LIFO-Backpressure Achieves Near Optimal Utility-Delay Tradeoff

Longbo Huang*, Scott Moeller†, Michael J. Neely†, Bhaskar Krishnamachari†

***Abstract*—There has been considerable work developing a stochastic network utility maximization framework using Backpressure algorithms, also known as MaxWeight. A key open problem has been the development of utility-optimal algorithms that are also delay efficient. In this paper, we show that the Backpressure algorithm, when combined with the LIFO queueing discipline (called LIFO-Backpressure), is able to achieve a utility that is within $O(1/V)$ of the optimal value, for any scalar $V \geq 1$, while maintaining an average delay of $O([\log(V)]^2)$ for all but a tiny fraction of the network traffic. This result holds for a general class of problems with Markovian dynamics. Remarkably, the performance of LIFO-Backpressure can be achieved by simply changing the queueing discipline; it requires no other modifications of the original Backpressure algorithm. We validate the results through empirical measurements from a sensor network testbed, which show a good match between theory and practice.**

Because some packets may stay in the queues for a very long time under LIFO-Backpressure, we further develop the LIFO^p-Backpressure algorithm, which generalizes LIFO-Backpressure by allowing interleaving between FIFO and LIFO. We show that LIFO^p-Backpressure also achieves the same $O(1/V)$ close-to-optimal utility performance, and guarantees an average delay of $O([\log(V)]^2)$ for the packets that are served during the LIFO period.

***Index Terms*—Queueing, Dynamic Control, LIFO scheduling, Lyapunov analysis, Stochastic Optimization**

I. Introduction

Recent developments in stochastic network optimization theory have yielded a very general framework that solves a large class of networking problems of the following form: We are given a discrete time stochastic network. The network state, which describes the current realization of the underlying network randomness, such as the network channel condition, is time varying according to some probability law. A network controller performs some action based on the observed network state at every time slot. The chosen action incurs a cost, [1] but also serves some amount of traffic and possibly generates new traffic for the network. This traffic causes congestion, and thus leads to backlogs at nodes in the network. The goal of the controller is to minimize its time average cost subject to the constraint that the time average total backlog in the network be kept finite.

This general setting models a large class of networking problems ranging from traffic routing [1] and flow utility maximization [2] to network pricing [3] and cognitive radio applications [4]. Also, many techniques have also been applied to this problem (see [5] for a survey). Among the approaches that have been adopted, the family of Backpressure algorithms [6] are recently receiving much attention due to their provable performance guarantees, robustness to stochastic network conditions and, most importantly, their ability to achieve the desired performance *without requiring any statistical knowledge* of the underlying randomness in the network.

Most prior performance results for Backpressure are given in the following $[O(1/V), O(V)]$ utility-delay tradeoff form [6]: 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 these results provide strong theoretical guarantees for the algorithms, the network delay can be unsatisfying when we achieve a utility that is very close to the optimal, i.e., when V is large.

There have been efforts to develop algorithms that can achieve better utility-delay tradeoffs. Previous works [7] and [8] show improved tradeoffs are possible for single-hop networks with certain structure, and develop optimal $[O(1/V), O(\log(V))]$ and $[O(1/V), O(\sqrt{V})]$ utility-delay tradeoffs. The algorithms are different from basic Backpressure and require knowledge of an "epsilon" parameter that measures distance to a performance region boundary. Work [9] uses a completely different analytical technique to show that similar poly-logarithmic tradeoffs, i.e., $[O(1/V), O([\log(V)]^2)]$, are possible by carefully modifying the actions taken by the basic Backpressure algorithms. However, the algorithm requires a pre-determined learning phase, which adds additional complexity to the implementation. The current work, following the line of analysis in [9], instead shows that similar poly-logarithmic tradeoffs, i.e., $[O(1/V), O([\log(V)]^2)]$, can be achieved by the *original* Backpressure algorithm by simply modifying the service discipline from First-in-First-Out (FIFO) to Last-In-First-

*Longbo Huang (http://www.iiis.tsinghua.edu.cn/∼huang) is with the Institute for Theoretical Computer Science and the Institute for Interdisciplinary Information Sciences, Tsinghua University, Beijing, P. R. China. †Scott Moeller, Michael J. Neely, and Bhaskar Krishnamachari (emails: {smoeller, mjneely, bkrishna}@usc.edu) are with the Department of Electrical Engineering, University of Southern California, Los Angeles, CA 90089, USA.

This work was supported in part by one or more of the following: the National Basic Research Program of China Grant 2011CBA00300, 2011CBA00301, the National Natural Science Foundation of China Grant 61033001, 61061130540, 61073174, the NSF Career grant CCF-0747525, NSF grants 1049541, 1017881, and the Network Science Collaborative Technology Alliance sponsored by the U.S. Army Research Laboratory W911NF-09-2-0053.

This paper was presented in part at the 9th International Symposium of Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks (WiOpt), May 2011, Princeton, USA.

[1]Since cost minimization is mathematically equivalent to utility maximization, below we will use cost and utility interchangeably.

Out (LIFO) (called LIFO-Backpressure below). This is a remarkable feature that distinguishes LIFO-Backpressure from previous algorithms in [7] [8] [9] while providing a deeper understanding of Backpressure itself and clarifying the role of queue backlogs as Lagrange multipliers (see also [2] [9]). However, this performance improvement is not for free: In order to dramatically improve delay for the the majority of the traffic, *a small fraction of packets* may need to stay in the queues for a very long time (possibly forever). We prove that as the V parameter is increased, the fraction of these "trapped" packets quickly converges to zero, while maintaining $O(1/V)$ close-to-optimal utility and $O([\log(V)]^2)$ average delay. This provides an analytical justification for experimental observations of [10], in which a related Backpressure implementation demonstrates that average delay can be reduced by two orders of magnitude for 98% of traffic when LIFO queue discipline is employed.

LIFO-Backpressure was proposed in a recent empirical work [10]. Without providing theoretical guarantees, the authors developed a practical implementation of Backpressure routing and showed experimentally that applying the LIFO queuing discipline drastically improves average packet delay. Another two notable recent works providing an alternative delay solution are [11] and [12]. [11] describes a novel Backpressure-based per-packet randomized routing framework that runs atop the shadow queue structure of [13] while minimizing hop count as explored in [14]. Their techniques reduce delay drastically and eliminates the per-destination queue complexity. [12] develops a queue-based routing algorithm for intermittently connected networks, and uses a shadow-queue based approach for delay improvement. However, neither works provide $O([\log(V)]^2)$ average delay guarantees.

Our analysis of the delay performance of LIFO-Backpressure is based on the recent "exponential attraction" result developed in [9]. The proof idea can be intuitively explained by Fig. 1, which depicts a simulated backlog process of a single queue system with unit packet size under Backpressure. The left figure demonstrates the "exponential

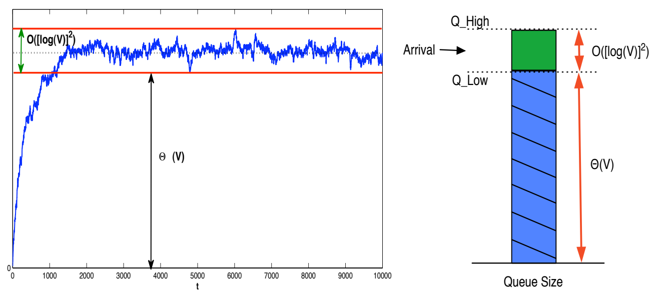

Fig. 1. The LIFO-Backpressure idea.

attraction" result in [9], which states that the queue size vector under Backpressure deviates from some fixed point with probability that decreases exponentially in the deviation distance. Hence the queue size will mostly fluctuate within the interval $[Q_{\text{Low}}, Q_{\text{High}}]$, which can be shown to be of $O([\log(V)]^2)$ size. This result holds under both FIFO and LIFO, as they result in the same queue process.

Now suppose that LIFO is used in this queue. Then, from the right side of Fig. 1, we see that because the queue mostly resides within the range of Q_{Low} and Q_{High}, most packets arrive to find the queue size in this region. These new packets will always be placed on the top of the queue due to the LIFO discipline. Most packets thus enter and leave the queue when the queue size is between Q_{Low} and Q_{High} and therefore "see" a queue with average size no more than $Q_{\text{High}} - Q_{\text{Low}} = O([\log(V)]^2)$. Now let λ be the packet arrival rate into the queue, and let $\tilde{\lambda}$ be the arrival rate of packets entering when the queue size is in $[Q_{\text{Low}}, Q_{\text{High}}]$ and that eventually depart. Because packets always occupy the same buffer slot under LIFO, we see that these packets can occupy at most $Q_{\text{High}} - Q_{\text{Low}} + \delta_{\max}$ buffer slots, ranging from Q_{Low} to $Q_{\text{High}} + \delta_{\max}$, where $\delta_{\max} = \Theta(1)$ is the maximum number of packets that can enter the queue at any time. We can now apply Little's Theorem [15] to the buffer slots in the interval $[Q_{\text{Low}}, Q_{\text{High}} + \delta_{\max}]$, and we see that average delay for these packets that arrive when the queue size is in $[Q_{\text{Low}}, Q_{\text{High}}]$ satisfies:

$$D \leq \frac{Q_{\text{High}} - Q_{\text{Low}} + \delta_{\max}}{\tilde{\lambda}} = \frac{O([\log(V)]^2)}{\tilde{\lambda}}. \tag{1}$$

The exponential attraction result implies that $\lambda \approx \tilde{\lambda}$. Hence for almost all packets entering the queue, the average delay is $D = O([\log(V)]^2/\lambda)$.

We also generalize LIFO-Backpressure to a scheme that allows interleaving between FIFO and LIFO, called LIFO^p-Backpressure. The development of LIFO^p-Backpressure is motivated by the fact that a few packets may stay in the queue for a very long time under LIFO-Backpressure. We show that LIFO^p-Backpressure is able to achieve the same $O(1/V)$ close-to-optimal utility performance, and guarantee an average delay of $O([\log(V)]^2)$ for the packets that are served during the LIFO period. Finally, we show how these results can be extended, allowing us to optimize functions of time average cost.

This paper is organized as follows. We first provide an example of our network model in Section II. We then present the general system model in Section III. We review the Backpressure algorithm in Section IV. The delay performance of LIFO-Backpressure is presented in Section V. We then present LIFO^p-Backpressure and its performance in Section VI. Simulation results and experimental testbed results are presented in Sections VII and VIII. Finally, we show how our results can be extended to optimize functions of time averages in Section IX.

II. An Example of Our System Model

To facilitate understanding of the general system model, we first provide an example in this section to illustrate the model.

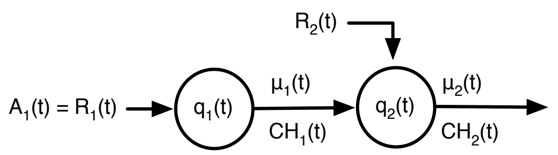

Fig. 2. A two-queue tandem example.

Consider the 2-queue network in Fig. 2. In this network, external packets enter the network from nodes 1 and 2 and will be relayed over the network. We assume time is slotted, and use $R_1(t)$ and $R_2(t)$ to denote the number of packets that arrive to nodes 1 and 2 at time t, respectively. We assume that $R_i(t) \in \{0, 1\}$ for $i = 1, 2$.

