Capacity Achieving Distributed Scheduling With Finite Buffers

Dongyue Xue, *Student Member, IEEE*, Robert Murawski, *Member, IEEE*, and Eylem Ekici, *Senior Member, IEEE*

***Abstract*—In this paper, we propose a distributed cross-layer scheduling algorithm for wireless networks with single-hop transmissions that can guarantee finite buffer sizes and meet minimum utility requirements. The algorithm can achieve a utility arbitrarily close to the optimal value with a tradeoff in the buffer sizes. The finite buffer property is not only important from an implementation perspective, but, along with the algorithm, also yields superior delay performance. In addition, another extended algorithm is provided to help construct the upper bounds of per-flow average packet delays. A novel structure of Lyapunov function is employed to prove the utility optimality of the algorithm with the introduction of novel virtual queue structures. Unlike traditional back-pressure-based optimal algorithms, our proposed algorithm does not need centralized computation and achieves fully local implementation without global message passing. Compared to other recent throughput/utility-optimal CSMA distributed algorithms, we illustrate through rigorous numerical and implementation results that our proposed algorithm achieves far better delay performance for comparable throughput/utility levels.**

***Index Terms*—Congestion control, distributed implementations, finite buffers, network scheduling.**

I. Introduction

CROSS-LAYER design of congestion control and link scheduling algorithms is one of the most challenging topics for wireless networks when bandwidth and quality-of-service (QoS) requirements are imposed. It is well known that the Maximal Weight Matching (MWM) algorithm [1] is throughput-optimal, i.e., it can stabilize the network for all arrival rates that lie within the network capacity region. In the grain of the seminal work [1], a group of optimal scheduling algorithms has been proposed for wireless networks with time-varying channels [2]–[5]. Cross-layer algorithms with congestion controllers at transport layer have been proposed in [6]–[9]. Delay issues have been addressed in [10]–[13]. However, the high complexity of these centralized algorithms (NP-hard for general interference models [14]) renders them impractical for implementation. Low-complexity and distributed alternatives have been proposed in [15]–[19], but they can only guarantee a fraction of the capacity region in general.

Recently, a class of random access algorithms has been introduced that achieves optimal throughput [25]–[32]. These algorithms can be implemented in a distributed manner with local feedback from surrounding nodes. However, the delay performance of these algorithms, as evaluated through simulations, leaves room for significant improvements [27]. Moreover, the delay analysis still remains elusive despite some preliminary results [32] that show a polynomial delay upper bound for *a fraction* of the network capacity region.

In this paper, we propose a cross-layer random-access-based scheduling algorithm for wireless networks with finite buffers and single-hop transmissions. The algorithm is theoretically shown to asymptotically achieve optimal throughput with growing buffer size. Our numerical and implementation results confirm the achievement of optimal throughput/utility and reveal superior delay performance of our algorithm.

In the algorithm development, for each communication link l in the network, we maintain an actual packet queue U_l and construct two virtual queues: a *virtual service queue* Z_l to guarantee the minimum utility requirement and a *virtual queue* Q_l that acts as a weight on the actual queue U_l when scheduling link rates. The cross-layer algorithm is composed of three parts: a regulator that regulates the evolution of virtual queues, a congestion controller that regulates data packet admission from the packet generator, and a link rate scheduler employing the Glauber dynamics method [27], [29], [32]. Moreover, we make use of the time–scale separation assumption (the underlying Markov chain of the scheduler converges instantaneously to its steady-state distribution compared to link weight adaptation rate of the scheduler) employed in [25] and [27], which is justified in [30] and [31].

Network stability for algorithms such as [32] and [36] is shown via the negative drift of Lyapunov function (or the positive recurrence of the underlying Markov chain) when the actual queue backlogs U_l are large enough. In other words, the network stability is achieved at the expense of large values of U_l, which leads to large average packet delays. Different from those algorithms, the link rate scheduler of our algorithm assigns the random access probabilities of a link l as the *product* of the packet queue backlog U_l and the virtual queue backlog Q_l. Consequently, the network stability is achieved by shifting the burden from actual packet queues to our proposed virtual

Manuscript received July 15, 2012; revised July 24, 2013; accepted January 10, 2014; approved by IEEE/ACM TRANSACTIONS ON NETWORKING Editor K. Kar. This work was supported in part by the NSF under Grant No. CCF-0914912. A preliminary version of this paper appeared in the Proceedings of the International Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks (WiOpt), Paderborn, Germany, May 14–18, 2012.

D. Xue and E. Ekici are with the Department of Electrical and Computer Engineering, The Ohio State University, Columbus, OH 43210 USA (e-mail: xued@ece.osu.edu; ekici@ece.osu.edu).

R. Murawski was with the Department of Electrical and Computer Engineering, The Ohio State University, Columbus, OH 43210 USA. He is now with the NASA Glenn Research Center, Cleveland, OH 44107 USA (e-mail: Robert.W.Murawski@nasa.gov).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TNET.2014.2303093

queues Q_l, while the actual queues U_l are upper-bounded by a finite buffer size. As a direct result, the delay performance of the algorithm is significantly improved. The stability and utility optimality of the algorithm are proved through Lyapunov optimization techniques. Different from the traditional Lyapunov optimization techniques using quadratic Lyapunov functions, we employ a novel type of Lyapunov function by multiplying the virtual queue backlog Q_l to the quadratic term of actual queue backlogs U_l^2. We show that this particular structure of Lyapunov function leads to the product form of virtual and actual queue backlogs in the proposed algorithm.

Salient properties of our algorithm can be listed as follows: 1) All link-based queues in the network use finite-sized buffers, which leads to network stability; 2) Minimum utility requirements are satisfied for individual communication links; 3) A utility "ϵ-close" to the optimal network utility (expressed as the sum of link utilities, given minimum data rate requirement of individual links) is achieved with a tradeoff of $O(1/\epsilon)$ in the buffer size; 4) The link rate scheduler of our algorithm uses the CSMA paradigm with RTS/CTS handshake, which admits distributed implementation without global message passing.

The algorithm is shown both in simulation and implementation to have close-to-optimal throughput proved theoretically and display very favorable delay performances. Our main algorithm is further extended: 1) to construct individual *upper bounds for average link-based delay*; 2) to approximate the time–scale separation. An extension to multihop wireless networks is considered in [42].

The rest of the paper is organized as follows. Section II introduces the related work. In Section III, we present the network model, followed by approaches considered for wireless networks with single-hop transmissions. In Section IV, the distributed utility-optimal algorithm is described and its performance analyzed. We provide a modified algorithm to construct per-flow average delay upper bounds in Sections V. Two methods to approach time–scale separation are introduced in Section VI. We present implementation and numerical results in Sections VII and VIII, respectively. Finally, we conclude our work in Section IX.

