_{s,t} &= (\Delta\mathcal{Q}_{s,t} - V \cdot \overline{ETX}_{s\to t}) \cdot \overline{\mathcal{R}}_{s\to t} \\ &= (\mathcal{Q}_s - V \cdot \overline{ETX}_{s\to t}) \cdot \overline{\mathcal{R}}_{s\to t}. \end{aligned}$$

The second equation comes from the fact that $\mathcal{Q}_t = 0$. 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 $\overline{\mathcal{R}}_{s\to t}$ 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 $\mathcal{Q} > V \cdot \overline{ETX}$. When $\mathcal{Q} \leq V \cdot \overline{ETX}$, the source node is waiting either for the queue $\mathcal{Q}$ to grow 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 with fixed $\overline{ETX}$. In this case, the threshold is static with value $V \cdot \overline{ETX}$. Due to the forwarding policy of BCP, $\mathcal{Q}$ will be lower bounded by $V \cdot \overline{ETX}$. Under FIFO, the average delay is $D = \mathcal{Q}/\lambda \geq V \cdot \overline{ETX}/\lambda$ 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, and reaches a high value when the load λ is low. As 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 head of the queue. These trapped packets will stay in the queue forever and cannot be transmitted while all the other packets transiting in the queue will be transmitted. Ignoring the trapped packets and assuming exponentially distributed service time, the Markov chain describing the evolution of the number of packets in the queue will be the same as that of an M/M/1 queue in the lossless case.

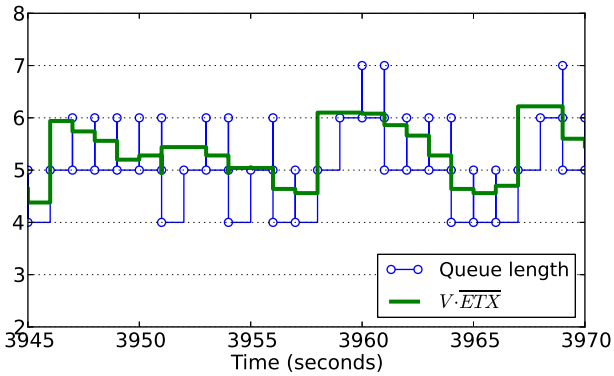


Fig. 3. Evolution of queue length and ETX of BCP over time with $V = 2$ from simulation. This illustrates the role of $V \cdot \overline{ETX}$ as the threshold on the queue.

Next, 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 $[V \cdot \overline{ETX}_{min}, V \cdot \overline{ETX}_{max}]$. Then $\mathcal{Q}$ will be lower bounded by $V \cdot \overline{ETX}_{min}$ due to the threshold range. Under FIFO, the average delay is $D = \mathcal{Q}/\lambda \geq V \cdot \overline{ETX}_{min}/\lambda$. However, under LIFO, the $V \cdot \overline{ETX}_{min}$ packets in the head of queue are trapped forever and the rest of the queue will be equivalent of a queue with dynamic threshold in the range of $[0, V \cdot \overline{ETX}_{max} - V \cdot \overline{ETX}_{min}]$. 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 $[0, 4]$. A snapshot of simulation (Fig. 3) illustrates the threshold effect of ETX on the queue length.

We note that the number of the ignored packets is a constant $V \cdot \overline{ETX}_{min}$. Therefore, the impact of the ignored packets on the packet delivery rate is negligible when the system is run for a long time, as also shown by our numerical results in Section VI-C.

B. Queueing model

Now we construct a queueing model with dynamic threshold based on the routing policy of LIFO-backpressure. Assume that the arrival process of packets is Poisson with rate λ. The channel is represented by the Gilbert model [18], a Markov chain that transits between two states, namely, 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 a positive integer 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, c) ($n \in \mathbb{N}, c \in \{0, 1\}$) denote the system states: n is the number of packets in the queue; $c = 0$ and

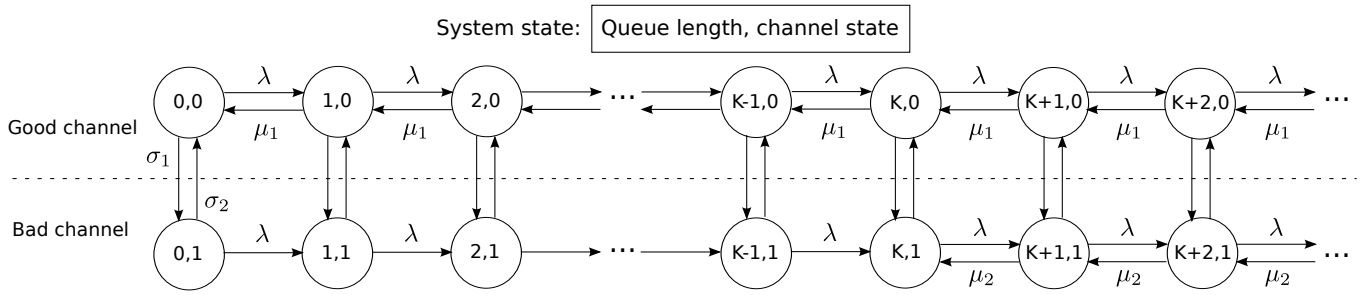


Fig. 4. Markov chain of the single-queue system.

$c = 1$ represent good channel and bad channel, 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. This channel model also *implicitly* captures the effect of interferences between nodes, whereas good and bad channels respectively imply low and high levels of interferences. We also note that under a lossless channel with fixed threshold, the system state has transitions restricted to the half of the Markov chain corresponding to good channel, which is the same as $M/M/1$.

C. Probability generating function

The steady-state distribution of the CTMC is denoted by $P_{n,c}$. Define $P_n \triangleq [P_{n,0}, P_{n,1}]^T$. Then the steady-state distribution of the queue length is

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

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

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

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

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

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:

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

$$= \frac{d}{dz}\mathcal{F}_N(z)\Big|_{z=1}, \tag{5}$$

$$\mathbb{E}[T] = \mathbb{E}[N]/\lambda. \tag{6}$$

Next we develop a matrix geometric method [6] for solving the steady-state distribution of the CTMC and calculating the average delay by (4) and (6). In Section III-E, 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 the special case $K = 1$.