The channel conditions of the queues are time varying, e.g., due to fading. We use $CH_1(t)$ and $CH_2(t)$ to denote the channel condition of both queues. We assume that $CH_j(t) \in \{G, B\}$, where $CH_j(t) = G/B$ means that channel j has a "Good" or "Bad" state. When $CH_j(t) = G$, one unit of power can serve 2 packets over the link, otherwise it can only serve one. We assume that the quadruple $(R_1(t), R_2(t), CH_1(t), CH_2(t))$ evolves according to a finite state Markov chain with three states $(1, 1, G, B), (1, 1, G, G)$, and $(0, 0, B, G)$.

We use $\boldsymbol{x}(t) \triangleq (x_1(t), x_2(t))$ to denote the operator's power allocation action, where $x_j(t)$ denotes the amount of energy spent at queue $j = 1, 2$. We assume $x_j(t) \in \{0, 1\}$ for all j and t. Also, if $x_1(t) = 1$ but queue 1 is empty, we assume that null packets will be sent. We assume that power can be allocated to both channels without impacting the link rates. The operator's goal is to find a power allocation policy to determine $(x_1(t), x_2(t))$ for every time slot t, so as to support all arriving traffic, i.e., maintain queue stability, with minimum energy expenditure.

For ease of presenting the general model later, we now restate the example problem above in the following general form.

• The system has a *network state*:

$$S(t) \triangleq (R_1(t), R_2(t), CH_1(t), CH_2(t)),$$

which evolves according to a Markov chain with state space $\mathcal{S} \triangleq \{s_1, s_2, s_3\}$, where $s_1 = (1, 1, G, B), s_2 = (1, 1, G, G)$, and $s_3 = (0, 0, B, G)$.

• Under a network state $S(t)$, the operator has a set of *feasible* actions, i.e., $\mathcal{X}^{S(t)} \triangleq \{\boldsymbol{x} : x_1 = 0/1, x_2 = 0/1\}$, and he chooses a power allocation decision $\boldsymbol{x}(t) \in \mathcal{X}^{S(t)}$.

• Under $S(t)$ and $\boldsymbol{x}(t)$,

- The operator pays a cost $f(t) \triangleq f(S(t), \boldsymbol{x}(t)) = x_1(t) + x_2(t)$, which is the total power consumption.
- The aggregate service rate allocated to queue j is denoted by the *rate function* $\mu_j(t) \triangleq \mu_j(S(t), \boldsymbol{x}(t))$. If in $S(t)$, $CH_j(t) = G$, $\mu_j(t) = 2x_j(t)$, else $\mu_j(t) = x_j(t)$.
- The aggregate amount of traffic entering queue j is denoted by the *traffic function* $A_j(t) \triangleq A_j(S(t), \boldsymbol{x}(t))$. In the example, $A_1(t) = R_1(t)$ and $A_2(t) = R_2(t) + \mu_1(t)$.
- The queues evolve according to:

$$q_j(t + 1) = \max[q_j(t) - \mu_j(t), 0] + A_j(t), \; j = 1, 2.$$

• The goal of the operator is to minimize the time average value of $f(t)$, subject to network stability.

The network states, the traffic functions, and the service rate functions are summarized in Fig. 3. It can be observed that the functions are all continuous in the action $\boldsymbol{x}(t)$. Note here $A_1(t) = R_1(t)$ is part of $S(t)$ and is independent of $\boldsymbol{x}(t)$; while $A_2(t) = \mu_1(t)+R_2(t)$ hence depends on $\boldsymbol{x}(t)$. Also note that $A_2(t)$ equals $\mu_1(t) + R_2(t)$ instead of $\min[\mu_1(t), q_1(t)] + R_2(t)$ due to our idle fill assumption.

S(t)	$R_1(t)$	$R_2(t)$	$CH_1(t)$	$CH_2(t)$	$A_1(t)$	$A_2(t)$	$\mu_1(t)$	$\mu_2(t)$
s_1	1	1	G	B	1	$2x_1 + 1$	$2x_1$	x_2
s_2	1	1	G	G	1	$2x_1 + 1$	$2x_1$	$2x_2$
s_3	0	0	B	G	0	x_1	x_1	$2x_2$

Fig. 3. The traffic and service functions under different states.

III. System Model

In this section, we specify the general network model we use. We consider a network controller that operates a network with the goal of minimizing the time average cost, subject to the queue stability constraint. The network is assumed to operate in slotted time, i.e., $t \in \{0, 1, 2, ...\}$. We assume there are $r \geq 1$ queues in the network.

A. Network State

In every slot t, we use $S(t)$ to denote the current network state, which indicates the current network parameters, such as a vector of channel conditions for each link, or a collection of other relevant information about the current network channels and arrivals. We assume that $S(t)$ evolves according a finite state irreducible and aperiodic Markov chain, with a total of M different random network states denoted as $\mathcal{S} = \{s_1, s_2, \dots, s_M\}$. Let π_{s_i} denote the steady state probability of being in state s_i. It is easy to see in this case that $\pi_{s_i} > 0$ for all s_i. The network controller can observe $S(t)$ at the beginning of every slot t, but the π_{s_i} and transition probabilities are not necessarily known.

B. The Cost, Traffic, and Service

At each time t, after observing $S(t) = s_i$, the controller chooses an action $x(t)$ from a set $\mathcal{X}^{(s_i)}$, i.e., $x(t) = x^{(s_i)}$ for some $x^{(s_i)} \in \mathcal{X}^{(s_i)}$. The set $\mathcal{X}^{(s_i)}$ is called the feasible action set for network state s_i and is assumed to be time-invariant and compact for all $s_i \in \mathcal{S}$. The cost, traffic, and service generated by the chosen action $x(t) = x^{(s_i)}$ are as follows:

(a) The chosen action has an associated cost given by the cost function $f(t) = f(s_i, x^{(s_i)}) : \mathcal{X}^{(s_i)} \mapsto \mathbb{R}_+$ (or $\mathcal{X}^{(s_i)} \mapsto \mathbb{R}_-$ in reward maximization problems);

(b) The amount of traffic generated by the action to queue j is determined by the traffic function $A_j(t) = A_j(s_i, x^{(s_i)}) : \mathcal{X}^{(s_i)} \mapsto \mathbb{R}_+$, in units of packets;

(c) The amount of service allocated to queue j is given by the rate function $\mu_j(t) = \mu_j(s_i, x^{(s_i)}) : \mathcal{X}^{(s_i)} \mapsto \mathbb{R}_+$, in units of packets.

Note that $A_j(t)$ includes both the exogenous arrivals from outside the network to queue j, and the endogenous arrivals from other queues, i.e., the transmitted packets from other queues, to queue j. We assume the functions $f(s_i, \cdot)$, $\mu_j(s_i, \cdot)$ and $A_j(s_i, \cdot)$ are continuous, time-invariant, their magnitudes are uniformly upper bounded by some constant $\delta_{\max} \in (0, \infty)$ for all s_i, j, and they are known to the network operator. We also assume that there exists a set of actions $\{x_k^{(s_i)}\}_{i=1,\dots,M}^{k=1,2,\dots,\infty}$ with $x_k^{(s_i)} \in \mathcal{X}^{(s_i)}$ and some variables $\vartheta_k^{(s_i)} \geq 0$ for all s_i and k with $\sum_k \vartheta_k^{(s_i)} = 1$ for all s_i, such that:

$$\sum_{s_i} \pi_{s_i} \Big\{ \sum_k \vartheta_k^{(s_i)} [A_j(s_i, x_k^{(s_i)}) - \mu_j(s_i, x_k^{(s_i)})] \Big\} \leq -\eta, \quad (2)$$

for some $\eta > 0$ and for all j. That is, the stability constraints are feasible with η-slackness. Thus, there exists a stationary randomized policy that stabilizes all the queues in the network (where $\vartheta_k^{(s_i)}$ represents the probability of choosing action $x_k^{(s_i)}$ when $S(t) = s_i$) [6].

C. Queueing, Average Cost, and the Stochastic Problem

Let $\boldsymbol{q}(t) = (q_1(t), ..., q_r(t))^T \in \mathbb{R}_+^r$, $t = 0, 1, 2, ...$ be the queue backlog vector process of the network, in units of packets. We assume the following queueing dynamics:

$$q_j(t+1) = \max\big[q_j(t) - \mu_j(t), 0\big] + A_j(t), \quad \forall j, \tag{3}$$

and $\boldsymbol{q}(0) = \boldsymbol{0}$. By using (3), we assume that when a queue does not have enough packets to send, null packets are transmitted. In this paper, we adopt the following notion of queue stability:

$$\mathbb{E}\Big\{\sum_{j=1}^{r} q_j\Big\} \triangleq \limsup_{t\to\infty} \frac{1}{t}\sum_{\tau=0}^{t-1}\sum_{j=1}^{r}\mathbb{E}\big\{q_j(\tau)\big\} < \infty. \tag{4}$$

We also use f_{av}^{Π} to denote the time average cost induced by an action-choosing policy Π, defined as:

$$f_{\text{av}}^{\Pi} \triangleq \limsup_{t\to\infty} \frac{1}{t}\sum_{\tau=0}^{t-1}\mathbb{E}\big\{f^{\Pi}(\tau)\big\}, \tag{5}$$

where $f^{\Pi}(\tau)$ is the cost incurred at time τ by policy Π. We call an action-choosing policy *feasible* if at every time slot t it only chooses actions from the feasible action set $\mathcal{X}^{(S(t))}$. We then call a feasible action-choosing policy under which (4) holds a *stable* policy, and use f_{av}^{*} to denote the optimal time average cost over all stable policies.

In every slot, the network controller observes the current network state and chooses a control action, with the goal of minimizing the time average cost subject to network stability. This goal can be mathematically stated as:

$$\textbf{(P1)}\ \ \textbf{min}:\ \boldsymbol{f_{\text{av}}^{\Pi}},\ \ \text{s.t.}\ (4).$$

In the following, we call **(P1)** *the stochastic problem.*

IV. Backpressure and the Deterministic Problem

In this section, we first review the Backpressure algorithm [6] for solving the stochastic problem. Then, we define the *deterministic problem* and its dual for our later analysis. We first recall the Backpressure algorithm for utility optimization problems [6].

Backpressure: At every time slot t, observe the current network state $S(t)$ and the backlog $\boldsymbol{q}(t)$. If $S(t) = s_i$, choose $x^{(s_i)} \in \mathcal{X}^{(s_i)}$ that solves the following:

$$\begin{aligned} \max: \quad & -Vf(s_i, x) + \sum_{j=1}^{r} q_j(t)\big[\mu_j(s_i, x) - A_j(s_i, x)\big] \quad (6)\\ \text{s.t.} \quad & x \in \mathcal{X}^{(s_i)}. \qquad \diamond \end{aligned}$$

Depending on the problem structure, (6) can usually be decomposed into separate parts that are easier to solve, e.g., [3], [4]. Also, when the network state process $S(t)$ is i.i.d., it has been shown in [6] that,

$$f_{\text{av}}^{\text{BP}} = f_{\text{av}}^{*} + O(1/V), \quad \overline{q}^{\text{BP}} = O(V), \tag{7}$$

where $f_{\text{av}}^{\text{BP}}$ and $\overline{q}^{\text{BP}}$ are the expected average cost and the expected average network backlog size under Backpressure, respectively. When $S(t)$ is Markovian, it is recently formally shown in [16] that Backpressure achieves the exact $[O(V), O(1/V)]$ utility-delay tradeoff. Note that the performance results in (7) hold under Backpressure with any queueing discipline for choosing which packets to serve.