II. Related Works

In this paper, we focus on the following aspects in algorithm performance: 1) throughput/utility optimality;) per-link minimum utility constraints; 3) bounded buffers/delays; and 4) distributed implementation. While the above aspects have been individually explored in one way or another in the existing literature, it is challenging to study them in an integrated framework. Throughput/utility optimality has been considered in a centralized setting in [1]–[9]. Minimum throughput/utility constraints has been discussed in [6]. A few recent works [20]–[24] in the literature proposed throughput/utility-optimal algorithms with finite buffers or worst-case delay guarantees. Specifically, the multihop scheduling algorithms in [20] and [21] guarantee a finite buffer size for queues in the network at the expense of the buffer occupancy of the source nodes, where *an infinite buffer size* in the network layer is assumed in each source node. Thus, the finite buffer size property in [20], [21] is not guaranteed for single-hop transmissions. Virtual queue structures are utilized in [22]–[24] to provide finite buffer size or worst-case delay performance. However, the above-mentioned algorithms are not well suited for distributed implementation due to their centralized nature and high complexity.

A class of random access algorithms have been proposed in [25]–[32] that achieve optimal network throughput. It has recently been shown in [32] that these algorithms employing the Glauber dynamics [27]–[29] can achieve polynomial delay upper bound for *a fraction* of capacity region in networks with single-hop transmissions. Specifically, the link weights in the Glauber dynamics are selected in the form of U_l in [32], $\log U_l$ in [27], $\log\log U_l$ in [28], $\log U_l/y(U_l)$ in [29], where $y(\cdot)$ is a function that increases arbitrarily slowly and U_l is the packet queue length for a link. The resulting schedulers of the Glauber dynamics react to changes in U_l. However, the queue lengths must be large enough to approach maximal weight matching [27], [29], or to ensure a negative Lyapnov drift [28], [32], resulting in undesirable delay performance. Instead, in our algorithm, the expense of large queue lengths is partially shifted to virtual queues Q_l, which results in significantly more favorable delay performance.

III. Network Model

A. Network Elements

We consider a time-slotted wireless network with single-hop transmissions consisting of N nodes and L directional links, with the node and the link sets denoted as $\mathcal{N}$ and $\mathcal{L}$, respectively. Each directional link $l=(m,n)\in\mathcal{L}$ carries a single-hop flow from a source node m to a destination node n, with $m,n\in\mathcal{N}$. The multihop extension of this model is presented in detail in [42].

Let $G=(\mathcal{L},\mathcal{E})$ be the interference graph (also called conflict graph) associated with the network topology. There is an edge $(l,j)\in\mathcal{E}$ if simultaneous transmissions over links l and j are not possible. We denote the set of interfering links of link $l\in\mathcal{L}$ as $\mathcal{N}(l)\triangleq\{j:(l,j)\in\mathcal{E}\}\cup\{l\}$, where we let $l\in\mathcal{N}(l)$ for analytical clarity. For simplicity of analysis, in each time-slot, each link's packet transmission rate is normalized and can take a fractional value in [0,1]. Thus, a feasible schedule at time-slot t can be represented by a vector $\mu(t)=(\mu_l(t))_{l\in\mathcal{L}}\in[0,1]^L$ such that: $\mu_l(t)\mu_j(t)=0,\ \forall l\in\mathcal{L}$ and $\forall j\in\mathcal{N}(l)\setminus\{l\}$, where $\mu_l(t)$ is the scheduled link rate for link $l\in\mathcal{L}$ at time-slot t. For each feasible schedule $\mu(t)$, we can find a corresponding link set $\mathbf{x}\subset\mathcal{L}$, called an independent set, such that $l\in\mathbf{x}$ iff $\mu_l(t)>0$. Let $\mathcal{I}$ be the set of all independent sets in the network. Note that for a link $l=(m,n)\in\mathcal{L}$ connecting m to n, with $m,n\in\mathcal{N}$, its scheduled link rate can also be denoted as $\mu_{mn}(t)=\mu_l(t)$.

We denote by $f_l(\cdot),\ l\in\mathcal{L}$, the utility functions of the time-average data transmission rate. As convention, we assume the utility functions are positively valued, concave, continuously differentiable, and strictly increasing, with $f_l(0)=0,\ l\in\mathcal{L}$. Let $d_l\geq 0$ be the minimum utility constraint associated with link $l\in\mathcal{L}$, i.e., the utility of the time-average data rate over link l must be greater than or equal to d_l.

According to [1] and [6], we define the capacity region Λ of the considered network as the closure of all feasible admitted

rate vectors $(a_l)_{l\in\mathcal{L}}$ each stabilizable by some scheduling algorithm. The capacity region Λ is convex, closed, and bounded [6]. Without loss of generality, we assume that $(f_l^{-1}(d_l))_{l\in\mathcal{L}}$ is inside Λ, i.e., there exists $\epsilon > 0$ with $(f_l^{-1}(d_l)+\epsilon)_{l\in\mathcal{L}} \in \Lambda$. To assist the analysis in the following sections, we let $(a^*_{l,\epsilon})_{l\in\mathcal{L}}$ denote a solution to the following optimization problem:

$$\begin{aligned}&\left(a^*_{l,\epsilon}\right)_{l\in\mathcal{L}} \in \arg\max_{(a_l)} \sum_{l\in\mathcal{L}} f_l(a_l)\\ &\text{s.t. } a_l \geq f_l^{-1}(d_l) \text{ and } (a_l+\epsilon)_{l\in\mathcal{L}} \in \Lambda. \end{aligned} \tag{1}$$

Note that $\sum_{l\in\mathcal{L}} f_l(a^*_{l,\epsilon})$ can be regarded as the overall utility ϵ-close to the optimality. Then, according to [34] we have

$$\lim_{\epsilon\to 0^+} \sum_{l\in\mathcal{L}} f_l\left(a^*_{l,\epsilon}\right) = \sum_{l\in\mathcal{L}} f_l\left(a^*_l\right)$$

where $(a^*_l)_{l\in\mathcal{L}}$ is a solution to the following optimization:

$$\begin{aligned}&(a^*_l)_{l\in\mathcal{L}} \in \arg\max_{(a_l)} \sum_{l\in\mathcal{L}} f_l(a_l)\\ &\text{s.t. } a_l \geq f_l^{-1}(d_l) \text{ and } (a_l)_{l\in\mathcal{L}} \in \Lambda. \end{aligned} \tag{2}$$

In most practical systems, congestion control is needed to avoid overloading the networks. Motivated by this, we consider a closed-loop system, i.e., the admission/generation of packets are determined by the scheduling algorithm. Thus, the packet delay is defined to be the period that starts when the packet gets admitted into the network and ends when it gets served. In addition, we assume the physical packet generator of each link's source is constantly backlogged. Thus, a congestion controller is needed to determine the rate of packet generation/admission into the network.[1] Let $A_l(t)$ denote the admitted rate of link l from the packet generator in time-slot t. Since the link's packet transmission capacity is normalized to 1, we assume that the admitted rate $A_l(t)$ is also normalized with respect to the link capacity. For the congestion controller, we also assume that $A_l(t)$ is bounded above by some constant $\mu_M > 0$, $\forall l \in \mathcal{L}\ \forall t$, i.e., the source node of a link can receive at most μ_M packets from the packet generator in any time-slot.[2]