D. 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:

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

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 $(n,1)$ $(1 \le n < K)$ are:

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

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

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

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:

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

In summary, the balance equations are the following:

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

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:

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

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:

$$P_n = \begin{cases} T_1 P_1, & \text{if } n = 0, \\ \mathcal{R}_1^{n-1} P_1, & \text{if } 1 \le n < K, \\ \mathcal{R}_2^{n-K} T_2 P_1, & \text{if } n \ge K. \end{cases} \tag{11}$$

The sum of P_n from 0 to ∞ is

$$\begin{bmatrix} \sum_{n=0}^{\infty} P_{n,0} \\ \sum_{n=0}^{\infty} P_{n,1} \end{bmatrix} = (T_1 + \sum_{n=1}^{K-1} \mathcal{R}_1^{n-1} + \sum_{n=K}^{\infty} \mathcal{R}_2^{n-K} T_2) P_1.$$

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

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

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

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

A sufficient condition to satisfy (13) is

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

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.

[6] has shown that by starting with $\mathcal{R}_{1(0)} = 0$, the sequence $\{\mathcal{R}_{1(0)}, \mathcal{R}_{1(1)}, \mathcal{R}_{1(2)}, ...\}$ 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

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

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 $K \ge 2$. The average packet delay when $K = 1$ can also be numerically computed using the matrix geometric method with minor change. Numerical results obtained by the matrix geometric method will be presented in Section VI.

E. z-transform method for special case

Since BCP updates ETX by the exponential moving weighted average and the default α in BCP implementation is 0.9, the change in the threshold $V \cdot \overline{ETX}$ every time is relatively small (e.g., +1/-1), which can also be seen from Fig. 3. Thus we analyze the special case where the threshold varies between 0 and 1 ($K = 1$).

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

$$\begin{cases} (\Lambda + Q)P_0 = M_1 P_1, & (16) \\ (\Lambda + M_1 + Q)P_1 = \Lambda P_0 + M_2 P_2, & (17) \\ (\Lambda + M_2 + Q)P_n = \Lambda P_{n-1} + M_2 P_{n+1}, \text{for } n \ge 2. & (18) \end{cases}$$

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),

$$\begin{aligned} (\Lambda + M_2 + Q)[\mathcal{G}(z) - P_0] = z(M_2 - M_1)P_1 + \Lambda z \mathcal{G}(z) \\ + M_2 \frac{1}{z}[\mathcal{G}(z) - P_0 - zP_1]. \end{aligned}$$

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

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

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

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

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

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

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

$$\mathbf{e}^T \mathsf{adj}\mathcal{A}(z_0)\mathcal{B}(z_0)P_1 = 0. \tag{22}$$

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

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

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

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

where $E_1 = \lambda\mu + 4\lambda\sigma + 2\mu\sigma - (\lambda + 2\sigma)E_2 + 3\lambda^2 + 4\sigma^2$ and $E_2 = \sqrt{(\lambda + \mu + 2\sigma)^2 - 4\mu\lambda}$.

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

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

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

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

By (25), the first term in the series (which is independent of λ) is sensitive to channel dynamics as captured by the parameter σ. If σ is small (i.e., the channel and threshold are slowly varying), then the average delay at low load (i.e., $\lambda \to 0$) may get high. For example, suppose that the average service time $1/\mu$ is on the order of dozens of milliseconds while the average time that the channel stays in the same state $1/\sigma$ is on the order of a few seconds. Then, the average delay will be on the order of seconds. This stands in contrast to the lossless case where the average delay is on the order of milliseconds. Furthermore, the first-order derivative with respect to λ of the average delay 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.

An intuitive explanation of the delay behavior is that the packet at the head of queue is stuck when the threshold is 1 (bad state) and only gets served when the threshold returns to 0 (good state). Thus, the queueing delay of the stuck packet is mostly determined by the transition time of the threshold from 1 to 0, which can be large when the channel is slowly varying. As the traffic load increases, the proportion of stuck packets decreases, thus reducing the overall average delay.

As discussed in Section III-A, the number of undelivered (trapped) packets in the queue is a constant, which is negligible over the long run. However, the probability that an arriving packet finds the threshold set to 1 is about $\sigma_1/(\sigma_1 + \sigma_2)$ at low load. Hence, a large number of packets may experience very high delay.

IV. Network analysis

In this section, we analyze the average packet delay of LIFO-backpressure for a chain network at low load. For the network analysis, we use similar notations as in Section III.

Consider a chain network consisting of $N+1$ nodes, labelled $\{0, 1, 2, .., N\}$. There is a bidirectional wireless channel i between node i and node $i - 1$. We assume that exogenous packets arrive into node N and are routed towards node 0 by LIFO-backpressure. The lossy channel has a dynamic threshold ($V \cdot \overline{ETX}$) varying between $V \cdot \overline{ETX}_{min}$ and $V \cdot \overline{ETX}_{max}$ as in the Gilbert model while the lossless channel has a fixed threshold $V \cdot \overline{ETX}_{min}$. For ease of discussion, we assume that $V \cdot \overline{ETX}_{min}$ and $V \cdot \overline{ETX}_{max}$ are positive integers. The service time (transmission time) of each channel is assumed to be exponentially distributed. We are interested in the average packet delay of LIFO-backpressure at low load, i.e., when $\lambda \to 0$. Before analyzing the delay, we characterize the steady-state queue occupancy at each node.

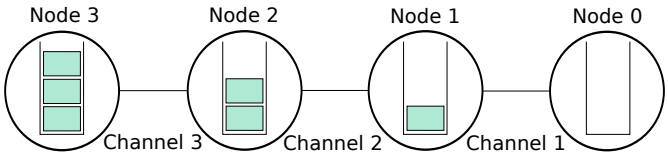


Fig. 5. The steady state of a four-node chain network under LIFO-backpressure with $V \cdot \overline{ETX}_{min} = 1$.