We also recall *the deterministic problem* defined in [9]:

$$\begin{aligned} \min: \quad & \mathcal{F}(\boldsymbol{x}) \triangleq V\sum_{s_i}\pi_{s_i} f(s_i, x^{(s_i)}) \qquad (8)\\ \text{s.t.} \quad & \mathcal{A}_j(\boldsymbol{x}) \triangleq \sum_{s_i}\pi_{s_i} A_j(s_i, x^{(s_i)})\\ & \qquad \leq \mathcal{B}_j(\boldsymbol{x}) \triangleq \sum_{s_i}\pi_{s_i}\mu_j(s_i, x^{(s_i)}),\ \forall j,\\ & x^{(s_i)} \in \mathcal{X}^{(s_i)} \quad \forall\, i = 1, 2, ..., M, \end{aligned}$$

where π_{s_i} corresponds to the steady state probability of $S(t) = s_i$ and $\boldsymbol{x} = (x^{(s_1)}, ..., x^{(s_M)})^T$. The dual problem of (8) can be obtained as follows:

$$\max:\ g(\boldsymbol{\gamma}), \quad \text{s.t. } \boldsymbol{\gamma} \succeq \boldsymbol{0}, \tag{9}$$

where $g(\boldsymbol{\gamma})$ is called the dual function and is defined as:

$$\begin{aligned} g(\boldsymbol{\gamma}) = \inf_{x^{(s_i)} \in \mathcal{X}^{(s_i)}} \sum_{s_i}\pi_{s_i}\Big\{ & Vf(s_i, x^{(s_i)}) \qquad (10)\\ & + \sum_j \gamma_j\big[A_j(s_i, x^{(s_i)}) - \mu_j(s_i, x^{(s_i)})\big]\Big\}. \end{aligned}$$

Here $\boldsymbol{\gamma} = (\gamma_1, ..., \gamma_r)^T$ is the *Lagrange multiplier* of (8). It is well known that $g(\boldsymbol{\gamma})$ in (10) is concave in the vector $\boldsymbol{\gamma}$, and hence the problem (9) can usually be solved efficiently, particularly when cost functions and rate functions are separable over different network components [17]. Below, we use $\boldsymbol{\gamma}_V^* = (\gamma_{V1}^*, \gamma_{V2}^*, ..., \gamma_{Vr}^*)^T$ to denote an optimal solution of the problem (9) with the corresponding V.

V. Performance of LIFO-Backpressure

In this section, we analyze the performance of Backpressure with the LIFO queueing discipline (called LIFO-Backpressure). Note that in this case, the queues still evolve according to (3), i.e., first serve the packets in the queue, and then admit the new arrivals. However, the served packets are chosen in a LIFO manner. The idea of using LIFO under Backpressure is first proposed in [10], although they did not provide any theoretical performance guarantee. We will show, under some mild conditions (to be stated in Theorem 3), that under LIFO-Backpressure, the time average delay for almost all packets entering the network is $O([\log(V)]^2)$ when the utility is pushed to within $O(1/V)$ of the optimal value. Note that the implementation complexity of LIFO-Backpressure is the same as the original Backpressure, and LIFO-Backpressure only requires knowledge of the instantaneous network condition. This is a remarkable feature that distinguishes it from the previous algorithms achieving similar poly-logarithmic tradeoffs in the i.i.d. case, e.g., [7] [8] [9], which all require knowledge of some implicit network parameters other than the instant network state. Below, we first provide a simple example to demonstrate the need for careful treatment of the usage of LIFO in Backpressure algorithms. Then, we present a modified Little's theorem that will be used for our proof.

A. A simple example on the LIFO delay

Consider a slotted system where two packets arrive at time 0, and one packet periodically arrives every slot thereafter (at times $1, 2, 3, \ldots$). The system is initially empty and can serve exactly one packet per slot. The arrival rate λ is clearly 1 packet/slot (so that $\lambda = 1$). Further, under either FIFO or LIFO service, there are always 2 packets in the system, so $\overline{Q} = 2$.

Under FIFO service, the first packet has a delay of 1 and all packets thereafter have a delay of 2:

$$W_1^{\text{FIFO}} = 1\,,\ W_i^{\text{FIFO}} = 2\ \forall i \in \{2, 3, 4, \ldots\},$$

where W_i^{FIFO} is the delay of the i^{th} packet under FIFO (W_i^{LIFO} is similarly defined for LIFO). We thus have:

$$\overline{W}^{\text{FIFO}} \triangleq \lim_{K\to\infty} \frac{1}{K}\sum_{i=1}^{K} W_i^{\text{FIFO}} = 2.$$

Thus, $\lambda\overline{W}^{\text{FIFO}} = 1 \times 2 = 2$, $\overline{Q} = 2$, and so $\lambda\overline{W}^{\text{FIFO}} = \overline{Q}$ indeed holds.

Now consider the same system under LIFO service. We still have $\lambda = 1$, $\overline{Q} = 2$. However, in this case the first packet never departs, while all other packets have a delay equal to 1 slot:

$$W_1^{\text{LIFO}} = \infty\,,\ W_i^{\text{LIFO}} = 1\ \forall i \in \{2, 3, 4, \ldots\}.$$

Thus, for all integers $K > 0$:

$$\frac{1}{K}\sum_{i=1}^{K} W_i^{\text{LIFO}} = \infty.$$

and so $\overline{W}^{\text{LIFO}} = \infty$. Clearly $\lambda\overline{W}^{\text{LIFO}} \neq \overline{Q}$. On the other hand, if we ignore the one packet with infinite delay, we note that all other packets get a delay of 1 (exactly half the delay in the FIFO system). Thus, in this example, LIFO service significantly improves delay for all but the first packet.

For the above LIFO example, it is interesting to note that if we define $\tilde{Q}$ and $\tilde{W}$ as the average backlog and delay *associated only with those packets that eventually depart*, then we have $\tilde{Q} = 1$, $\tilde{W} = 1$, and the equation $\lambda\tilde{W} = \tilde{Q}$ indeed holds. This motivates the theorem in the next subsection, which considers a time average only over those packets that eventually depart.

B. A Modified Little's Theorem for LIFO systems

We now present the modified Little's theorem. Let $\mathcal{B}$ represent a finite set of buffer locations in a LIFO queueing system. Let $N(t)$ be the number of arrivals that use a buffer location within set $\mathcal{B}$ up to time t. Let $D(t)$ be the number of departures from a buffer location within the set $\mathcal{B}$ up to time t. Let W_i be the delay of the ith job to depart from the set $\mathcal{B}$. Define $\overline{W}$ as the $\limsup$ average delay *considering only those jobs that depart*:

$$\overline{W} \triangleq \limsup_{t\to\infty} \frac{1}{D(t)}\sum_{i=1}^{D(t)} W_i.$$

We then have the following theorem:

***Theorem** 1:* Suppose there is a constant $\lambda_{\min} > 0$ such that with probability 1:

$$\liminf_{t\to\infty} \frac{N(t)}{t} \geq \lambda_{\min}.$$

Further suppose that $\lim_{t\to\infty} D(t) = \infty$ with probability 1 (so the number of departures is infinite). Then the average delay $\overline{W}$ satisfies:

$$\overline{W} \triangleq \limsup_{t\to\infty} \frac{1}{D(t)}\sum_{i=1}^{D(t)} W_i \leq |\mathcal{B}|/\lambda_{\min},$$

where $|\mathcal{B}|$ is the size of the finite set $\mathcal{B}$.

Proof: See Appendix A. ■

We note that under FIFO, the delay experienced by a packet in a set $\mathcal{B}$ may not be equal to its delay in the queue. This is because packets under FIFO will gradually move from buffer slots at the end of the queue towards the buffer slots at the front. Under LIFO, however, once a packet leaves a buffer slot, it also leaves the queue. Hence, the average delay packets experience in their buffer slots is also the average delay they experience in the queue. Thus, Theorem 1 can be used to study packet queueing delay for LIFO systems.

C. LIFO-Backpressure Proof

We now provide the analysis of LIFO-Backpressure. To prove our result, we will use the following theorem from [16], which is the first to show that Backpressure (with either FIFO or LIFO) achieves the exact $[O(1/V), O(V)]$ utility-delay tradeoff under a Markovian network state process. It generalizes the $[O(1/V), O(V)]$ performance result of Backpressure in the i.i.d. case in [6].

***Theorem** 2:* Suppose $S(t)$ is a finite state irreducible and aperiodic Markov chain[2] and the slackness condition (2) holds, Backpressure (with either FIFO or LIFO) achieves the following:

$$f_{\text{av}}^{\text{BP}} = f_{\text{av}}^{*} + O(1/V),\ \overline{q}^{\text{BP}} = O(V), \tag{11}$$

where $f_{\text{av}}^{\text{BP}}$ and $\overline{q}^{\text{BP}}$ are the expected time average cost and backlog under Backpressure.

Proof: See [16]. ■

Theorem 2 thus shows that LIFO-Backpressure guarantees an average backlog of $O(V)$ when pushing the utility to within $O(1/V)$ of the optimal value. We now consider the delay performance of LIFO-Backpressure. For our analysis, we need the following theorem (which is Theorem 1 in [9]).

***Theorem** 3:* Suppose $\boldsymbol{\gamma}_V^*$ is unique, the slackness condition (2) holds, and the dual function $g(\boldsymbol{\gamma})$ satisfies:

$$g(\boldsymbol{\gamma}_V^*) \geq g(\boldsymbol{\gamma}) + L||\boldsymbol{\gamma}_V^* - \boldsymbol{\gamma}||, \quad \forall\, \boldsymbol{\gamma} \succeq \mathbf{0}, \tag{12}$$

for some constant $L > 0$ independent of V. Then, under Backpressure with FIFO or LIFO, there exist constants $D, K, c^* = \Theta(1)$, i.e., all independent of V, such that for any $m \in \mathbb{R}_+$,

$$\mathcal{P}^{(r)}(D, Km) \quad \leq \quad c^* e^{-m}, \tag{13}$$

where $\mathcal{P}^{(r)}(D, Km)$ is defined:

$$\mathcal{P}^{(r)}(D, Km) \tag{14}$$
$$\triangleq \limsup_{t\to\infty} \frac{1}{t}\sum_{\tau=0}^{t-1} \Pr\{\exists\, j, |q_j(\tau) - \gamma_{Vj}^*| > D + Km\}.$$

Proof: See [9]. ■

Note that if a steady state distribution exists for $\boldsymbol{q}(t)$, e.g., when all queue sizes are integers, then $\mathcal{P}^{(r)}(D, Km)$ is indeed the steady state probability that there exists a queue j whose queue value deviates from γ_{Vj}^* by more than $D+Km$ distance. In this case, Theorem 3 states that $q_j(t)$ deviates from γ_{Vj}^* by $\Theta(\log(V))$ distance with probability $O(1/V)$. Hence when V is large, $q_j(t)$ will mostly be within $O(\log(V))$ distance from γ_{Vj}^*. Also note that the conditions of Theorem 3 are not

[2]In [16], the theorem is proven under more general Markovian $S(t)$ processes that include the $S(t)$ process assumed here.

very restrictive. The condition (12) can usually be satisfied in practice when the action space is finite, in which case the dual function $g(\boldsymbol{\gamma})$ is polyhedral (see [9] for more discussions). The uniqueness of $\boldsymbol{\gamma}_V^*$ can usually be satisfied in network utility optimization problems, e.g., [2].