B. Network Constraints and Approaches

To present network stability, we begin with a definition of queue stability with respect to a generic queue backlog $X(t)$: the queue is *stable* if $\limsup_{t\to\infty}(1/t)\sum_{\tau=0}^{t-1}\mathbb{E}\{X(\tau)\} < \infty$. Let $U_l(t)$ be the actual packet queue backlog of link l, maintained at the source node of the link. Then, the network is *stable* if queues $U_l(t)$ are stable, $\forall l \in \mathcal{L}$. The queue dynamics have the following evolution:

$$U_l(t+1) = \left[U_l(t) - \mu_l(t)\right]^+ + A_l(t) \qquad \forall l \in \mathcal{L} \tag{3}$$

[1]Note that the constantly backlogged packet generator is not necessarily a physical queue. It can represent an application controlling its packet generation, e.g., a variable rate multimedia encoder. After packets are generated and admitted to the source node, they are delivered from the source to the destination node via the network layer.

[2]To be specific, we require $\mu_M \leq q_M$ to guarantee the finite buffer sizes in Proposition 1, where q_M denotes the uniform buffer size introduced in Section IV.

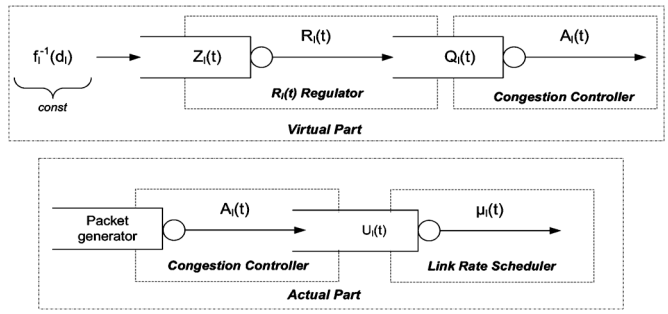


Fig. 1. Queue relationships.

where we recall $\mu_l(t) \in [0,1]$ is the scheduled link rate in time-slot t and we define the operator $[\cdot]^+$ as $[\cdot]^+ \triangleq \max\{\cdot, 0\}$. Note that the data transmitted for a link l cannot exceed its backlog and a feasible scheduling algorithm may be independent of queue backlog information. For simplicity of analysis, we assume the input rates $A_l(t)$ will be added to the queue backlogs at the end of time-slot t.

For each link l, we construct a virtual queue $Q_l(t)$ at link l's source node to later assist the development of our proposed algorithm in Section IV. We denote by $R_l(t)$ the virtual input rate to queue $Q_l(t)$ at the end of time-slot t, and denote by r_l the time-average of $R_l(t)$. We place an upper bound v_M on $R_l(t)$ with

$$\max_{(a_l)_{l\in\mathcal{L}}\in\Lambda} \max_{l\in\mathcal{L}} a_l \leq v_M \leq \mu_M$$

and update queue $Q_l(t)$ *virtually* as follows:

$$Q_l(t+1) = \left[Q_l(t) - A_l(t)\right]^+ + R_l(t) \tag{4}$$

with $Q_l(0) = 0$. Considering the admitted rate $A_l(t)$ as the service rate of the virtual queue, if $Q_l(t)$ is stable, then the time-average admitted rate a_l of link l satisfies

$$a_l \triangleq \liminf_{t\to\infty} \frac{1}{t}\sum_{\tau=0}^{t-1} \mathbb{E}\left\{A_l(\tau)\right\} \geq r_l \triangleq \liminf_{t\to\infty} \frac{1}{t}\sum_{\tau=0}^{t-1} \mathbb{E}\left\{R_l(\tau)\right\}. \tag{5}$$

We construct a virtual service queue $Z_l(t)$ at the source node of each $l \in \mathcal{L}$ with its virtual queue dynamics given by

$$Z_l(t+1) = \left[Z_l(t) - R_l(t)\right]^+ + f_l^{-1}(d_l) \tag{6}$$

with $Z_l(0) = 0$. When $Z_l(t)$ and $Q_l(t)$ are stable, we have $f_l(a_l) \geq f_l(r_l) \geq d_l$, i.e., the minimum utility constraint is satisfied.

The queue relationships between queues $Z_l(t)$, $Q_l(t)$, and $U_l(t)$ are shown in Fig. 1, where we note that $A_l(t)$ represents both the *physical* packet admission rate (i.e., the arrival rate to actual queue $U_l(t)$) and the *virtual* service rate of $Q_l(t)$. Physical packets are only involved in the "actual part" block. We note that the average packet delay is only affected by the number of packets in the actual packet queue. The size of virtual queues have no effect on the delay as they only determine rate of packet arrivals. Even if a virtual queue has a large number of virtual packets, they are not propagated in the network through an actual packet queue and actual packets never passed through virtual queues.

In addition, the reason we construct *two* virtual queues is that the utility optimality and minimum utility constraint cannot be both guaranteed with the employment of one single virtual queue. Specifically, for link $l \in \mathcal{L}$, virtual queue $Z_l(t)$ monitors the achievement of minimum utility constraint and virtual queue $Q_l(t)$ acts as a weight for actual packet queue $U_l(t)$ in the link rate scheduler (introduced in the next section) that approaches a maximal weight matching in an aim to achieve optimal utility. For instance, in Section V, only one (modified) virtual queue $Z_l(t)$ is employed in a modified algorithm to construct average delay upper bounds at the expense of losing utility optimality.

IV. Proposed Algorithm for Single-Hop Networks

In this section, employing the queue structures developed in Section III-B, we propose a distributed cross-layer scheduling algorithm ***ALG*** for the wireless network model introduced in Section III such that ***ALG*** can stabilize the network and satisfy the minimum utility constraints. Moreover, ***ALG*** achieves a utility arbitrarily close to the optimal value $\sum_{l\in\mathcal{L}} f_l(a_l^*)$ under certain conditions related to queue buffer sizes that will be later given in Theorem 1.

Let $q_M \geq \mu_M$ be a control parameter indicating the uniform buffer size for $U_l(t)$ at each link $l \in \mathcal{L}$, where we recall μ_M limits the maximal rate of packet arrival to a source node. ***ALG*** ensures that all queue backlogs are bounded by q_M, *which further leads to a bounded average packet delay* according to Little's Law. We construct the ***ALG*** with the following three functions: $R_l(t)$ regulator, a congestion controller, and a link rate scheduler, which decide on the arrival and service rates of all queues $\forall t$ (as shown in Fig. 1).