Lemma 1. *As $\lambda \to 0$, the steady-state queue occupancy in the chain network under LIFO-backpressure is almost surely (a.s.)*

$$\mathcal{Q}_i = iV \cdot \overline{ETX}_{min}, \forall i = 1, 2, ..., N, \quad (26)$$

for both lossless and lossy channel models.

Proof: See the Appendix. ■

Fig. 5 illustrates the steady state of a four-node chain network under LIFO-backpressure, where $V \cdot \overline{ETX}_{min} = 1$. These packets will stay in the network forever while all other packets will reach node 0 within a finite time a.s. As explained in the proof of Lemma 1, there can be at most one untrapped packet in the network at any time when $\lambda \to 0$. The packet will go through N hops to get delivered: $N \to N - 1 \to N - 2 \to ... \to 1 \to 0$. *If we ignore the trapped packets in all the queues*, then, under lossless channel, each queue is equivalent of a queue with threshold 0. Under lossy channel, each queue is equivalent of a queue with dynamic threshold in the range of $[0, K]$, where $K = V \cdot \overline{ETX}_{max} - V \cdot \overline{ETX}_{min}$.

Next, we detail the models for the lossless and lossy channels. In the lossless channel model, we assume that the thresholds of all the channels, $1, 2, ..., N$, are fixed to 0 and the packet service time is exponentially distributed with rate μ. In the lossy channel model, the channels of each node are identically and independently represented by a Gilbert model, a Markov chain that transits between 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 channel 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 (with $\mu_1 \geq \mu_2$).

As $\lambda \to 0$, a packet arriving to the network sees each queue in steady state. Thus, for the lossless channel case, the average delay per hop is $1/\mu$ and the total average packet delay is N/μ.

For the lossy channel case, suppose a new packet arrives to node i. Let (n, c) $(n, c \in \{0, 1\})$ denote the possible system states for node i. Here, n corresponds to the number of packets in the queue of node i, while $c = 0$ and $c = 1$ respectively represent good and bad states of the channel at node i. State

transitions can be described by the first two columns of the Markov chain in Fig. 4.

Just after the arrival of a new packet, the system can either be in state $(1,0)$ (i.e., one packet and good channel) with probability $\sigma_2/(\sigma_1+\sigma_2)$, or in state $(1,1)$ (one packet and bad channel) with probability $\sigma_1/(\sigma_1+\sigma_2)$. The system transits to the state of no packets and good channel, i.e., (0,0), when the packet gets transmitted to node $i-1$. Thus the delay that the packet experiences in the queue is the time that the system needs to transit from either $(1,0)$ or $(1,1)$ to $(0,0)$, which we denote by T_1 and T_2, respectively. Let T_0 denote the time taken for transition from the current state to the next state.

If the system is at state $(1,0)$ after the packet arrival, the average delay of the packet is

$$\begin{aligned}\mathbb{E}[T_1] &= \mathbb{E}[T_0] + Pr\{\text{next state} = (1,1)\}\mathbb{E}[T_2] \\ &\quad + Pr\{\text{next state} = (0,0)\}\cdot 0 \\ &= \frac{1}{\mu_1+\sigma_1} + \frac{\sigma_1}{\mu_1+\sigma_1}\mathbb{E}[T_2]. \end{aligned} \tag{27}$$

On the other hand, if the system is at state $(1,1)$ after the packet arrival, the average delay of the packet is

$$\begin{aligned}\mathbb{E}[T_2] &= \mathbb{E}[T_0] + Pr\{\text{next state} = (1,0)\}\mathbb{E}[T_1] \\ &= \frac{1}{\sigma_2} + \mathbb{E}[T_1]. \end{aligned} \tag{28}$$

Solving (27) and (28), we have

$$\mathbb{E}[T_1] = \frac{\sigma_1+\sigma_2}{\sigma_2\mu_1}, \quad \mathbb{E}[T_2] = \frac{\sigma_1+\sigma_2+\mu_1}{\sigma_2\mu_1}.$$

Therefore, the expected packet delay for one hop at low load is

$$\begin{aligned}\mathbb{E}[T] &= \frac{\sigma_2}{\sigma_1+\sigma_2}\mathbb{E}[T_1] + \frac{\sigma_1}{\sigma_1+\sigma_2}\mathbb{E}[T_2] \\ &= (1+\frac{\sigma_1}{\sigma_2})\frac{1}{\mu_1} + \frac{\sigma_1}{\sigma_2(\sigma_1+\sigma_2)}. \end{aligned} \tag{29}$$

Assuming $\sigma_1 = \sigma_2 = \sigma$ and $\mu_1 = \mu$, the average packet delay per hop is $2/\mu + 1/2\sigma$. Thus, by the linearity of expectation, the total average packet delay under lossy channel is $N(2/\mu + 1/2\sigma)$. As in the two-node network case, we observe that the delay gets high if σ is small, i.e., the channel dynamics are slow. Note that channel contention between different neighbors does not occur because the analysis is for low load. However, our simulations in Section VI do capture wireless interference effects.

V. Replication-based LIFO-backpressure

A. Algorithm description

Our previous analysis indicates that the high delay of LIFO-backpressure at low load is due to threshold dynamics. Consider again a scenario where the threshold varies between 0 and 1. The idea underlying RBL is as follows: when a packet at the tail gets stuck due to threshold increasing from 0 to 1, we do not wait indefinitely till the threshold returns to 0. Rather, after a certain period of time, we generate a replica of the stuck packet and place it at the head of the queue. Note that due to the LIFO scheduling policy, replicas have lower priority than the original packets for transmission. Based on the routing policy of LIFO-backpressure, the original packet can be served immediately since the queue length is larger than 1 after adding the replica. Algorithm 2 gives the pseudo-code of RBL-BCP, an implementation of RBL.