We now present the main result of this paper with respect to the delay performance of LIFO-Backpressure. Below, the notion "average arrival rate" is defined as follows: Let $A_j(t)$ be the number of packets entering queue j at time t. Then, the time average arrival rate of these packets is defined as (assuming it exists): $\lambda_j = \lim_{t\to\infty}\frac{1}{t}\sum_{\tau=0}^{t-1}A_j(\tau)$. For the theorem, we assume that time averages under Backpressure exist with probability 1. This is a reasonable assumption, and holds whenever the resulting discrete time Markov chain for the queue vector $\boldsymbol{q}(t)$ under Backpressure is countably infinite and irreducible. Note that the state space is indeed countably infinite if we assume packets take integer units. If the system is also irreducible, then the finite average backlog result of Theorem 2 implies that all states are positive recurrent.

Let D, K, c^* be constants as defined in Theorem 3, and recall that these are $\Theta(1)$ (independent of V). Assume $V \geq 1$, and define $Q_{j,\text{High}}$ and $Q_{j,\text{Low}}$ as:

$$Q_{j,\text{High}} \triangleq \gamma_{Vj}^* + D + K[\log(V)]^2, \qquad (15)$$

$$Q_{j,\text{Low}} \triangleq \max[\gamma_{Vj}^* - D - K[\log(V)]^2, 0]. \qquad (16)$$

Define the interval $\mathcal{B}_j \triangleq [Q_{j,\text{Low}}, Q_{j,\text{High}}]$. The following theorem considers the rate and delay of packets that enter when $q_j(t) \in \mathcal{B}_j$ and that eventually depart.

Theorem 4: Suppose $V \geq 1$, $\boldsymbol{\gamma}_V^*$ is unique, the slackness assumption (2) holds, and the dual function $g(\boldsymbol{\gamma})$ satisfies:

$$g(\boldsymbol{\gamma}_V^*) \geq g(\boldsymbol{\gamma}) + L||\boldsymbol{\gamma}_V^* - \boldsymbol{\gamma}||, \quad \forall\, \boldsymbol{\gamma} \succeq \mathbf{0}, \qquad (17)$$

for some constant $L > 0$ independent of V. Define D, K, c^* as in Theorem 3, and define $\mathcal{B}_j$ as above. Then, for any queue j with a time average input rate $\lambda_j > 0$, we have under LIFO-Backpressure that:

(a) The rate $\tilde{\lambda}_j$ of packets that both arrive to queue j when $q_j(t) \in \mathcal{B}_j$ and that eventually depart the queue satisfies:

$$\lambda_j \geq \tilde{\lambda}_j \geq \left[\lambda_j - \frac{\delta_{\max}c^*}{V^{\log(V)}}\right]^+. \qquad (18)$$

(b) The average delay of these packets is at most W_{bound}, where:

$$W_{\text{bound}} \triangleq [2D + 2K[\log(V)]^2 + \delta_{\max}]/\tilde{\lambda}_j.$$

This theorem says that the delay of packets that enter when $q_j(t) \in \mathcal{B}_j$ and that eventually depart is at most $O([\log(V)]^2)$. Further, by (18), when V is large, these packets represent the overwhelming majority, in that the rate of packets not in this set is at most $O(1/V^{\log(V)})$.

Proof: (Theorem 4) Theorem 2 shows that the average network queue backlog is finite. Thus, there can be at most a finite number of packets that enter the queue and never depart. So the rate of packets arriving that never depart must be 0. It follows that $\tilde{\lambda}_j$ is equal to the rate at which packets arrive when $q_j(t) \in \mathcal{B}_j$. Define the indicator function $1_j(t)$ to be 1 if $q_j(t) \notin \mathcal{B}_j$, and 0 else. Define $\tilde{\lambda}_j^c \triangleq \lambda_j - \tilde{\lambda}_j$. Then, with probability 1, we get: [3]

$$\begin{aligned}\tilde{\lambda}_j^c &= \lim_{t\to\infty}\frac{1}{t}\sum_{\tau=0}^{t-1}A_j(\tau)1_j(\tau)\\ &= \lim_{t\to\infty}\frac{1}{t}\sum_{\tau=0}^{t-1}\mathbb{E}\big\{A_j(\tau)1_j(\tau)\big\}.\end{aligned}$$

Then, using the fact that $A_j(t) \leq \delta_{\max}$ for all j, t, we have:

$$\begin{aligned}\mathbb{E}\big\{A_j(t)1_j(t)\big\} &= \mathbb{E}\big\{A_j(t)|q_j(t)\notin\mathcal{B}_j\big\}\Pr\{q_j(t)\notin\mathcal{B}_j)\\ &\leq \delta_{\max}\Pr(q_j(t)\notin[Q_{j,\text{Low}}, Q_{j,\text{High}}]).\end{aligned}$$

Therefore:

$$\begin{aligned}\tilde{\lambda}_j^c &\leq \delta_{\max}\lim_{t\to\infty}\frac{1}{t}\sum_{\tau=0}^{t-1}\Pr(q_j(\tau)\notin[Q_{j,\text{Low}}, Q_{j,\text{High}}])\\ &\leq \delta_{\max}\lim_{t\to\infty}\frac{1}{t}\sum_{\tau=0}^{t-1}\Pr(|q_j(\tau)-\gamma_{V,j}^*| > D + Km),\end{aligned}$$

where we define $m \triangleq [\log(V)]^2$. Note that $V \geq 1$ and $m \geq 0$. From Theorem 3 we thus have:

$$0 \leq \tilde{\lambda}_j^c \leq \delta_{\max}c^*e^{-m} = \frac{\delta_{\max}c^*}{V^{\log(V)}}. \qquad (19)$$

This proves Part (a).

To prove Part (b), define $\hat{\mathcal{B}}_j \triangleq [Q_{j,\text{Low}}, Q_{j,\text{High}}+\delta_{\max}]$. Since $\mathcal{B}_j \subset \hat{\mathcal{B}}_j$, we see that the rate of the packets that enter and leave $\hat{\mathcal{B}}_j$ is at least $\tilde{\lambda}_j$. Part (b) then follows from Theorem 1 and that $|\hat{\mathcal{B}}_j| \leq 2D + 2K[\log(V)]^2 + \delta_{\max}$. ■

Note that if $\lambda_j = \Theta(1)$, we see from Theorem 4 that, under LIFO-Backpressure, the time average delay for almost all packets going through queue j is only $O([\log(V)]^2)$. Applying this argument to all network queues with $\Theta(1)$ input rates, we see that all but a tiny fraction of the traffic entering the network only experiences a delay of $O([\log(V)]^2)$. This contrasts with the delay performance result of the usual Backpressure with FIFO, which states that the time average delay will be $\Theta(V)$ for all packets [9].

We remark that the queue process $\{\boldsymbol{q}(t), t = 0, 1, ...\}$ remains the *same* under Backpressure with both FIFO and LIFO. Thus, the required buffer sizes are the same in both cases. However, under LIFO, we are able to reduce the delay of the majority of the packets, by "shifting" most of their delay to a tiny set of packets. Also note that, in this paper we have implicitly assumed that each queue has an *infinite* buffer size (by assuming (3)). Extending our results to networks where nodes only have finite buffer sizes is an interesting direction for future research. Finally, we note that in many cases, e.g., [7], [19], Backpressure automatically ensures that the queue sizes remain bounded for all time. In such cases, using finite buffer sizes is sufficient to guarantee the performance of Backpressure.

VI. The LIFO^p-Backpressure Algorithm

In this section, we generalize the LIFO-Backpressure technique to allow interleaving between both LIFO and FIFO. Specifically, at every time slot, each queue will randomly decide to serve packets from either the back of the queue or the front of the queue. The motivation of this interleaving

[3]The time average expectation is the same as the pure time average by the Lebesgue Dominated Convergence Theorem [18], because we assume the pure time average exists with probability 1, and that $0 \leq A_j(t) \leq \delta_{\max}\ \forall t$.

approach is the observation that, under LIFO-Backpressure, a few packets may stay in the queues for a very long time, or they may never leave the queue. Thus, we want to resolve this problem by also allowing the FIFO discipline, which ensures that all the packets eventually leave each queue. We parameterize the algorithm by a single parameter $p \in [0,1]$, which represents the probability that a queue serves the packets from the back of the queue at a given time. [4] We call this approach the LIFO^p-Backpressure algorithm.

A. The Algorithm

The idea of LIFO^p-Backpressure is shown in Fig. 4. We first pre-specify a probability p. [5] Then, at every time slot, after choosing all the network action decisions according to Backpressure, the queue serves packets from the end with probability p; and from the front otherwise. Note that this back/front decision is independent of the action taken by Backpressure, and is independent for each queue. As we vary p from 0 to 1, the algorithm transitions from the usual Backpressure algorithm to LIFO-Backpressure.

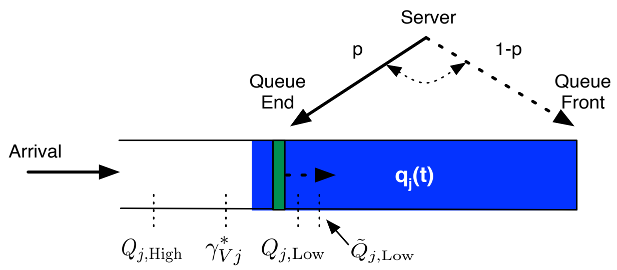

Fig. 4. The LIFO-FIFO interleaving technique. A packet is either served when the queue is serving the end of the queue, or it gradually moves to the right towards the front of the queue and is served when the queue is serving packets from the front of the queue (the green packet and the arrow show how packets gradually move to the right).

B. Performance Analysis

In this section, we analyze the performance of the LIFO^p-Backpressure algorithm. Without loss of generality, we assume that each packet has unit size. Our following theorem shows that LIFO^p-Backpressure ensures that roughly a fraction p of the packets experience only $O([\log(V)]^2)$ queueing delay. This result generalizes Theorem 4. In the theorem, we will use $Q_{j,\text{High}}$ and $Q_{j,\text{Low}}$ defined in (15) and (16), i.e.,

$$Q_{j,\text{High}} \triangleq \gamma^*_{Vj} + D + K[\log(V)]^2,$$
$$Q_{j,\text{Low}} \triangleq \max[\gamma^*_{Vj} - D - K[\log(V)]^2, 0],$$

as well as:

$$\tilde{Q}_{j,\text{Low}} \triangleq \max[\gamma^*_{Vj} - D - K[\log(V)]^2 - \delta_{\max}, 0].$$

Note that $\tilde{Q}_{j,\text{Low}}$ is slightly different from the $Q_{j,\text{Low}}$ (See Fig. 4) and is introduced for our analysis. Then, we similarly define $\tilde{\mathcal{B}}_j \triangleq [\tilde{Q}_{j,\text{Low}}, Q_{j,\text{High}}]$ and assume throughout that all the corresponding limits exist with probability 1.

Theorem 5: Suppose that $V \geq 1$, γ^*_V is unique, the slackness assumption (2) holds, and the dual function $g(\gamma)$ satisfies:

$$g(\gamma^*_V) \geq g(\gamma) + L||\gamma^*_V - \gamma||, \quad \forall\, \gamma \succeq \mathbf{0}, \tag{20}$$

for some constant $L > 0$ independent of V. Define D, K, c^* as in Theorem 3, and define $\tilde{\mathcal{B}}_j$ as above. Then, for any queue j with a time average input rate $\lambda_j > 0$, we have under LIFO^p-Backpressure that:

(a) All the packets eventually leave the queue if $0 \leq p < 1$.