1) $R_l(t)$ *Regulator*:

$$\min_{0\leq R_l(t)\leq v_M} g_l\left(R_l(t)\right) \tag{7}$$

where $g_l(R_l(t)) \triangleq R_l(t)(((q_M-\mu_M)/q_M)Q_l(t) - Z_l(t)) - Vf_l(R_l(t))$ and $V > 0$ is a control parameter. Note that $g_l(\cdot)$ is convex and $\min_{R_l(t)} g_l(R_l(t)) \leq g_l(0) = 0$.

$R_l(t)$ regulator operates locally at the source node of link $l \in \mathcal{L}$ and actually regulates the virtual queue evolutions. Since $R_l(t)$ is the input rate to the virtual queue $Q_l(t)$ and the service rate to the virtual queue $Z_l(t)$, $R_l(t)$ is weighted by the (weighted) difference between $Q_l(t)$ and $Z_l(t)$ in (7). We will show later in Theorem 1 that with a large control parameter V, ***ALG*** can approach the optimal utility. With the numerical results in Section VIII-A, we will further analyze the effect of V on the throughput performance and the convergence rates of virtual queues $Q_l(t)$ and $Z_l(t)$.

2) *Congestion Controller*:

$$\max_{0\leq A_l(t)\leq \mu_M} A_l(t)\left(q_M-\mu_M-U_l(t)\right). \tag{8}$$

Specifically, when $q_M - \mu_M - U_l(t) < 0$, $A_l(t)$ is set to zero; otherwise, $A_l(t) = \mu_M$.

The congestion controller operates locally at the source node of link $l \in \mathcal{L}$, with the decision sent as feedback to the packet generator to generate $A_l(t)$ data packets to be admitted to the source node at the end of time-slot t. The congestion controller also guarantees the finite buffer size property (see Proposition 1 in this section for detail).

3) *Link Rate Scheduler*: The link rate scheduler follows the form of parallel Glauber dynamics [27], [29], [32] with a probability vector $(p_l(t))_{l\in\mathcal{L}} \in [0,1]^L$ to be defined later in Proposition 2. The scheduler operates as follows:
   1) Randomly select an independent set $\mathbf{x}(t) \in \mathcal{I}$ with probability (w.p.) $p_{\mathbf{x}(t)}$, such that

$$\sum_{\mathbf{x}(t)\in\mathcal{I}} p_{\mathbf{x}(t)} = 1 \text{ and } \cup_{p_{\mathbf{x}(t)}>0}\mathbf{x}(t) = \mathcal{L}. \tag{9}$$

   2) $\forall l \in \mathbf{x}(t)$:
      - if $\sum_{j\in\mathcal{N}(l)\setminus\{l\}}\mu_j(t-1) = 0$: $\mu_l(t) = 1$, w.p. $p_l(t)$; $\mu_l(t) = 0$, w.p. $1-p_l(t)$;
      - else, $\mu_l(t) = 0$.

      $\forall l \in \mathcal{L}\setminus\mathbf{x}(t)$, $\mu_l(t) = \mu_l(t-1)$.

It is not difficult to check that the link set corresponding to the schedule $(\mu_l(t))_{l\in\mathcal{L}}$ is an independent set [27] and evolves as a discrete-time Markov chain (DTMC). The distributed implementation of the link rate scheduler is provided in Appendix A, which is based on the CSMA paradigm with RTS/CTS handshake and requires no global signaling or message passing.

To analyze the performance of ***ALG***, we first introduce the following two propositions. Proposition 1 shows that ***ALG*** has a deterministic worst-case upper bound for all queues $U_l(t)$, $\forall l \in \mathcal{L}$. In Proposition 2, we show that the link rate scheduler finds a schedule approaching arbitrarily close to that of a maximal weight matching scheduler with high probability, when virtual queues $Q_l(t)$ are large enough. Using these two propositions, we derive the main results of utility optimality and virtual queue stability in Theorem 1.

Proposition 1: Employing ***ALG***, if $\mu_M \leq q_M$, then each actual queue backlog in the network has a deterministic worst-case bound

$$U_l(t) \leq q_M \qquad \forall t,\ \forall l \in \mathcal{L} \tag{10}$$

where we recall that q_M indicates the uniform link-based buffer size.

Proof: We use mathematical induction on time-slot in the proof. When $t = 0$, we have $U_l(0) = 0\ \forall l \in \mathcal{L}$.

Now suppose in time-slot t, $U_l(t) \leq q_M\ \forall l \in \mathcal{L}$. In the induction step, for any given l, we consider two separate cases. In the first case, $U_l(t) \leq q_M - \mu_M$, then according to the queue dynamics (3), $U_l(t+1) \leq U_l(t) + \mu_M \leq q_M$. Otherwise, $U_l(t) > q_M - \mu_M$, and according to the congestion controller (8), we have $A_l(t) = 0$, which leads to $U_l(t+1) \leq U_l(t) \leq q_M$ by the queue dynamics (3).

Since the above analysis is true for any given $l \in \mathcal{L}$, the induction step holds, i.e., $U_l(t+1) \leq q_M\ \forall l \in \mathcal{L}$, which completes the proof. ■

Let $w_l(t) = (1/q_M)U_l(t)Q_l(t)$. The Glauber dynamics structure of the link rate scheduler leads to the following proposition.

Proposition 2: Let $p_l(t) = e^{w_l(t)}/(1+e^{w_l(t)})$, $\forall l \in \mathcal{L}$, in the link rate scheduler. For any given ϵ' and δ' satisfying $0 < \epsilon'$, $\delta' < 1$, we can find a constant $B(\epsilon',\delta') > 0$ such that for any

time-slot t, with probability greater than $(1-\delta')$, the link rate scheduler finds a schedule $(\mu_l(t))_{l\in\mathcal{L}}$, satisfying

$$\sum_{l\in\mathcal{L}} w_l(t)\mu_l(t) \geq (1-\epsilon')w^*(t), \text{ whenever } \|\mathbf{q}(t)\| > B \quad (11)$$

where $\mathbf{q}(t) \triangleq (w_l(t))_{l\in\mathcal{L}}$, $\|\mathbf{q}(t)\| \triangleq \sum_{l\in\mathcal{L}} w_l(t)$, and $w^*(t)$ is the result of a maximal weight matching scheduler: $w^*(t) \triangleq \max_{\mathbf{x}\in\mathcal{I}} \sum_{l\in\mathbf{x}} w_l(t)$.

Proposition 2 directly follows [27, Proposition 2] with *the time–scale separation assumption* (the DTMC of the schedules chosen by the scheduler is in steady state in each time-slot), so we omit the proof for brevity. By the time–scale separation assumption, we assume that the schedule converges much faster to its steady-state distribution compared to the link weight ($w_l(t) = (1/q_M)U_l(t)Q_l(t)$) adaptation rate of the scheduler. This assumption has been used in previous works such as [25] and [27]. While the queue backlogs do not stay constant, we can update the link weights less frequently (i.e., the link weights stay constant for a sufficiently long period of time) to approach the time–scale separation, which is further discussed in Section VI-B.