Algorithm 2 RBL-BCP

```
1:  REP_FLAG = false // This flag indicates whether a replica has been generated or not
2:  while Q_i > 0 do
3:      Compute the backpressure weight w_{i,j} for each neighbor j
4:      Find the neighbor j* such that j* = arg max_j w_{i,j}
5:      if w_{i,j*} > 0 then
6:          Transmit one packet to j*
7:          Update ETX_{i→j*} and R_{i→j*}
8:          REP_FLAG = false
9:      else
10:         if ⌊ETX_{i→j*}⌋ has increased and Q_{j*} has not increased and REP_FLAG is false then
11:             call Replicate
12:         end if
13:         Wait for a reroute period
14:     end if
15: end while
16:
17: function REPLICATE
18:     Wait for a replication period; exit upon packet arrival
19:     Copy the packet at the tail of queue
20:     Place the replica at the head of queue
21:     REP_FLAG = true
22: end function
```

In general, the threshold at a node depends both on the queue length of its neighbors and on the link quality to its neighbors. Therefore, the threshold may increase under two scenarios: (1) the queue length of the selected neighbor has increased; (2) the ETX to the selected neighbor has increased. In the former case, adding packets to the network would only exacerbate congestion. In the latter case, however, replication may help. As a result, RBL works as follows: if packets gets stuck in the queue solely due to increasing ETX (see line 10), then a packet at the tail of the queue is replicated onto the head.

The replication is delayed by a certain amount of time, the *replication period*, to avoid congesting the network (see line 18). More precisely, if new packets arrive to the queue during the replication period, then the replica generation is cancelled. In Section VI, we will show how the length of the replication period affects the number of generated replicas and delay performance. In the case that the threshold increases by more than one, RBL-BCP generates only one replica to avoid causing congestion. A binary variable REP_FLAG is used for this purpose. As explained in Section III-E, fluctuations in the threshold $V \cdot \overline{ETX}$ are generally small and this situation is uncommon.

Under static channel conditions, where the ETX remains constant, the replication condition will never be satisfied and there will be no replicas generated. In these cases, the RBL-BCP reverts to the original BCP, meaning that the replication mechanism does not affect the delay performance of BCP for lossless channels.

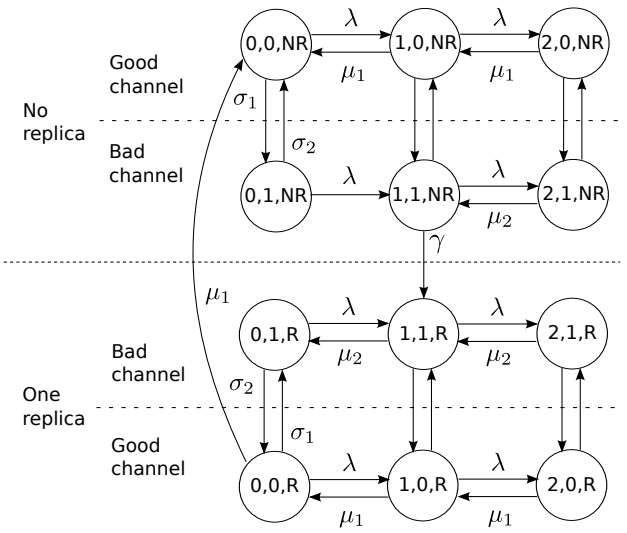


Fig. 6. Truncated Markov chain of RBL for low load analysis

B. Analysis of RBL

We analyze the average delay of RBL on the two-node network when the threshold varies between 0 and 1, i.e., $K = 1$. Note that there could be at most one replica in the queue when $K = 1$. Let the tuple (n, c, r) ($n \in \mathbb{N}, c \in \{0, 1\}, r \in \{NR, R\}$) denote the system states: n is the number of *original packets* in the queue (excluding replicas); $c = 0$ and $c = 1$ represents good channel and bad channel, respectively; $r = NR$ means that there is no replica in the queue and $r = R$ means that there is one replica in the queue. Note that the queue length (i.e., the number of packets including replicas) is $n + I_{\{r=R\}}$, where $I_{\{\bullet\}}$ is the indicator function. Fig. 6 depicts a truncated Markov chain of the queueing system under RBL.

In RBL, a replica is generated whenever packets get stuck in the queue due to threshold increasing. Since the threshold varies between 0 and 1, the only possible stuck packet is the packet at the head. When the packet gets stuck, i.e., at state $(1, 1, NR)$, RBL waits for a replication period, which is assumed to be an exponential random variable with rate γ. If there is no packet arrival during the replication period, RBL generates a replica and the system transits to state $(1, 1, R)$. Since the replica is put in the queue head, the LIFO scheduler will serve the original packets first. The system transits from the states with one replica to the states with no replica only at $(0, 0, R)$, where the threshold is 0 and the replica is the only packet in the queue.

Next, we calculate the average delay based on the average number of original packets in the queue. We do not use average queue length because the delay of the replica does not contribute to the average delay and is thus ignored. Let the steady-state distribution of the CTMC be denoted by $P_{n,c,r}$. Then the steady-state distribution of number of original packets is $\pi_n = P_{n,0,NR} + P_{n,1,NR} + P_{n,1,R} + P_{n,0,R}$.

The same as LIFO-backpressure, the CTMC of RBL has infinite number of states. However, we can still use the truncated CTMC of 12 states in Fig. 6 and obtain correct first-order Maclaurin series. To see this, note that we have the following balance equation by considering states $(0, 0, NR)$ and $(0, 1, NR)$ together,

$$\lambda(P_{0,0,NR}+P_{0,1,NR}) = \mu_1 P_{1,0,NR}+\mu_1 P_{0,0,R} \geq \mu_1 P_{1,0,NR},$$

since $P_{0,0,NR} + P_{0,1,NR} \leq 1$, we have $P_{1,0,NR} = O(\lambda)$.

Another balance equation is

$$\lambda P_{0,1,NR} + (\sigma_1 + \lambda)P_{1,0,NR} = \mu_1 P_{2,0,NR} + \sigma_2 P_{1,1,NR} + \mu_1 P_{0,0,R} \geq \sigma_2 P_{1,1,NR}.$$

Hence $P_{1,1,NR} = O(\lambda)$.

We also have

$$\lambda\pi_0 = \mu_1 P_{1,0,NR} + \mu_2 P_{1,1,R} + \mu_1 P_{1,0,R} \geq \mu_2(P_{1,0,NR} + P_{1,1,R} + P_{1,0,R}).$$

Thus $P_{1,0,NR} + P_{1,1,R} + P_{1,0,R} = O(\lambda)$ and further $\pi_1 = O(\lambda)$.