(b) There exists a set of packets $\mathbb{P}_{j0}$ that arrive to queue j when $q_j(t) \in \tilde{\mathcal{B}}_j$, depart before they move to the right of $\tilde{Q}_{j,\text{Low}}$, and are served when the queue is serving the back of the queue. [6] The set $\mathbb{P}_{j0}$ has an average rate $\lambda_{\mathbb{P}_{j0}}$ that satisfies:

$$p\lambda_j \geq \lambda_{\mathbb{P}_{j0}} \geq \left[p\lambda_j - O\left(\frac{\delta_{\max} c^*}{V^{\log(V)}}\right)\right]^+. \tag{21}$$

(c) If $\lambda_{\mathbb{P}_{j0}} > 0$, the average delay of these packets is at most $\hat{W}_{\text{bound}}$, where:

$$\hat{W}_{\text{bound}} \triangleq 2(D + K[\log(V)]^2 + \delta_{\max})/\lambda_{\mathbb{P}_{j0}}.$$

Proof: See Appendix B. ∎

Theorem 5 says that by allocating a portion of the time to serve the packets from the front of the queue, the problem of packets being stuck in the queue can be resolved. *If the p parameter is chosen to be very close to one, then LIFO^p-Backpressure achieves almost the same performance guarantee as LIFO-Backpressure, while ensuring that all the packets are delivered.*

The formal proof of Theorem 5 is given in Appendix B. Here we sketch the proof idea: under the LIFO^p-Backpressure policy, for any queue j with an input rate $\lambda_j > 0$, the fraction of packets that are served when the queue is serving the back of the queue is $p\lambda_j$. Now we look at the packets that are served from the back. First, we see that they will almost all arrive to the queue when $q_j \in \tilde{\mathcal{B}}_j$ by Theorem 3. Second, they will also almost all leave the queue before they move to the right of $\tilde{Q}_{j,\text{Low}}$. The reason for this is that, if a packet moves to the right of $\tilde{Q}_{j,\text{Low}}$, i.e., it moves into a buffer spot in $[0, \tilde{Q}_{j,\text{Low}})$, then since $q_j(t)$ rarely gets below $Q_{j,\text{Low}} \approx \tilde{Q}_{j,\text{Low}} + \delta_{\max}$, it is very unlikely that this packet will be served from the back of the queue. It can then only gradually move to the front of the queue and be served there. Therefore, almost all the packets that are served from the back will enter and leave the queue when they are in the interval $\tilde{\mathcal{B}}_j$, which is of size $O([\log(V)]^2)$. Also, they have an average rate of roughly $p\lambda_j$. Using Theorem 1, we see that they experience an average delay of no more than $O([\log(V)]^2)/p\lambda_j$.

Note that similar to the LIFO-Backpressure case, LIFO^p-Backpressure reduces the delay of p-fraction of the packets by "shifting" their delay to the other $(1-p)$-fraction of the packets. Therefore, if we reduce the delay of more packets, i.e., p closer to 1, the other fraction of the packets will experience a larger average delay. This intuition will also be illustrated in the simulation results in Section VII.

[4]All the results in this section hold even if we choose different probabilities for different queues.

[5]This p value will be pre-determined and not changed afterwards. It is possible to use a p value that is a function of time in implementation. However, the analysis will be very challenging.

[6]Note that the third condition is needed because, when V is small, serving the queue from the front may also serve a packet to the left of $\tilde{Q}_{j,\text{Low}}$.

VII. Simulation

In this section, we provide simulation results of the LIFO-Backpressure algorithm and the LIFO^p-Backpressure algorithm. We consider the network shown in Fig. 5, where we try to support a flow sourced by Node 1 destined for Node 7 with minimum energy consumption.

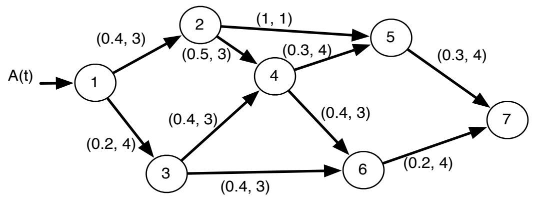

Fig. 5. A multihop network. (a, b) represents the HIGH probability a and the rate b obtained with one unit of power when HIGH.

We assume that $A(t)$ evolves according to the 2-state Markov chain in Fig. 6. When the state is HIGH, $A(t) = 3$, else $A(t) = 0$. We assume that the channel condition of each link can either be HIGH or LOW at a time. All the links except link $(2, 4)$ and link $(6, 7)$ are assumed to be i.i.d. every time slot, whereas the channel conditions of link $(2, 4)$ and link $(6, 7)$ are assumed to be evolving according to independent 2-state Markov chains in Fig. 6. Each link's HIGH probability and unit power rate at the HIGH state is shown in Fig. 5. The unit power rates of the links at the LOW state are all assumed to be 1. We assume that the link states are all independent and there is no interference. However, each node can only spend one unit of power per slot to transmit over one outgoing link, although it can simultaneously receive from multiple incoming links. The goal is to minimize the time average power while maintaining network stability.

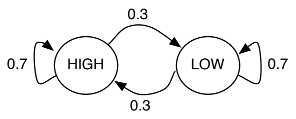

Fig. 6. The two state Markov chain with the transition probabilities.

We simulate Backpressure with both LIFO and FIFO for 10^6 slots with $V \in \{20, 50, 100, 200, 500\}$. It can be verified that the backlog vector converges to a unique attractor as V increases in this case. The left two plots in Fig. 7 show the average power consumption and the average backlog under LIFO-Backpressure. We observe that the average power quickly converges to the optimal value. We also see in the middle plot that the average packet delay under LIFO-Backpressure scales only as $O([\log(V)]^2)$, whereas the delay under FIFO increases linearly in V. The right plot of Fig. 7 shows the percentage of time when there exists a q_j whose value deviates from γ^*_{Vj} by more than $2[\log(V)]^2$. As we can see, this percentage is always very small, i.e., between 0.002 and 0.013, showing a good match between the theory and the simulation results.

Fig. 8 compares the delay statistics of LIFO and FIFO for more than 99.9% of the packets that leave the system before the simulation ends, under the cases where $V = 100$ and $V = 500$. We see that LIFO not only dramatically reduces the average packet delay for these packets, but also greatly

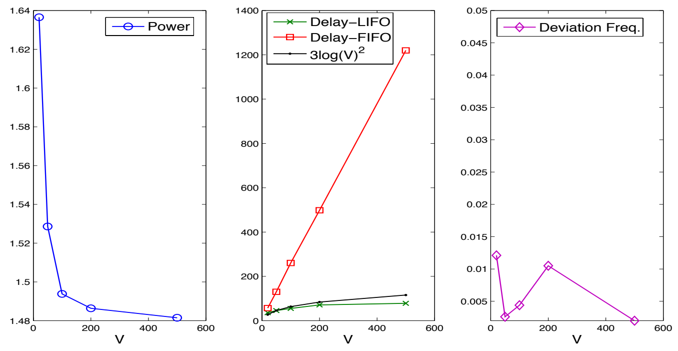

Fig. 7. LEFT: average network power consumption. MIDDLE: average packet delay under FIFO and LIFO considering only the packets that depart from the network (more than 99.9%). RIGHT: percentage of time when $\exists\ q_j$ such that $|q_j - \gamma^*_{Vj}| > 2[\log(V)]^2$.

reduces the delay for most packets. For instance, when $V = 500$, under FIFO, almost all packets experience the average delay around 1220 slots. Whereas under LIFO, the average packet delay is brought down to 78. Moreover, 52.9% of the packets only experience delay less than 20 slots, and 90.4% of the packets experience delay less than 100 slots. *Hence most packets' delay are reduced by a factor of* 12 *under LIFO as compared to that under FIFO!*

V=100				
Case	Avg. DL	$\% DL < 20$	$\% DL < 50$	$\% DL < 100$
LIFO	55.4	55.0	82.1	91.8
FIFO	260.6	0	0	0

V=500				
Case	Avg. DL	$\% DL < 20$	$\% DL < 50$	$\% DL < 100$
LIFO	78.3	52.9	80.4	90.4
FIFO	1219.8	0	0	0

Fig. 8. Delay statistics under Backpressure with LIFO and FIFO for packets that leave the system before simulation ends (more than 99.9%). $\%DL < a$ is the percentage of packets that enter the network and has delay less than a.

Fig. 9 also shows the delay for the first 20000 packets that enter the network in the case when $V = 500$. We see that under Backpressure plus LIFO, most packets experience very small delay; while under Backpressure with FIFO, each packet experiences roughly the average delay.

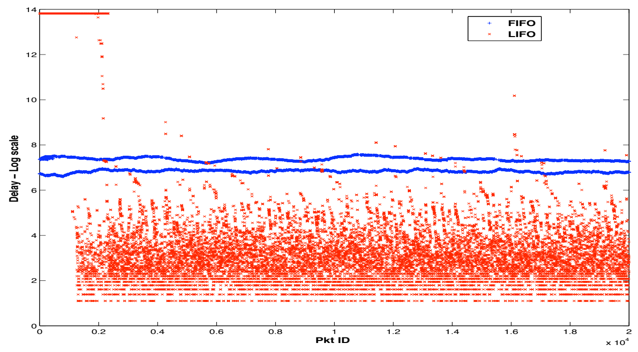

Fig. 9. Packet delay under Backpressure with LIFO and FIFO

Finally, we simulate the LIFO^p-Backpressure algorithm and plot the results in Fig. 10. In this case, it can be verified that the average delay of *all* packets is the same as that under the original Backpressure. Since in the example network packets need to go through multiple nodes to reach their destination, the fraction of packets that experience only LIFO phases is not exactly equal to p. However, the fraction will still be proportional in p. In this case, as we increase the value of p, more packets will be served in the LIFO phase. However, the average delay of the packets that experience FIFO phases will also increase.

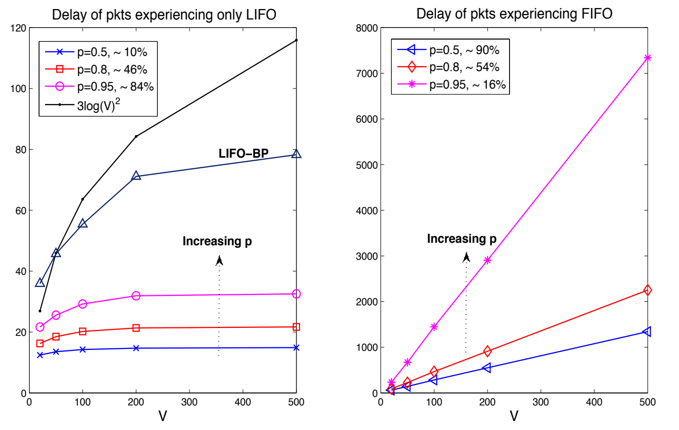

Fig. 10. Average packet delay under LIFO^p-Backpressure. The number next to each p value indicates the percentage of packets that only experience LIFO phases (or experience at least one FIFO phase in the right plot). Under a larger p, more packets will experience only LIFO phases, which on the other hand increases the average delay of the packets experience FIFO phases.