Remark 1 (Implications of the Propositions): A maximal weight matching scheduler that obtains $w^*(t)$ in each time-slot t is usually centralized and computation-prohibitive, but can lead to optimal throughput/utility [1], [6], [22]. Proposition 2 states that the proposed link rate scheduler can find a schedule using weight matching ϵ'—close to $w^*(t)$ with high probability $(1-\delta')$, when link weights $w_l(t)$ are large enough. By definition, $w_l(t)$ is the product of the virtual queue $Q_l(t)$ and the buffer occupancy $U_l(t)/q_M$. Since $U_l(t) \leq q_M$ holds $\forall l \in \mathcal{L}$ by Proposition 1, (11) in Proposition 2 holds with high probability when some virtual queues $Q_l(t)$ are large enough with nonzero buffer occupancy. Since the actual queues are still upper-bounded by q_M, a bounded average packet delay is implied by Little's Law. In other words, the design of the link weight $w_l(t)$ allows us to "shift the burden" of large queue backlogs from the actual queues $U_l(t)$ to the virtual queues $Q_l(t)$. Note that Proposition 2 also holds for link rate schedulers with link weights $w_l(t) = \log U_l(t)$ proposed in Q-CSMA [27], $w_l(t) = \log\log U_l(t)$ proposed in [28], $w_l(t) = \log U_l(t)/y(U_l(t))$ in [29], where $y(\cdot)$ is some function that increases arbitrarily slowly, and $w_l(t)$ is in the form of $U_l(t)$ in [32]. However, these schedulers approach a maximal weight matching scheduler when *the actual queues* are large enough, leading to large delays. On the other hand, our algorithm ***ALG*** does the same while limiting the maximum buffer size, which guarantees bounded average delay.

For notational simplicity, we define $\gamma \triangleq (1-\epsilon')(1-\delta')$, where we note that ϵ' and δ' in Proposition 2 can take values arbitrarily small. We present our main results in Theorem 1.

Theorem 1: Given some $\epsilon > 0$ arbitrarily small and $\gamma > \max_{l\in\mathcal{L}}(f_l^{-1}(d_l)/a^*_{l,\epsilon} + (1/2)\epsilon)$, if

$$q_M > \frac{\mu_M^2+1}{\gamma\epsilon} + \mu_M \quad (12)$$

ALG ensures that the virtual queues are stable

$$\limsup_{t\to\infty} \frac{1}{t}\sum_{\tau=0}^{t-1}\sum_{l\in\mathcal{L}} \mathbb{E}\left\{Q_l(\tau)+Z_l(\tau)\right\} \leq \frac{B_2+Vg_M}{\delta_1} \quad (13)$$

where $B_1 \triangleq (1/2)\mu_M q_M L + \mu_M^2((q_M-\mu_M)/q_M)L + (1/2)\mu_M^2 L + (1/2)\sum_{l\in\mathcal{L}}(f_l^{-1}(d_l))^2$, $B_2 \triangleq B_1 + \gamma B(\max_{l\in\mathcal{L}} a^*_{l,\epsilon} + \epsilon)$

$$\delta_1 \triangleq \min\left\{\frac{\gamma\epsilon(q_M-\mu_M)-\mu_M^2-1}{2q_M}, \min_{l\in\mathcal{L}}\left[\gamma\left(a^*_{l,\epsilon}+\frac{1}{2}\epsilon\right) - f_l^{-1}(d_l)\right]\right\}$$

and

$$g_M \triangleq \limsup_{t\to\infty}\frac{1}{t}\sum_{\tau=0}^{t-1}\sum_{l\in\mathcal{L}}\mathbb{E}\left\{f_l\left(R_l(\tau)\right)\right\} - \gamma\sum_{l\in\mathcal{L}} f_l\left(a^*_{l,\epsilon}+\frac{1}{2}\epsilon\right) \leq \sum_{l\in\mathcal{L}} f_l(\mu_M) - \gamma\sum_{l\in\mathcal{L}} f_l\left(a^*_{l,\epsilon}+\frac{1}{2}\epsilon\right).$$

In addition, ***ALG*** can achieve

$$\sum_{l\in\mathcal{L}} f_l(a_l) \geq \gamma\sum_{l\in\mathcal{L}} f_l\left(a^*_{l,\epsilon}+\frac{1}{2}\epsilon\right) - \frac{B_2}{V}. \quad (14)$$

Proof: We prove Theorem 1 in Appendix B, with the employment of Propositions 1 and 2. ■

Remark 2 (Algorithm Performance):

1) *Network Stability and Minimum Utility Constraints*: Equation (10) in Proposition 1 indicates that ***ALG*** stabilizes the actual queue backlogs. As an immediate result, the network is stable. Equation (13) in Theorem 1 shows that the virtual queues are stable and, hence, the minimum utility constraints are satisfied. Note that from (13), the virtual queues can grow with an increased value of parameter V. In fact, a large V implies a slow convergence rate of virtual queues, which will be explained in more detail with numerical results in Section VIII-A.
2) *Utility Optimality and Buffer Size of Order $O(1/\epsilon)$*: The inequality (14) gives the lower bound of the utility that ***ALG*** can achieve. Since the constant B_2 is independent of V and γ can be chosen arbitrarily close to 1 by definition, (14) ensures that ***ALG*** can achieve a utility arbitrarily close to $\sum_{l\in\mathcal{L}} f_l(a^*_{l,\epsilon} + (1/2)\epsilon)$ by selecting a V large enough and by choosing small ϵ', δ' such that γ is close to 1. When ϵ tends to 0, ***ALG*** can achieve a utility arbitrarily close to the optimal value $\sum_{l\in\mathcal{L}} f_l(a^*_l)$ with the tradeoff in buffer size q_M, which is of order $O(1/\epsilon)$ as shown in (12). Note that the only assumption for Proposition 2 (and hence Theorem 1) to hold is the time–scale separation assumption, which has also been employed in [25] and [27] and justified in [30] and [31]. When this assumption holds, the buffer size has an order of $O(1/\epsilon)$ that is independent of the network size, i.e., *an order-optimal delay performance* [33] *is achieved*.

As a result of Proposition 1 and Theorem 1, we show in the following corollary that the average queuing delay for each link is upper-bounded.