Similarly, from the balance equations we can obtain $\pi_2 = O(\lambda^2)$ and $\pi_n = O(\lambda^n)$ for $n \geq 3$.

The average delay is thus

$$\mathbb{E}(T) = \frac{\mathbb{E}(N)}{\lambda} = \frac{1}{\lambda}\sum_{n=0}^{\infty} n\pi_n = \frac{1}{\lambda}(\pi_1 + 2\pi_2 + \sum_{n=3}^{\infty} n\pi_n),$$

in which

$$\sum_{n=3}^{\infty} n\pi_n \leq \sum_{n=3}^{\infty} n\lambda^n = \frac{3\lambda^3 - 2\lambda^4}{(1-\lambda)^2} = O(\lambda^3).$$

Then the average delay can be calculated by

$$\mathbb{E}(T) = \frac{1}{\lambda}(\pi_1 + 2\pi_2) + O(\lambda^2).$$

This indicates that we can assign zero probability to system states with queue length over 2 and obtain the same first-order Maclaurin series, justifying the truncation method.

After obtaining the steady-state distribution of the truncated CTMC, we can calculate the average delay by

$$\begin{aligned}\mathbb{E}(T) &= \frac{1}{\lambda}(\pi_1 + 2\pi_2) \\ &= \frac{1}{\lambda}[P_{1,0,NR} + P_{1,1,NR} + P_{1,1,R} + P_{1,0,R} \\ &\quad + 2(P_{2,0,NR} + P_{2,1,NR} + P_{2,1,R} + P_{2,0,R})]. \quad (30)\end{aligned}$$

Assuming that $\sigma_1 = \sigma_2 = \sigma$, $\mu_1 = \mu_2 = \mu$, the average delay of RBL at low load is approximated by

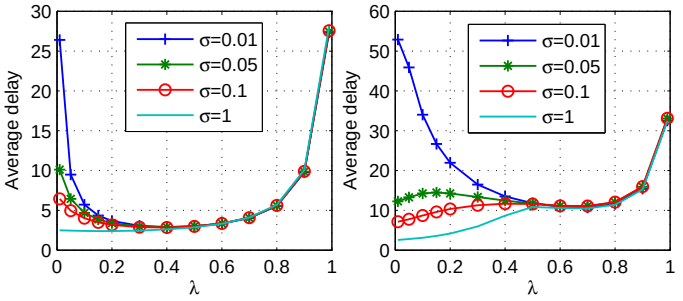
$$\begin{aligned}\mathbb{E}(T) = &\left[\frac{\frac{\mu}{2}+\sigma}{\gamma(\mu+\sigma)+\mu\sigma}+\frac{1}{\mu}\right]+\left[\frac{1}{\mu^2}-\frac{\sigma(\mu-2\sigma)}{2(\gamma(\mu+\sigma)+\mu\sigma)^2}\right. \\ &\left.-\frac{\mu^3+4\mu^2\sigma+6\mu\sigma^2+8\sigma^3}{2\mu\sigma(\mu+2\sigma)(\gamma(\mu+\sigma)+\mu\sigma)}\right]\lambda + o(\lambda). \quad (31)\end{aligned}$$

When $\gamma \to 0$,

$$\lim_{\gamma\to 0}\mathbb{E}(T) = (\frac{2}{\mu}+\frac{1}{2\sigma}) - \frac{(\mu+4\sigma)(\mu+\sigma)}{2\mu\sigma^2(\mu+2\sigma)}\lambda + o(\lambda),$$

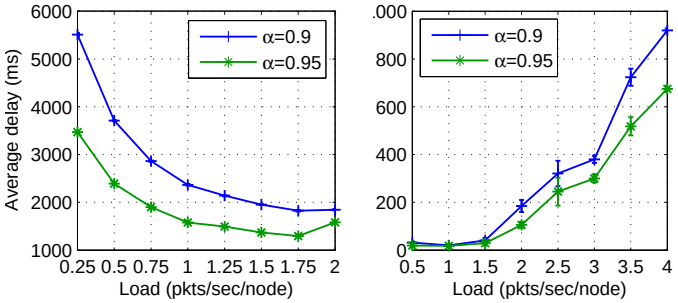
which is the same as the original LIFO-backpressure.

From (31), $\mathbb{E}(T)|_{\lambda=0}$ is monotonically decreasing with increasing γ. Thus with $\gamma > 0$ and λ being small enough, the average delay of RBL is smaller than that of the original LIFO-backpressure.



(a) $K = 1$ (b) $K = 10$

Fig. 7. Average packet delay versus packet arrival rate (traffic load) λ for different threshold transition rates σ and fixed service rate μ = 1.



(a) Lossy channel (Received power = -80 dBm) (b) Lossless channel (Received power = -75 dBm)

Fig. 8. Average delay versus load with fixed noise power -85 dBm in a 25-node grid network.

VI. Numerical and simulation results

In this section, we provide numerical results obtained by the matrix geometric method described in Section III. We also provide simulation results of (LIFO-)BCP to verify the existence of the high delay of LIFO-backpressure at low load in large networks. Simulation results comparing BCP and RBL-BCP are also provided.

A. Numerical results

Numerical results for the average packet delay in the two-node queueing model are depicted in Fig. 7 (a) and (b), for the cases $K = 1$ and $K = 10$ respectively. For $K = 1$, the delay at $\lambda \to 0$ increases as σ gets smaller, which is consistent with the first term of the analytical result derived using the z-transform method in Eq. (25). In addition, the average delay decreases with λ at low load, as predicted by the negative first-order derivative of (25).

For $K = 10$, the results are qualitatively similar to the case $K = 1$ when σ is small (e.g., $\sigma = 0.01$). However, with faster channel dynamics (e.g., $\sigma = 1$), we observe that the delay is small at low traffic load and increases with λ. As pointed out in Section III-E, fluctuations in the threshold are generally small (e.g., +1/-1) and, therefore, the case $K = 1$ appears more realistic.

Finally, we note that for both the cases $K = 1$ and $K = 10$, all the curves merge as $\lambda \to 1$. This means that temporal channel dynamics do not have as much effect at high load.