VIII. Empirical Validation

In this section we validate our analysis of LIFO-Backpressure empirically by carrying out new experiments over the same testbed and Backpressure Collection Protocol (BCP) code of [10]. This prior work did not empirically evaluate the relationship between V, finite storage availability, packet latency and packet discard rate. We note that BCP runs atop the default CSMA MAC for TinyOS which is not known to be throughput optimal, that the testbed may not precisely be defined by a finite state Markovian evolution, and finally that limited storage availability on real wireless sensor nodes mandates the introduction of virtual queues to maintain backpressure values in the presence of data queue overflows.

In order to avoid using very large data buffers, in [10] the forwarding queue of BCP has been implemented as a *floating queue*. The concept of a floating queue is shown in Fig. 11, which operates with a finite data queue of size $D_{\max}$ residing atop a virtual queue which preserves backpressure levels. Packets that arrive to a full data queue result in a data queue discard and the incrementing of the underlying virtual queue counter. Underflow events (in which a virtual backlog exists but the data queue is empty) results in null packet generation, which are filtered and then discarded by the destination.

Despite these real-world differences, we are able to demonstrate clear order-equivalent delay gains due to LIFO usage in BCP in the following experimentation.

A. Testbed and General Setup

To demonstrate the empirical results, we deployed a collection scenario across 40 nodes within the Tutornet testbed (see Fig. 12). This deployment consisted of Tmote Sky devices embedded in the 4th floor of Ronald Tutor Hall at the University of Southern California.

In these experiments, one sink mote (ID 1 in Fig. 12) was designated and the remaining 39 motes sourced traffic

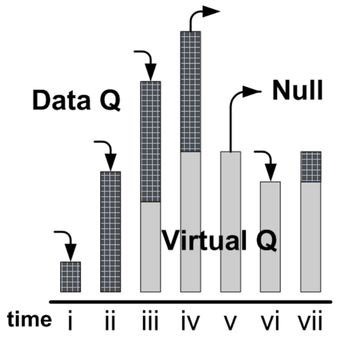

Fig. 11. The floating LIFO queues of [10] drop from the data queue during overflow, placing the discards within an underlying virtual queue. Services that cause data queue underflows generate null packets, reducing the virtual queue size.

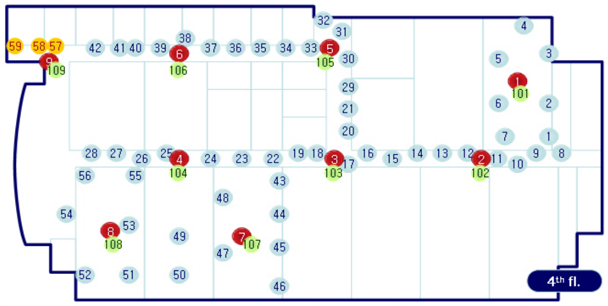

Fig. 12. The 40 tMote Sky devices used in experimentation on Tutornet.

simultaneously, to be collected at the sink. The Tmote Sky devices were programmed to operate on 802.15.4 channel 26, selected for the low external interference in this spectrum on Tutornet. Further, the motes were programmed to transmit at -15 dBm to provide reasonable interconnectivity. These experimental settings are identical to those used in [10].

We vary $D_{\max}$, the buffer size of the motes, over experimentation. In practice, BCP defaults to a $D_{\max}$ setting of 12 packets, the maximum reasonable resource allocation for a packet forwarding queue in these highly constrained devices.

B. Experiment Parameters

Experiments consisted of Poisson traffic at 1.0 packets per second per source for a duration of 20 minutes. This source load is moderately high, as the boundary of the capacity region for BCP running on this subset of motes has previously been documented at 1.6 packets per second per source [10]. A total of 36 experiments were run using the standard BCP LIFO queue mechanism, for all combinations of $V \in \{1, 2, 3, 4, 6, 8, 10, 12\}$ and LIFO storage threshold $D_{\max} \in \{2, 4, 8, 12\}$. In order to present a delay baseline for Backpressure we additionally modified the BCP source code and ran experiments with 32-packet FIFO queues (no floating queues) for $V \in \{1, 2, 3\}$. [7]

C. Results

Testbed results in Fig. 13 provide the system average packet delay from source to sink over V and $D_{\max}$, and includes 95% confidence intervals. Delay in our FIFO implementation scales linearly with V, as predicted by the analysis in [9]. This

[7]These relatively small V values are due to the constraint that the motes have small data buffers. Using larger V values will cause buffer overflow at the motes.

yields an average delay that grows very rapidly with V, already greater than 9 seconds per packet for $V = 3$. Meanwhile, the LIFO floating queue of BCP performs much differently. We have plotted a scaled $[\log(V)]^2$ target, and note that as $D_{\max}$ increases the average packet delay remains bounded by $\Theta([\log(V)]^2)$. [8]

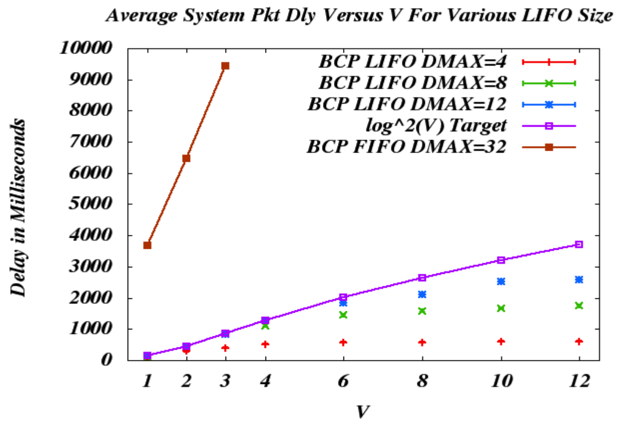

Fig. 13. System average source to sink packet delay for BCP FIFO versus BCP LIFO implementation over various V parameter settings.

These delay gains are only possible as a result of discards made by the LIFO floating queue mechanism that occur when the queue size fluctuates beyond the capability of the finite data queue to smooth. Fig. 14 gives the system packet loss rate of BCP's LIFO floating queue mechanism over V. Note that the poly-logarithmic delay performance of Fig. 13 is achieved even for data queue size 12, which itself drops at most 5% of traffic at $V = 12$. We cannot state conclusively from these results that the drop rate scales like $O(\frac{1}{V^{c_0 \log(V)}})$. We hypothesize that a larger V value would be required in order to observe the predicted drop rate scaling. Bringing these results back to real-world implications, note that BCP (which minimizes a penalty function of packet retransmissions) performs very poorly with $V = 0$, and was found to have minimal penalty improvement for V greater than 2. At this low V value, BCP's 12-packet forwarding queue demonstrates zero packet drops in the results presented here. These experiments, combined with those of [10] suggest strongly that the drop rate scaling may be inconsequential in many real world applications.

In order to explore the queue backlog characteristics and compare with our analysis, Fig. 15 presents a histogram of queue backlog frequency for rear-network-node 38 over various V settings. This node was observed to have the worst queue size fluctuations among all thirty-nine sources. For $V = 2$, the queue backlog is very sharply distributed and fluctuates outside the range $[11 - 15]$ only 5.92% of the experiment. As V is increased, the queue attraction is evident. For $V = 8$ we find that the queue deviates outside the range $[41 - 54]$ only 5.41% of the experiment. The queue deviation is clearly scaling sub-linearly, as a four-fold increase in V

[8]We note in Fig. 13 that the FIFO case is implemented with $D_{\max} = 32$. Without a clear floating mechanism to allow virtual queue growth while maintaining the FIFO discipline, a $D_{\max}$ less than 32 is unstable even for $V = 3$. The backlogs that must be achieved can be seen in rear network node 38 of Fig. 15. The reason to present the FIFO results here is to show the linear delay growth characteristic of the usual Backpressure technique and compare it with the logarithmic growth of LIFO-Backpressure.

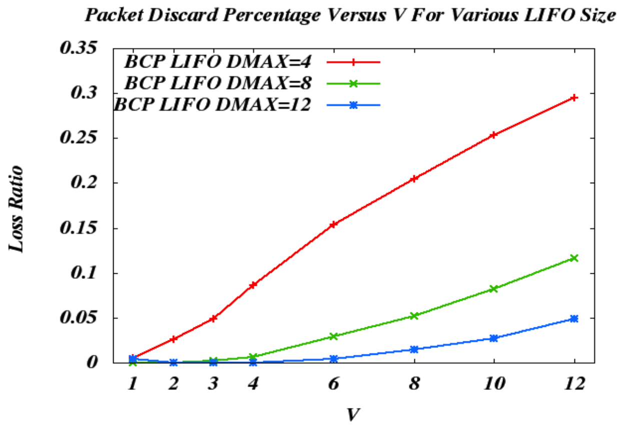

Fig. 14. System packet loss rate of BCP LIFO implementation over various V parameter settings.

required only a 2.8 fold increase in $D_{\max}$ for comparable drop performance.

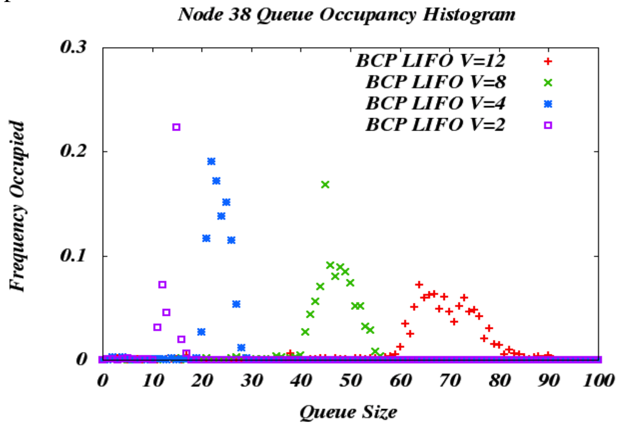

Fig. 15. Histogram of queue backlog frequency for rear-network-node 38 over various V settings.

IX. Optimizing Functions of Time averages

So far we have focused on optimizing time averages of functions, we now consider the case when the objective of the network controller is to optimize a function of some time average metric, e.g., [20]. Specifically, we assume that the action $x(t)$ at time t incurs some instantaneous *network attribute vector* $\boldsymbol{y}(t) = \boldsymbol{y}(x(t)) = (y_1(t), ..., y^K(t))^T \in \mathbb{R}_+^K$, and the objective of the network controller is to minimize a cost function $\text{Cost}(\overline{\boldsymbol{y}(t)}) : \mathbb{R}_+^K \rightarrow \mathbb{R}_+$, [9] where $\overline{\boldsymbol{y}(t)}$ represents the time average value of $\boldsymbol{y}(t)$. We assume that the function $\text{Cost}(\cdot)$ is continuous, convex and is component-wise increasing, and that $|y_k(x(t))| \leq \delta_{\max}$ for all k, $x(t)$. In this case, we see that the Backpressure algorithm in Section IV cannot be directly applied and the deterministic problem (8) also needs to be modified.

To tackle this problem using the Backpressure algorithm, we introduce an *auxiliary vector* $\boldsymbol{z}(t) = (z_1(t), ..., z_K(t))^T$. We then define the following virtual queues $H_k(t), j = 1, ..., K$ that evolves as follows:

$$H_k(t+1) = \max\left[H_k(t) - y_k(t), 0\right] + z_k(t). \quad (22)$$

These virtual queues are introduced for ensuring that the average value of $y_k(t)$ is no less than the average value of

[9]The case for maximizing a utility function of long term averages can be treated in a similar way.