Corollary 1: Let $M(T)$ be the number of packets admitted to any given queue $U_l(t)$ by any given time-slot T, and let W_j be the queuing delay of packet j by time-slot T, $j = 1, 2, \ldots, M(T)$. Under ***ALG***, the average queuing delay is upper-bounded, i.e., for any given link $l \in \mathcal{L}$ and any $0 < \epsilon_0 < f_l^{-1}(d_l)$

$$\limsup_{T\to\infty} \frac{1}{M(T)} \sum_{j=1}^{M(T)} W_j \leq \frac{q_M}{f_l^{-1}(d_l) - \epsilon_0}. \tag{15}$$

Proof: According to the derivation of Little's Law, we have the following equation:

$$\sum_{t=1}^{T} U_l(t) = \sum_{j=1}^{M(T)} W_j$$

from which we have

$$q_M \geq \frac{1}{T} \sum_{t=1}^{T} U_l(t) = \frac{1}{T} \sum_{j=1}^{M(T)} W_j.$$

Note that under ***ALG***, the time-averaged admitted rate is lower-bounded by $f_l^{-1}(d_l)$ (since virtual queues $Q_l(t)$ and $Z_l(t)$ are stable from Theorem 1), i.e.,

$$\liminf_{T\to\infty} \frac{M(T)}{T} \geq f_l^{-1}(d_l).$$

Hence, there exists $K > 0$ such that for all $T > K$

$$\inf_{t\geq T} \frac{M(t)}{t} \geq f_l^{-1}(d_l) - \epsilon_0.$$

Thus, we have, for all $T > K$

$$\frac{1}{M(T)} \sum_{j=1}^{M(T)} W_j \leq \frac{1}{T(f_l^{-1}(d_l) - \epsilon_0)} \sum_{i=1}^{M(T)} W_j \leq \frac{q_M}{f_l^{-1}(d_l) - \epsilon_0}.$$

We take the limsup of T on both sides of the above inequality, proving (15). ∎

V. Constructing Individual Average Delay Upper Bound

For delay-sensitive traffic, the delay performance of a network is of more significance than the utility optimality. In this section, we propose an algorithm modified from the original one that guarantees link-based minimum utility constraint. Specifically, we employ a modified version of virtual queues $Z_l(t)$ without using $Q_l(t)$ in the modified algorithm. Note that the employment of a single virtual queue $Z_l(t)$ no longer guarantees the utility optimality. However, at the expense of losing utility optimality, we can construct nontrivial link-based delay upper bounds for the modified algorithm.

Instead of setting q_M as the global buffer size, we now define and employ in this section different buffer sizes for each link, i.e., q_l, $l \in \mathcal{L}$. Instead of (6), we now update the virtual service queue as

$$Z_l(t+1) = [Z_l(t) - A_l(t)]^+ + f_l^{-1}(d_l),\ \forall l \in \mathcal{L}$$

where we recall that $(d_l)_{l\in\mathcal{L}}$ is the link-based minimum utility constraint, and the minimum utility constraint is met when the virtual queues $Z_l(t)$ are stable. Without loss of generality, we assume $d_l > 0,\ \forall l \in \mathcal{L}$, in this section. The algorithm is composed of two functions.

1) *Congestion Controller*:

$$\max_{0\leq A_l(t)\leq \mu_M} A_l(t)\left(q_l - \mu_M - U_l(t)\right)$$

which is modified from the congestion controller (8) introduced in Section IV by replacing q_M with q_l.

2) *Link Rate Scheduler*: Let $w_l(t) = (1/q_l)Z_l(t)U_l(t),\ \forall l \in \mathcal{L}$. The link rate scheduler is the same as that in Section IV with $p_l(t) = e^{w_l(t)}/(1 + e^{w_l(t)}),\ l \in \mathcal{L}$.

It is not difficult to check that $U_l(t) \leq q_l,\ \forall l \in \mathcal{L}\ \forall t$, (i.e., the finite buffer sizes and network stability) and Proposition 2 still hold for the above algorithm. In addition, the minimum utility constraints are satisfied, which is formally stated in the following theorem.

Theorem 2: For any arbitrarily vector $(\epsilon_l)_{l\in\mathcal{L}}$ such that

$$\epsilon_l > 0,\ \forall l \in \mathcal{L},\ \text{and}\ \left(f_l^{-1}(d_l) + \epsilon_l\right)\big)_{l\in\mathcal{L}} \in \Lambda \tag{16}$$

given $\gamma \geq \max_{l\in\mathcal{L}} f_l^{-1}(d_l)/f_l^{-1}(d_l) + \epsilon_l$, if

$$q_l > \frac{\mu_M^2 + 1}{2\left[(\gamma-1)f_l^{-1}(d_l) + \gamma\epsilon_l\right]} + \mu_M \tag{17}$$

then the virtual service queues $Z_l(t)$ are stable as shown in (18) at the bottom of the page.

The proof for Theorem 2 is similar to that of Theorem 1 developed in Appendix B and is omitted for brevity. From Theorem 2, we can construct the delay upper bounds, given in the following corollary.

Corollary 2: The modified algorithm can guarantee an upper bound for the average delay D_l for link $l \in \mathcal{L}$

$$D_l \leq \frac{1}{f^{-1}(d_l)}\left(\mu_M + \frac{\mu_M^2 + 1}{2\epsilon_l}\right) + \eta \tag{19}$$

where η is any arbitrarily small positive constant and $(\epsilon_l)_{l\in\mathcal{L}}$ satisfies (16).

Proof: The proof of Corollary 2 and the design of buffer sizes are given in Appendix C. ∎

Remark 3: In the above algorithm, we employ $Z_l(t)$ instead of $Q_l(t)$ in the weight $w_l(t),\ \forall l \in \mathcal{L}$. From the queue dynamics,

$$\limsup_{t\to\infty} \frac{1}{t} \sum_{\tau=0}^{t-1} \sum_{l\in\mathcal{L}} \mathbb{E}\left\{Z_l(\tau)\right\} \leq \frac{\sum_{l\in\mathcal{L}} \left\{q_l^2 f_l^{-1}(d_l) + (q_l - \mu_M)\left(1 + f_l^{-1}(d_l)^2\right) + 2\gamma q_l(d_l + \epsilon_l)B\right\}}{\min_{l\in\mathcal{L}} \left\{2(q_l - \mu_M)\left[(\gamma-1)f_l^{-1}(d_l) + \gamma\epsilon_l\right] - \mu_M^2 - 1\right\}}. \tag{18}$$

$Z_l(t)$ can be small when the link service rate is greater than $f_l^{-1}(d_l)$, which results in a small link activation probability p_l. Hence, the algorithm no longer guarantees utility optimality. At the expense of losing utility optimality, nontrivial per-flow delay upper bounds can be derived and tailored by selecting different $(\epsilon_l)_{l\in\mathcal{L}}$ values, where $(\epsilon_l)_{l\in\mathcal{L}}$ can be chosen anywhere between the minimum data rate vector $(f_l^{-1}(d_l))_{l\in\mathcal{L}}$ and some rate vector on the boundary of the capacity region Λ. Since the average delay D_l is of order $O(1/\epsilon_l)$ from (19), we can choose $(\epsilon_l)_{l\in\mathcal{L}}$ as the difference between $(f_l^{-1}(d_l))_{l\in\mathcal{L}}$ and some rate vector on the boundary (or close to the boundary) of Λ to obtain tighter delay upper bounds.