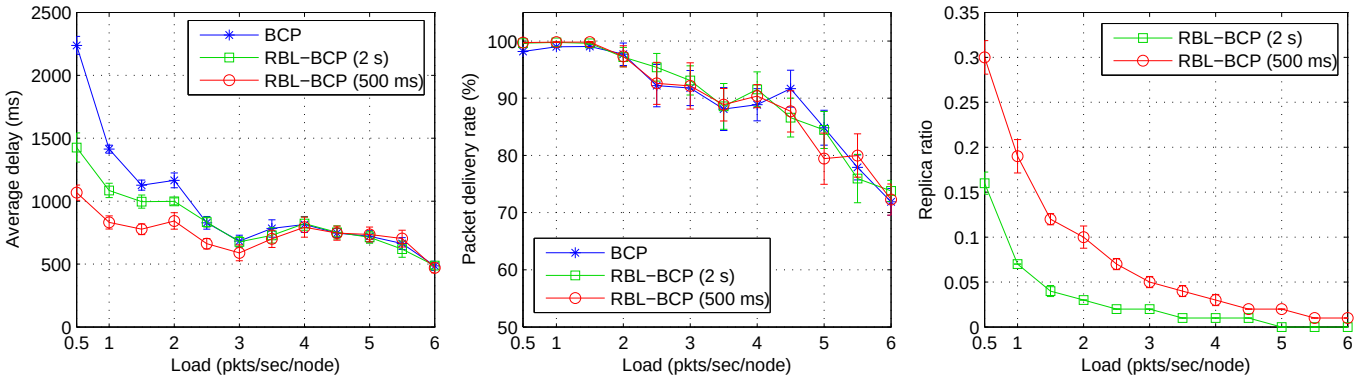
B. Simulation results of BCP

We next describe simulations of the BCP protocol (with $V = 2$). 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 [19], 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 [20]. 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 [21]. 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 high delay at low traffic load and initial decrease of the delay with load occurs under bursty channel conditions, but not under perfect channel conditions.

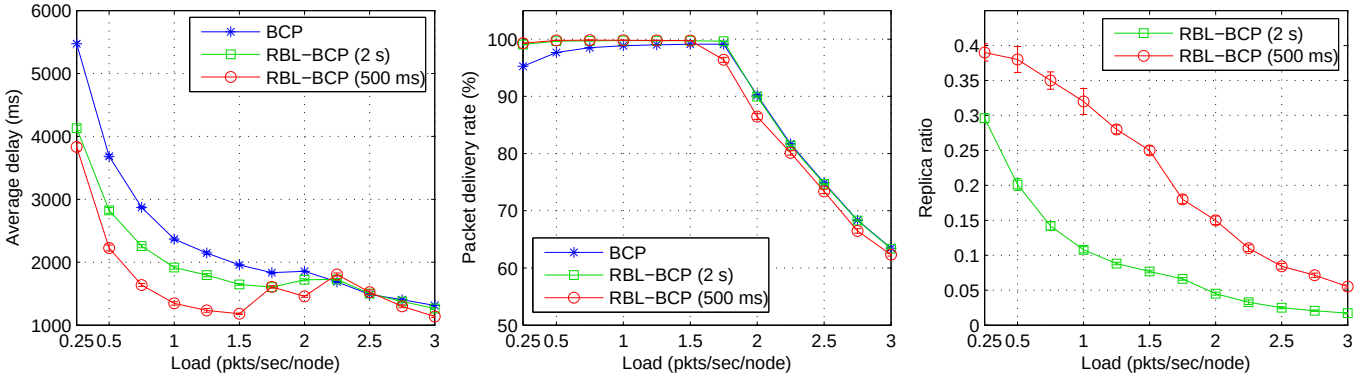
Our second set of simulations are conducted for a network consisting of 24 sensor nodes and one root node. The topology is a 5×5 grid where the root node is located at the center. In this topology, a link only exists between direct neighbors. In other words, nodes that are two hops away cannot hear each 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.

Fig. 8 shows results for the two different received powers under different α values. Recall that BCP updates ETX by $\overline{ETX}_{new} = \alpha\overline{ETX}_{old} + (1-\alpha)ETX$. Each point represents an average taken over 10 simulations, and 95% confidence intervals are also depicted. In Fig. 8(a), when the channel is lossy and the threshold is dynamic, the average delay is as high as 3710.92 ms at 0.5 pkts/sec/node and decreases with the load. In Fig. 8(b), on the other hand, when the channel is lossless and the threshold is static, the average delay is only 31.46 ms at 0.5 pkts/sec/node and is non-decreasing with the load. 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 behavior 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.



(a) Average delay versus load (b) Packet delivery rate versus load (c) Replica ratio of RBL-BCP versus load

Fig. 9. Performance of BCP and RBL-BCP on a simulated 15-node intra-car wireless sensor network.



(a) Average delay versus load (b) Packet delivery rate versus load (c) Replica ratio of RBL-BCP versus load

Fig. 10. Performance of BCP and RBL-BCP in the 25-node grid network.

C. Simulation results of RBL-BCP

In the implementation of RBL-BCP, we design the replication period to be an exponential random variable with a certain average value. In the following, we compare three protocols: (1) original BCP; (2) RBL-BCP with average replication period 2 s; (3) RBL-BCP with average replication period 500 ms. In addition to evaluating average delay, we also consider two other important metrics: the packet delivery rate, i.e., the ratio of the number of delivered original packets to the number of generated original packets; and the replica ratio, i.e., the ratio of the number of generated replicas to the number of generated original packets.

Our simulation models a 15-node intra-car wireless sensor network. The network consists of 15 nodes, in which the root is on the driver seat, three sensors are placed in the engine compartment, four sensors are respectively attached to the four wheels, three sensors are placed on passenger seats and the rest placed on the chassis. We use the same real noise traces as the first simulation in VI-B. The simulation results (average delay, packet delivery rate and replica ratio) are plotted in Fig. 9.