$z_k(t)$. We will then try to maximize the time average of the function $\text{Cost}(\boldsymbol{z}(t))$, subject to the constraint that the actual queues $q_j(t)$ and the virtual queues $H_k(t)$ must all be stable. Specifically, the Backpressure algorithm for this problem works as follows:

Backpressure-TimeAverage: At every time slot t, observe the current network state $S(t)$, and the backlogs $\boldsymbol{q}(t)$ and $\boldsymbol{H}(t)$. If $S(t) = s_i$, do the following:

1) Auxiliary vector: choose the vector $\boldsymbol{z}(t)$ by solving:

$$\begin{aligned} \min: \quad & V\text{Cost}(\boldsymbol{z}) + \sum_k H_k(t) z_k \\ \text{s.t.} \quad & 0 \leq z_k \leq \delta_{\max}, \forall k. \end{aligned} \tag{23}$$

2) Action: choose the action $x(t) \in \mathcal{X}^{(s_i)}$ that solves:

$$\begin{aligned} \max: \quad & \sum_k H_k(t) y_k(x) + \sum_j q_j(t)[\mu_j(s_i, x) - A_j(s_i, x)] \\ \text{s.t.} \quad & x \in \mathcal{X}^{(s_i)}. \qquad \diamond \end{aligned} \tag{24}$$

In this case, one can also show that this Backpressure algorithm achieves the $[O(1/V), O(V)]$ utility-delay tradeoff under a Markovian $S(t)$ process. We also note that in this case, the deterministic problem is slightly different. Indeed, the intuitive formulation will be of the following form:

$$\begin{aligned} \min: \quad & \mathcal{F}(\boldsymbol{x}) \triangleq V\text{Cost}(\sum_{s_i} \pi_{s_i} \boldsymbol{y}(x^{(s_i)})) \qquad (25) \\ \text{s.t.} \quad & \mathcal{A}_j(\boldsymbol{x}) \triangleq \sum_{s_i} \pi_{s_i} A_j(s_i, x^{(s_i)}) \\ & \qquad \leq \mathcal{B}_j(\boldsymbol{x}) \triangleq \sum_{s_i} \pi_{s_i} \mu_j(s_i, x^{(s_i)}) \quad \forall j \\ & x^{(s_i)} \in \mathcal{X}^{(s_i)} \quad \forall i = 1, 2, ..., M. \end{aligned}$$

However, the dual problem of this optimization problem is not separable, i.e., not of the form of (10), unless the function $\text{Cost}(\cdot)$ is linear or if there exists an optimal action that is in every feasible action set $\mathcal{X}^{(s_i)}$, e.g., [20]. To resolve this problem, we introduce the auxiliary vector $\boldsymbol{z} = (z_1, ..., z_K)^T$ and change the problem to:

$$\begin{aligned} \min: \quad & \mathcal{F}(\boldsymbol{x}) \triangleq V\text{Cost}(\boldsymbol{z}) \qquad (26) \\ \text{s.t.} \quad & \boldsymbol{z} \preceq \sum_{s_i} \pi_{s_i} \boldsymbol{y}(x^{(s_i)}) \\ & \mathcal{A}_j(\boldsymbol{x}) \triangleq \sum_{s_i} \pi_{s_i} A_j(s_i, x^{(s_i)}) \\ & \qquad \leq \mathcal{B}_j(\boldsymbol{x}) \triangleq \sum_{s_i} \pi_{s_i} \mu_j(s_i, x^{(s_i)}) \quad \forall j \\ & x^{(s_i)} \in \mathcal{X}^{(s_i)} \quad \forall i = 1, 2, ..., M. \end{aligned}$$

It can be shown that this modified problem is equivalent to (25). Now we see that it is indeed due to the non-separable feature of (25) that we need to introduce the auxiliary vector $\boldsymbol{z}(t)$ in the deterministic problem. We also note that the problem (26) actually has the form of (8). Therefore, all previous results on (8), e.g., Theorem 3 and 4 will also apply to problem (26).

X. Conclusion

In this paper, we show that the Backpressure algorithm, when combined with the LIFO queueing discipline (called LIFO-Backpressure), is able to achieve the near-optimal $[O(1/V), O([\log(V)]^2)]$ utility-delay tradeoff. The results are validated by simulations and sensor network testbed implementations. We also develop the LIFO^p-Backpressure algorithm, which generalizes LIFO-Backpressure by allowing an interleaving between LIFO and FIFO. We show that LIFO^p-Backpressure achieves the same near-optimal tradeoff, and guarantees that all packets are delivered.

Acknowledgement

We would like to thank Professor Adam Wierman from Caltech for helpful discussions on the LIFO^p-Backpressure algorithms.

Appendix A – Proof of Theorem 1

Here we provide the proof of Theorem 1.

Proof: Consider a sample path for which the $\liminf$ arrival rate is at least $\lambda_{\min}$ and for which we have an infinite number of departures (this happens with probability 1 by assumption). There must be a non-empty subset of $\mathcal{B}$ consisting of buffer locations that experience an infinite number of departures. Call this subset $\bar{\mathcal{B}}$. Now let $W_i^{(b)}$ be the delay of the i^{th} departure from b, let $D^{(b)}(t)$ denote the number of departures from a buffer slot $b \in \bar{\mathcal{B}}$ up to time t, and use $Q^{(b)}(t)$ to denote the occupancy of the buffer slot b at time t. Note that $Q^{(b)}(t)$ is either 0 or 1. For all $t \geq 0$, it can be shown that:

$$\sum_{i=1}^{D^{(b)}(t)} W_i^{(b)} \leq \int_0^t Q^{(b)}(\tau) d\tau. \tag{27}$$

This can be seen from Fig. 16 below. Therefore, we have:

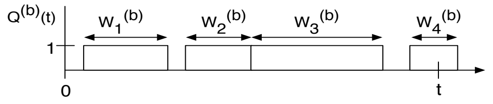

Fig. 16. An illustration of inequality (27) for a particular buffer location b. At time t in the figure, we have $D^{(b)}(t) = 3$.

$$\begin{aligned} \sum_{b \in \bar{\mathcal{B}}} \sum_{i=1}^{D^{(b)}(t)} W_i^{(b)} &\leq \int_0^t \sum_{b \in \bar{\mathcal{B}}} Q^{(b)}(\tau) d\tau \\ &\leq \int_0^t |\bar{\mathcal{B}}| d\tau \\ &\leq |\mathcal{B}| t. \end{aligned} \tag{28}$$

The left-hand-side of the above inequality is equal to the sum of all delays of jobs that depart from locations in $\bar{\mathcal{B}}$ up to time t. All other buffer locations (in $\mathcal{B}$ but not in $\bar{\mathcal{B}}$) experience only a finite number of departures. Let $\mathcal{J}$ represent an index set that indexes all of the (finite number) of jobs that depart from these other locations. Note that the delay W_j for each job $j \in \mathcal{J}$ is finite (because, by definition, job j eventually departs). We thus have:

$$\sum_{i=1}^{D(t)} W_i \leq \sum_{b \in \bar{\mathcal{B}}} \sum_{i=1}^{D^{(b)}(t)} W_i^{(b)} + \sum_{j \in \mathcal{J}} W_j.$$

where the inequality is because the second term on the right-hand-side sums over jobs in $\mathcal{J}$, and these jobs may not have departed by time t. Combining the above and (28) yields for all $t \geq 0$:

$$\sum_{i=1}^{D(t)} W_i \leq |\mathcal{B}| t + \sum_{j \in \mathcal{J}} W_j.$$

Dividing by $D(t)$ yields:

$$\frac{1}{D(t)}\sum_{i=1}^{D(t)} W_i \leq \frac{|\mathcal{B}|t}{D(t)} + \frac{1}{D(t)}\sum_{j\in\mathcal{J}} W_j.$$

Taking a $\limsup$ as $t\rightarrow\infty$ yields:

$$\limsup_{t\rightarrow\infty}\frac{1}{D(t)}\sum_{i=1}^{D(t)} W_i \leq \limsup_{t\rightarrow\infty}\frac{|\mathcal{B}|t}{D(t)}, \qquad (29)$$

where we have used the fact that $\sum_{j\in\mathcal{J}} W_j$ is a finite number, and $D(t)\rightarrow\infty$ as $t\rightarrow\infty$, so that:

$$\limsup_{t\rightarrow\infty}\frac{1}{D(t)}\sum_{j\in\mathcal{J}} W_j = 0.$$

Now note that, because each buffer location in $\mathcal{B}$ can hold at most one job, the number of departures $D(t)$ is at least $N(t)-|\mathcal{B}|$, which is a positive number for sufficiently large t. Thus:

$$\begin{aligned}\limsup_{t\rightarrow\infty}\frac{|\mathcal{B}|t}{D(t)} &\leq \limsup_{t\rightarrow\infty}\left[\frac{|\mathcal{B}|t}{N(t)-|\mathcal{B}|}\right]\\ &= \limsup_{t\rightarrow\infty}\left[\frac{|\mathcal{B}|}{N(t)/t-|\mathcal{B}|/t}\right]\\ &\leq |\mathcal{B}|/\lambda_{\min}.\end{aligned}$$

Using this in (29) proves the result. ■

Appendix B – Proof of Theorem 5

Here we prove Theorem 5. In the proof, we assume that all the corresponding limits exist.

Proof: First we see that under the conditions of Theorem 5, the network is stable. Thus, the average rate out of any queue j is equal to λ_j, which is the total input rate. That is, letting $\mu_j(t)$ be the number of packets that depart from queue j at time t, we have:

$$\mu_j \triangleq \lim_{t\rightarrow\infty}\frac{1}{t}\sum_{\tau=0}^{t-1}\mu_j(\tau) = \lambda_j. \qquad (30)$$

Now we use $e_j(t)$ to denote the event that queue j decides to serve packets from the end of the queue, and let μ_j^E be the time average departure rate of the packets that are served *during the times when queue j serves the end of the queue*. We see that: [10]

$$\begin{aligned}\mu_j^E &\triangleq \lim_{t\rightarrow\infty}\frac{1}{t}\sum_{\tau=0}^{t-1}\mu_j(\tau)1_{[e_j(\tau)]}\\ &\overset{(*)}{=} \lim_{t\rightarrow\infty}\frac{1}{t}\sum_{\tau=0}^{t-1}\mathbb{E}\big\{\mu(\tau)1_{[e_j(\tau)]}\big\} \qquad (31)\\ &= \lim_{t\rightarrow\infty}\frac{1}{t}\sum_{\tau=0}^{t-1}\mathbb{E}\big\{\mu_j(\tau)\big\}p = p\lambda_j.\end{aligned}$$

Here $1_{[\cdot]}$ is the indicator function, and $(*)$ follows from the fact that the limit exists, that $0\leq\mu(\tau)1_{[e(\tau)]}\leq\delta_{\max}$, and the Lebesgue Dominated Convergence Theorem [18]. As a result, the average rate of the packets that are served when the queue serves the front of the queue, denoted by μ_j^F, is $(1-p)\lambda_j$. Thus, if $0\leq p<1$, we see that $\mu_j^F>0$. This implies that every packet will eventually leave the queue. To see this, suppose this is not the case. Then, there exists at least one packet P^* that never leaves the queue. In this case, for any finite number $K>0$, there must be more than K packets in the queue when P^* arrives, since the packets are drained out from the front with an average rate $\mu_j^F>0$. However, this implies that the queue size must be infinite when P^* arrives, which contradicts the fact that the queue is stable. This proves Part (a).