VI. Approaching Time–Scale Separation

We recall that Proposition 2 (and hence Theorem 1) in Section IV is based on the time–scale separation assumption. This assumption requires the schedule determined by the link rate scheduler to converge to its steady-state faster than the rate at which link weights $w_l(t)$ change over time. In this section, we discuss two approaches to approximate this time–scale separation: 1) changing the link weights at a slow rate; 2) updating the link weights less frequently.

A. Small Changes in $w_l(t)$

In the link rate scheduler introduced for ***ALG*** in Section IV, we assign the link weights as

$$w_l(t) = \frac{\alpha}{q_M} U_l(t) Q_l(t)$$

where α is a small positive constant such that $w_l(t)$, and hence the link activation probability p_l, change slowly over time.

It is not difficult to show that Propositions 1 and 2 and Theorem 1 hold for this choice of link weights, with a minor change in the constants B and B_2. Hence, if time–scale separation holds, we can still expect an order of $O(1/\epsilon)$ in buffer sizes as in (12), which further induces that the average link delay is of order $O(1/\epsilon)$, i.e., order-optimal delay can be achieved. However, an α too small may reduce both the link activation probabilities and the responsiveness of the scheduler to the queue length variations. An alternative method to approach the time–scale separation is provided in Section VI-B.

B. Slow Updates in $w_l(t)$

We can make the DTMC of the schedules converge to the steady state distributions by updating the weights in the link rate scheduler periodically every T time-slots:

$$w_l(t) = \frac{1}{q_M} U_l(kT) Q_l(kT),\ kT \le t < (k+1)T, \qquad (20)$$

while other components of the algorithm remain the same, where T is the update period and k takes integer values.

Then, Proposition 1 still holds for ***ALG*** with slow updates, and we have the following theorem.

Theorem 3: Given some $\epsilon > 0$ arbitrarily small and $\gamma > \max_{l\in\mathcal{L}}(f_l^{-1}(d_l)/(a^*_{l,\epsilon} + (1/2)\epsilon))$, if

$$q_M > \frac{2(T-1)(1+\gamma\mu_M) + \mu_M^2 + 1}{\gamma\epsilon} + \mu_M \qquad (21)$$

ALG with slow updates ensures the stability of virtual queues

$$\limsup_{t\to\infty} \frac{1}{t} \sum_{\tau=0}^{t-1} \sum_{l\in\mathcal{L}} \mathbb{E}\left\{Q_l(\tau) + Z_l(\tau)\right\} \le \frac{B_4 + V g_M}{\delta_3}$$

where $B_4 \triangleq B_2 + (1+\gamma)\mu_M L(T-1)$ and

$$\delta_3 \triangleq \min\left\{ \frac{\gamma\epsilon(q_M - \mu_M) - \mu_M^2 - 1 - 2(T-1)(1+\gamma\mu_M)}{2q_M}, \min_{l\in\mathcal{L}} \left[\gamma\left(a^*_{l,\epsilon} + \frac{1}{2}\epsilon\right) - f_l^{-1}(d_l) \right] \right\}.$$

In addition, ***ALG*** can achieve

$$\sum_{l\in\mathcal{L}} f(a_l) \ge \sum_{l\in\mathcal{L}} f\left(a^*_{l,\epsilon} + \frac{1}{2}\epsilon\right) - \frac{B_4}{V}.$$

Proof: The proof is provided in Appendix D. ■

Remark 4: In [32], by defining the *mixing time* of a Markov chain as the time required for the Markov chain to get close to the stationary distribution, it has been shown that the mixing time corresponding to the parallel Glauber dynamics of the link rate scheduler is of order $O(\log N)$, where we recall N is the number of nodes in the network. Thus, we can expect the buffer size (and hence the average delay bound) to be of order $O(\log N)$ from (21) by choosing $T = O(\log N)$ in (20). This still presents a significant improvement over existing solutions using the periodic update approach. The average actual queue length under the algorithm proposed in [32] is of order $O(N^2 \log N)$ (from [32, Theorem 8]). Note that our results do not conflict with the findings in [37]. Specifically, in [37], it is shown that in open-loop systems, the network delay grows exponentially with the network size. However, in our work, a closed-loop system is considered, i.e., the rate of packet admission/generation into the network is determined by the scheduler.

VII. Implementation Study

To validate ***ALG***, the proposed algorithm for systems with single-hop transmissions, and constantly backlogged sources in Section IV, we compare ***ALG*** to an algorithm Q-CSMA, adistributed throughput-optimal algorithm proposed in [27], with link weights in the scheduler chosen as $w_l(t) = \log(U_l(t))$. This has been shown in [27] and [29] to achieve better delay performance than the throughput-optimal algorithms with link weights of the form $U_l(t)$ [25] and $\log\log(U_l(t))$ [28].

To guarantee fairness in the comparison, we construct the same congestion controller (8) for Q-CSMA and ***ALG***. It is easy to check that queue backlogs $U_l(t)$ are also bounded by q_M in our implementation of Q-CSMA. Therefore, q_M can be considered as the buffer size for both algorithms, and the packet delay is defined in the same way for both algorithms (i.e., the period from the time a packet gets admitted into the network to the time it gets served).

We implement the proposed algorithm and Q-CSMA in hardware on the Crossbow Telos-B platform as shown in Fig. 2(a). Each node is equipped with a CC2420 802.15.4 wireless transceiver [38] and a programmable MSP430 processor [39]. For

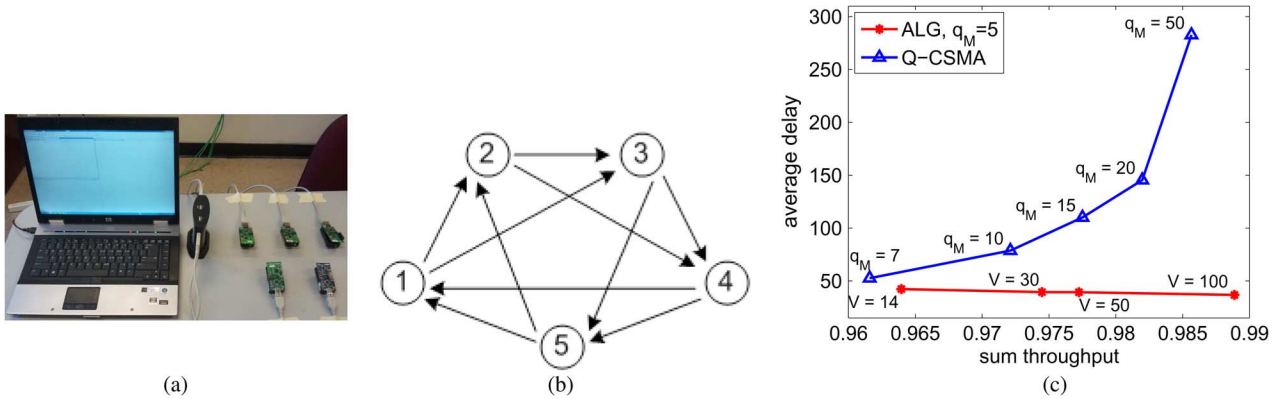


Fig. 2. Implementation results. (a) Hardware testbed setup. (b) Network topology for implementation. (c) Test results: delay versus throughput.