Fig. 9(a) shows that at low load, the average delay of RBL-BCP is much lower than BCP. For example, at a load of 0.5 pkts/sec/node, the average delay of BCP is 2,237 ms while that of RBL-BCP (500 ms) is 1,067 ms, which is almost a two-fold improvement. From Fig. 9(c), the number of replicas generated with replication period 2 s is less than with replication period 500 ms, as expected. With more replicas generated, the reduction on the delay with replication period 500 ms is larger than with replication period 2 s. Since transmissions of more replicas consumes more power, this could be viewed as a tradeoff between power consumption and delay. From Fig. 9(c), we can also see that the number of replicas vanishes as load increases, which confirms that the replication mechanism does not further congest the network at high load. This can also be verified by the observation that the average delays (Fig. 9(a)) and packet delivery rates (Fig. 9(b)) of the three protocols are undistinguishable at high load.

Fig. 9(b) shows that both BCP and RBL-BCP achieve high packet delivery rates (around 98% or higher), at low load. This result indicates that the impact of trapped packets is relatively negligible. In fact, one can observe that RBL-BCP achieves higher packet delivery rates than BCP. Indeed, RBL-BCP replicates and delivers packets that would be trapped under BCP.

Our second comparison is on the same 5×5 grid network as described in the second simulation of VI-B. We fix the noise power to be -85 dBm and the received signal power is configured to be -80 dBm. The simulation results are plotted in Fig. 10, which are similar with the first simulation on the 15-node intra-car wireless sensor network.

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 that showcases the high delay problem due to channel dynamics. The results were extended to a chain network, in a low load regime. Through simulations, we further verified the existence and significance of the channel-sensitive delay behavior of LIFO-backpressure in large networks. Therefore, an important finding of this paper was to show that, under lossy channel conditions, LIFO-based backpressure may suffer from similar high delay issues at low load as FIFO-based backpressure. An intuitive explanation for the high delay is that some packets get stuck in the queue due to the threshold increase (i.e., the channel quality degrades). These packets have to wait until the threshold decreases (i.e., the channel quality improves) to get transmitted.

To remedy the high-delay problem, we proposed a lightweight replication-based LIFO-backpressure (RBL) algorithm, which improves delay performance without compromising high throughput performance. RBL comes, however, at the expense of additional traffic transmissions at low load. Hence, achieving an optimal *power/delay* tradeoff could be an interesting direction to explore. Besides generating replicas like RBL, another possible solution for improving the delay performance of LIFO-backpressure at low load is to inject encoded packets. This problem is also left as an interesting area for future work.

APPENDIX
PROOF OF LEMMA 1

Proof: We first prove that in the described state, none of the packets in the network can leave it. Consider the packets at node i, whose queue length is $\mathcal{Q}_i = iV \cdot \overline{ETX}_{min}$. Node i has two neighbors, node $i+1$ and node $i-1$. The backpressure weight of node $i+1$ is

$$\begin{aligned} w_{i,i+1} &= (\mathcal{Q}_i - \mathcal{Q}_{i+1} - V \cdot \overline{ETX}_{i\to i+1}) \cdot \overline{\mathcal{R}}_{i\to i+1} \\ &\le (iV \cdot \overline{ETX}_{min} - (i+1)V \cdot \overline{ETX}_{min} \\ &\quad - V \cdot \overline{ETX}_{min}) \cdot \overline{\mathcal{R}}_{i\to i+1} < 0, \end{aligned}$$

for both lossless and lossy channels. Thus the transmission condition of BCP is not satisfied and the packets of node i can not be transmitted to node $i+1$. The backpressure weight of node $i-1$ is

$$\begin{aligned} w_{i,i-1} &= (\mathcal{Q}_i - \mathcal{Q}_{i-1} - V \cdot \overline{ETX}_{i\to i-1}) \cdot \overline{\mathcal{R}}_{i\to i-1} \\ &\le (iV \cdot \overline{ETX}_{min} - (i-1)V \cdot \overline{ETX}_{min} \\ &\quad - V \cdot \overline{ETX}_{min}) \cdot \overline{\mathcal{R}}_{i\to i-1} = 0. \end{aligned}$$

Similarly, the packets can not be transmitted to node $i-1$, either.

Second, we prove that in the described state, any new packet arriving into this network will leave the network within a finite time a.s. Assume that a new packet arrives to node N and sees the network in the steady state. Then,

$$\begin{aligned} w_{N,N-1} &= (\mathcal{Q}_N - \mathcal{Q}_{N-1} - V \cdot \overline{ETX}_{N\to N-1}) \cdot \overline{\mathcal{R}}_{N\to N-1} \\ &= ((NV \cdot \overline{ETX}_{min} + 1) - (N-1)V \cdot \overline{ETX}_{min} \\ &\quad - V \cdot \overline{ETX}_{N\to N-1}) \cdot \overline{\mathcal{R}}_{N\to N-1} \\ &= (V \cdot \overline{ETX}_{min} + 1 - V \cdot \overline{ETX}_{N\to N-1}) \cdot \overline{\mathcal{R}}_{N\to N-1}. \end{aligned}$$

For the lossless case, since the channel threshold $V \cdot \overline{ETX}_{N\to N-1} = V \cdot \overline{ETX}_{min}$, the backpressure weight will be positive and the transmission condition is satisfied. Then the packet will be transmitted to node $N-1$, with the service time being exponentially distributed. For the lossy case, since the channel threshold $V \cdot \overline{ETX}_{N\to N-1}$ will return to $V \cdot \overline{ETX}_{min}$ within a finite time a.s. and the service time is exponentially distributed, the packet will move to node $N-1$ within a finite time a.s. Likewise, the packet will experience finite delay a.s. at the next hops before leaving the network. As $\lambda \to 0$, the next packet arrival to the network will take place a.s. after the current packet leaves. Therefore, all the packets arriving to the network see the system in the steady state and leave the network within a finite time. ■

REFERENCES

[1] W. Si and D. Starobinski, "On the channel-sensitive delay behavior of LIFO-backpressure," in *Communication, Control, and Computing (Allerton), 2013 51st Annual Allerton Conference on*.