Now define $\mathbb{P}_{j0}$ to be the set of packets that *(i) arrive to the queue when $q_j(t)\in\tilde{\mathcal{B}}_j$, (ii) are served before they move to the right of $\tilde{Q}_{j,Low}$, and (iii) are served when the queue is serving the end of the queue.* Let $\lambda_{\mathbb{P}_{j0}}$ be the average rate of the packets in $\mathbb{P}_{j0}$. It can be observed that $\lambda_{\mathbb{P}_{j0}}\leq p\lambda_j$. We thus want to show that:

$$\lambda_{\mathbb{P}_{j0}} \geq \big[p\lambda_j - O(\frac{\delta_{\max}c^*}{V^{\log(V)}})\big]^+. \qquad (32)$$

Then, using the fact that the packets from $\mathbb{P}_{j0}$ occupy an interval of size at most $Q_{j,\text{High}}+A_{\max}-\tilde{Q}_{j,\text{Low}}\leq 2(D+K[\log(V)]^2+\delta_{\max})=\Theta([\log(V)]^2)$, we can use Theorem 1 and conclude from (32) that the average delay for the packets in $\mathbb{P}_{j0}$ is $O([\log(V)]^2)/\lambda_{\mathbb{P}_{j0}}$ if $\lambda_{\mathbb{P}_{j0}}>0$. [11]

To prove (32), we first note that the packets that are served *when the queue is serving the end of the queue* consist of the following packet sets:

1) $\mathbb{P}_{j0}$
2) $\mathbb{P}_{j1}$, the set of packets that arrive when $q_j(t)\in\tilde{\mathcal{B}}_j$, and move to the right of $\tilde{Q}_{j,\text{Low}}$ but are still served from the end of the queue
3) $\mathbb{P}_{j2}$, the set of packets that are served from the end of the queue but arrive to the queue when $q_j(t)>Q_{j,\text{High}}$
4) $\mathbb{P}_{j3}$, the set of packets that are served from the end of the queue but arrive to the queue when $q_j(t)<\tilde{Q}_{j,\text{Low}}$

Now define $\lambda_{\mathbb{P}_{j1}}$, $\lambda_{\mathbb{P}_{j2}}$ and $\lambda_{\mathbb{P}_{j3}}$ to be the average rate of the packets in $\mathbb{P}_{j1}$, $\mathbb{P}_{j2}$ and $\mathbb{P}_{j3}$, respectively. We have:

$$\mu_j^E = p\lambda_j = \lambda_{\mathbb{P}_{j0}}+\lambda_{\mathbb{P}_{j1}}+\lambda_{\mathbb{P}_{j2}}+\lambda_{\mathbb{P}_{j3}}.$$

This implies that:

$$\lambda_{\mathbb{P}_{j0}} = p\lambda_j - (\lambda_{\mathbb{P}_{j1}}+\lambda_{\mathbb{P}_{j2}}+\lambda_{\mathbb{P}_{j3}}). \qquad (33)$$

Therefore, to prove (32), it suffices to show that $\lambda_{\mathbb{P}_{ji}}=O(\frac{\delta_{\max}c^*}{V^{\log(V)}})$ for $i=1,2,3$. To do this, we first note that $\lambda_{\mathbb{P}_{j2}}$ and $\lambda_{\mathbb{P}_{j3}}$ are upper bounded by the total arrival rates of the packets that enter the queue when $q_j(t)>Q_{j,\text{High}}$ and $q_j(t)<\tilde{Q}_{j,\text{Low}}$, respectively. Using the definition of $Q_{j,\text{High}}$ and $\tilde{Q}_{j,\text{Low}}$ and Theorem 3, we have $\lambda_{\mathbb{P}_{j2}}=O(\frac{\delta_{\max}c^*}{V^{\log(V)}})$, $\lambda_{\mathbb{P}_{j3}}=O(\frac{\delta_{\max}c^*}{V^{\log(V)}})$.

We now compute an upper bound on $\lambda_{\mathbb{P}_{j1}}$. Note that when the queue decides to serve packets from the end of the queue at time t, in order for it to serve a packet in $\mathbb{P}_{j1}$, i.e., a packet that arrives to the queue at some time $t'<t$ when $q_j(t')\in\tilde{\mathcal{B}}_j$ but moves to the right of $\tilde{Q}_{j,\text{Low}}$, we must have $q_j(t)<Q_{j,\text{Low}}$. Therefore, if we denote $\mu_{j1}(t)$ the number of packets in $\mathbb{P}_{j1}$ that are served at time t, we see that $\mu_{j1}(t)\leq\mu_{\max}1_{[q_j(t)<Q_{j,\text{Low}}]}1_{[e_j(t)]}$ for all t. Therefore,

$$\lambda_{\mathbb{P}_{j1}} = \lim_{t\rightarrow\infty}\frac{1}{t}\sum_{\tau=0}^{t-1}\mathbb{E}\big\{\mu_{j1}(\tau)\big\}$$

[10]Note that the existence of μ_j^E can be rigorously established using (30) and the fact that $\mu_j(\tau)\leq\delta_{\max}$. However, the details are quite involved and hence are omitted for brevity.

[11]While Theorem 1 was stated for LIFO systems where each packet stays in the same buffer location until departure, the result generalizes to the case when buffer locations can shift within a set $\mathcal{B}$ as long as the group of packets under consideration never leave the set $\mathcal{B}$.

$$\begin{aligned} &\leq \lim_{t\to\infty}\frac{1}{t}\sum_{\tau=0}^{t-1}\mathbb{E}\big\{\delta_{\max}1_{[q_j(\tau)<Q_{j,\text{Low}}]}1_{[e_j(\tau)]}\big\} \\ &\overset{(*)}{=} p\delta_{\max}\lim_{t\to\infty}\frac{1}{t}\sum_{\tau=0}^{t-1}\mathsf{Pr}(q_j(\tau)<Q_{j,\text{Low}}) \\ &\overset{(**)}{\leq} p\delta_{\max}\mathcal{P}^{(r)}(D,K[\log(V)]^2) \\ &= \frac{\delta_{\max}c^*}{V^{\log(V)}}. \end{aligned}$$

Here $(*)$ uses the fact that $e_j(t)$ is independent of the queue process, and $(**)$ follows from the definition of $\mathcal{P}^{(r)}(D,K[\log(V)]^2)$. Now using the fact that $\lambda_{\mathbb{P}_{ji}} = O(\frac{\delta_{\max}c^*}{V^{\log(V)}})$ for $i=1,2,3$ in (33), we conclude that:

$$\lambda_{\mathbb{P}_{j0}} \geq \big[p\lambda_j - O(\frac{\delta_{\max}c^*}{V^{\log(V)}})\big]^+. \tag{34}$$

This proves Part (b).

Now by the definition of $\mathbb{P}_{j0}$, we see that the packets in $\mathbb{P}_{j0}$ only occupy an interval of size no more than $Q_{j,\text{High}} - Q_{j,\text{Low}} + \delta_{\max} \leq 2(D+K[\log(V)]^2+\delta_{\max}) = \Theta([\log(V)]^2)$. Thus, using Theorem 1, the average delay for the packets in $\mathbb{P}_{j0}$ is $O([\log(V)]^2)/\lambda_{\mathbb{P}_{j0}}$ if $\lambda_{\mathbb{P}_{j0}} > 0$. This proves Part (c) and completes the proof of the theorem. ■

References

[1] L. Tassiulas and A. Ephremides. Stability properties of constrained queueing systems and scheduling policies for maximum throughput in multihop radio networks. *IEEE Trans. on Automatic Control, vol. 37, no. 12, pp. 1936-1949*, Dec. 1992.

[2] A. Eryilmaz and R. Srikant. Fair resource allocation in wireless networks using queue-length-based scheduling and congestion control. *IEEE/ACM Trans. Netw.*, 15(6):1333–1344, 2007.

[3] L. Huang and M. J. Neely. The optimality of two prices: Maximizing revenue in a stochastic network. *IEEE/ACM Transactions on Networking, Vol. 18, No.2*, April 2010.

[4] R. Urgaonkar and M. J. Neely. Opportunistic scheduling with reliability guarantees in cognitive radio networks. *IEEE Transactions on Mobile Computing, vol. 8, no. 6, pp. 766-777*, June 2009.

[5] Y. Yi and M. Chiang. Stochastic network utility maximization: A tribute to Kelly's paper published in this journal a decade ago. *European Transactions on Telecommunications*, vol. 19, no. 4, pp. 421-442, June 2008.

[6] L. Georgiadis, M. J. Neely, and L. Tassiulas. *Resource Allocation and Cross-Layer Control in Wireless Networks*. Foundations and Trends in Networking Vol. 1, no. 1, pp. 1-144, 2006.

[7] M. J. Neely. Super-fast delay tradeoffs for utility optimal fair scheduling in wireless networks. *IEEE Journal on Selected Areas in Communications (JSAC), Special Issue on Nonlinear Optimization of Communication Systems, vol. 24, no. 8, pp. 1489-1501*, Aug. 2006.

[8] M. J. Neely. Optimal energy and delay tradeoffs for multi-user wireless downlinks. *IEEE Transactions on Information Theory vol. 53, no. 9, pp. 3095-3113*, Sept. 2007.

[9] L. Huang and M. J. Neely. Delay reduction via Lagrange multipliers in stochastic network optimization. *IEEE Transactions on Automatic Control, Volume 56, Issue 4, pp. 842-857*, April 2011.

[10] S. Moeller, A. Sridharan, B. Krishnamachari, and O. Gnawali. Routing without routes: The backpressure collection protocol. *9th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN), pp. 279-290*, 2010.

[11] E. Athanasopoulou, L. X. Bui, T. Ji, R. Srikant, and A. Stolyar. Backpressure-based packet-by-packet adaptive routing in communication networks. *IEEE/ACM Trans. on Networking*, to appear.

[12] J. Ryu, V. Bhargava, N. Paine, and S. Shakkottai. Back-pressure routing and rate control for icns. *Proceedings of MobiCom*, 2010.

[13] L Bui, R Srikant, and A Stolyar. Optimal resource allocation for multicast sessions in multi-hop wireless networks. *Philosophical Transactions of The Royal Society, Series A*, pages 2059–2074, Jan 2008.

[14] L. Bui, R. Srikant, and A. Stolyar. Novel architectures and algorithms for delay reduction in back-pressure scheduling and routing. *Proceedings of IEEE INFOCOM 2009 Mini-Conference*, April 2009.

[15] D. P. Bertsekas and R. G. Gallager. *Data Networks*. Prentice Hall, 1992.

[16] L. Huang and M. J. Neely. Max-weight achieves the exact $[O(1/V), O(V)]$ utility-delay tradeoff under Markov dynamics. *arXiv:1008.0200v1*, 2010.

[17] D. P. Bertsekas, A. Nedic, and A. E. Ozdaglar. *Convex Analysis and Optimization*. Boston: Athena Scientific, 2003.

[18] G. B. Folland. *Real Analysis: Modern Techniques and their Applications*. Wiley, 2nd edition, 1999.

[19] L. Huang and M. J. Neely. Utility optimal scheduling in energy harvesting networks. *Proceedings of ACM International Symposium on Mobile Ad Hoc Networking and Computing (MOBIHOC)*, May 2011.

[20] M. J. Neely, E. Modiano, and C. Li. Fairness and optimal stochastic control for heterogeneous networks. *IEEE/ACM Trans. on Networking, vol. 16, no. 2, pp. 396-409*, April 2008.