[2] L. Georgiadis, M. J. Neely, and L. Tassiulas, *Resource Allocation and Cross-Layer Control in Wireless Networks*. Foundations and Trends in Networking, 2006.

[3] S. Moeller, A. Sridharan, B. Krishnamachari, and O. Gnawali, "Routing without routes: the backpressure collection protocol," in *IPSN*, 2010.

[4] 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, June 2013.

[5] O. Gnawali, R. Fonseca, K. Jamieson, M. Kazandjieva, D. Moss, and P. Levis, "Ctp: An efficient, robust, and reliable collection tree protocol for wireless sensor networks," *ACM Trans. Sen. Netw.*, vol. 10, no. 1, pp. 16:1–16:49, Dec. 2013.

[6] M. F. Neuts, *Matrix Geometric Solutions in Stochastic Models*. The Johns Hopkins University Press, Baltimore, 1981.

[7] J. N. Daigle, *Queueing Theory for Telecommunications*. Addison-Wesley Publishing Company, Inc., 1992.

[8] 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.

[9] M. J. Neely, *Notes on Backpressure Routing*. Course of Stochastic Network Optimization, Spring 2011.

[10] M. Alresaini, M. Sathiamoorthy, B. Krishnamachari, and M. Neely, "Backpressure with adaptive redundancy (BWAR)," in *INFOCOM*, 2012.

[11] B. Ji, C. Joo, and N. Shroff, "Delay-based back-pressure scheduling in multihop wireless networks," *Networking, IEEE/ACM Transactions on*, vol. 21, no. 5, pp. 1539–1552, Oct 2013.

[12] 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.

[13] L. Ying, S. Shakkottai, A. Reddy, and S. Liu, "On combining shortest-path and back-pressure routing over multihop wireless networks," *IEEE/ACM Trans. Netw.*, vol. 19, no. 3, pp. 841–854, Jun. 2011.

[14] L. Bui, R. Srikant, and A. Stolyar, "A novel architecture for reduction of delay and queueing structure complexity in the back-pressure algorithm," *Networking, IEEE/ACM Transactions on*, vol. 19, no. 6, pp. 1597–1609, Dec 2011.

[15] 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.

[16] T. Spyropoulos, K. Psounis, and C. S. Raghavendra, "Efficient routing in intermittently connected mobile networks: The single-copy case," *Networking, IEEE/ACM Transactions on*, vol. 16, no. 1, pp. 63–76, 2008.

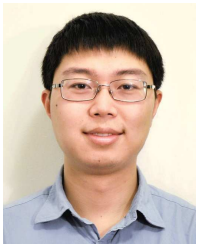
[17] ——, "Efficient routing in intermittently connected mobile networks: the multiple-copy case," *Networking, IEEE/ACM Transactions on*, vol. 16, no. 1, pp. 77–90, 2008.

[18] E. N. Gilbert, "Capacity of a burst-noise channel," *Bell System Technical Journal*, vol. 39, pp. 1253–1265, Sep. 1960.

[19] P. Levis, N. Lee, M. Welsh, and D. Culler, "TOSSIM: accurate and scalable simulation of entire TinyOS applications," in *SenSys*, 2003.

[20] M. Hashemi, W. Si, M. Laifenfeld, D. Starobinski, and A. Trachtenberg, "Intra-car wireless sensors data collection: A multi-hop approach," in *VTC*, 2013.

[21] H. Lee, A. Cerpa, and P. Levis, "Improving wireless simulation through noise modeling," in *IPSN*, 2007.



Wei Si received B.S. degree in Information Engineering from Shanghai Jiao Tong University, Shanghai, China, in 2010. Currently, he is a Ph.D. candidate in Systems Engineering at Boston University. His research interests include routing protocols for wireless sensor networks and disruption tolerant networks, data synchronization algorithms and queueing theory.

David Starobinski is a Professor of Electrical and Computer Engineering at Boston University, with a joint appointment in the Division of Systems Engineering. He is also a Faculty Fellow at the U.S. DoT Volpe National Transportation Systems Center. He received his Ph.D. in Electrical Engineering from the Technion - Israel Institute of Technology, in 1999. In 1999-2000, he was a visiting post-doctoral researcher in the EECS department at UC Berkeley. In 2007-2008, he was an invited Professor at EPFL (Switzerland). Dr. Starobinski received a CAREER award from the U.S. National Science Foundation (2002), an Early Career Principal Investigator (ECPI) award from the U.S. Department of Energy (2004), the best paper award at the WiOpt 2010 conference, and the 2010 BU ECE Faculty Teaching Award. He was an Associate Editor of the IEEE/ACM Transactions on Networking from 2009 to 2013. His research interests are in wireless networking, network economics, and cybersecurity.

Morteza Hashemi received his B.Sc. (2011) in Electrical Engineering from Sharif University of Technology, Tehran, Iran. He is currently a PhD candidate in Electrical and Computer Engineering at Boston University. His research interests include error correcting code, networks performance evaluation, and wireless communications.

Moshe Laifenfeld received his BSc ('92), MSc ('98) and Ph.D ('08) from the Technion, Tel-Aviv University and Boston University, respectively, all in electrical and computer engineering. After a joint post-doctoral position at MIT and Boston University, he joined General Motors R&D, focusing on in-vehicle wireless communications. In his past, Moshe led the algorithms development of a 3rd generation UMTS transceiver, and held several R&D positions in medical devices start-ups.

Ari Trachtenberg is a Professor of Electrical and Computer Engineering at Boston University, where he has been since September 2000. He received his PhD and MS in Computer Science (2000,1996) from the University of Illinois at Urbana-Champaign, and his SB in 1994 from MIT. He has also been a visiting professor at the Technion - Israel Institute of Technology, and worked at MIT Lincoln Lab, HP Labs, and the Johns Hopkins Center for Talented Youth, and has been awarded ECE Teaching Awards (BU, 2013/2003), a Kern fellowship (BU 2012), an NSF CAREER (BU 2002), and the Kuck Outstanding Thesis (UIUC 2000). His research interests include cyber security (smartphones, offensive and defensive), networking (security, sensors, localization); algorithms (data synchronization, file edits, file sharing), and error-correcting codes (rateless coding, feedback).