each protocol implementation, a time-slotted structure was used to coordinate the distributed link rate scheduler as described in Appendix A. To maintain the time-slotted structure, nodes periodically exchange timing information every N_{sync} time-slots to realign their internal clocks. The value of N_{sync} is dependent on the estimated clock drift of the nodes and the duration of the mini-slots and time-slots. For our implementation, resynchronizing the node clocks every 100 time-slots is sufficient. Time synchronization is initialized based on *a priori* knowledge of the network topology. If the network topology is not known, this information can be obtained via a neighbor discovery protocol [40].

We test both ***ALG*** and Q-CSMA on the topology shown in Fig. 2(b) with 5 nodes and 10 directional links. In this test, all nodes are within communication range of each other and therefore are in the same interference set, i.e., we employ the two-hop interference model. Under this model, at most one link can be activated in each time-slot, and hence the optimal throughput is 1. For simplicity, we choose the utility functions as $f_l(x) = x,\ \forall l \in \mathcal{L}$, i.e., we consider data rate constraints and throughput optimization instead of utility constraint and utility optimization. We choose the link rate weights as $w_l = (0.1/q_M)U_l(t)Q_l(t),\ \forall l \in \mathcal{L}$, as proposed in Section VI-A where $\alpha = 0.1$. We fix the number of mini-slots (introduced in Appendix A) as $T_s = \max_{l\in\mathcal{L}} |N(l)| + 1 = 11$ and let $v_M = \mu_M = 2$.

Fig. 2(c) shows the delay performance comparison between ***ALG*** and Q-CSMA as a function of network throughput, where packet delay is measured in unit of time-slots and throughput in unit of packets. We set the buffer size $q_M = 5$ in ***ALG*** since we observe that $q_M = 5$ is sufficient to achieve a near-optimal throughput. Each result represents the average delay and throughput over 10 000 time-slots. Note that in Fig. 2(c), to ensure a fair comparison to Q-CSMA, we do not impose minimum utility/throughput constraints to individual links for both ***ALG*** and Q-CSMA. By increasing V, the network throughput can be improved *without sacrificing delay performance* under ***ALG***. This might sound counterintuitive, but can be explained by the Little's Law. With an increasing V, the throughput increases, i.e., the actual packet queues are emptying faster and being serviced at an increased rate, while we observe that the actual queue backlogs are close to (and upper-bounded by) the finite buffer size ($q_M = 5$ in this experiment). Hence, the average delay can be slightly reduced when V increases. With Q-CSMA, buffer size q_M must be increased significantly to achieve a throughput similar to that of ***ALG***, which results in a significantly large average delays. Note that for small values of q_M, the average delay for Q-CSMA may be less than that of ***ALG***. However, by tuning V for ***ALG***, higher throughput can be achieved without sacrificing delay performance.

Note that we can increase V to further improve network performance of ***ALG***. However, very large values of V cause the virtual queue $(Q_l(t))$ to increase over the course of the 10 000 time-slot tests. A more detailed explanation of the effect of V on the system convergence time is given through simulation results in Section VIII.

VIII. Further Numerical Results

In this section, we further evaluate the performance of ***ALG*** for constantly backlogged sources in simulation.

A. Effect of Parameter V

In Sections VIII-A and VIII-B, for the same topology of Fig. 2(b), we employ an interference model different than that of Section VII, the well-known node-exclusive interference model in our simulation studies. Under this model, at most two links can be activated in each time-slot, and hence the optimal throughput is 2. As in Section VII, we adopt the utility function $f_l(x) = x$ and choose the link weights as $w_l = (0.1/q_M)U_l(t)Q_l(t),\ \forall l \in \mathcal{L}$. The results reflect averages obtained over 10^5 time-slots for each run. We fix the maximal data admission rate as $\mu_M = 2$ and let $v_M = 2$.

We first illustrate the algorithm performance in Fig. 3 by varying the parameter V, which we recall is a control parameter in the $R_l(t)$ regulator (7) proposed for the backlogged source system. In the simulation, we fix the buffer size as $q_M = 5$ and the utility constraint (equivalent to data rate constraint) as $d_l = 0.1,\ \forall l \in \mathcal{L}$. By increasing V, the proposed ***ALG*** approaches the optimal throughput as stated in Remark 2, while the minimum data rate requirements are satisfied and the delay remains almost unaffected.

However, a large value of V has a negative effect on the convergence rates of virtual queues. Note that with a large V, the term $-V f_l(R_l(t))$ will dominate the $R_l(t)$ regulator (7) when

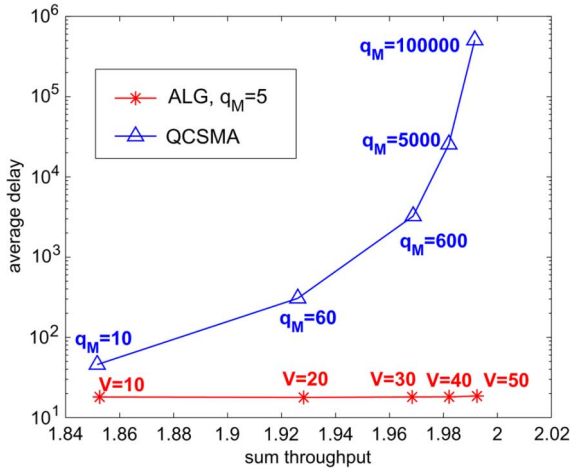


Fig. 3. Simulation comparison for constantly backlogged sources: delay versus throughput.

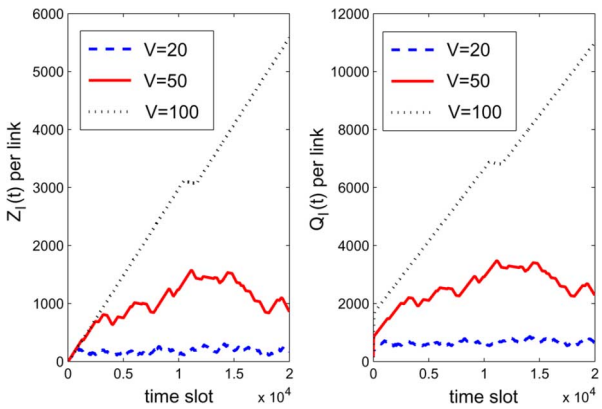


Fig. 4. Virtual queue evolutions.

the virtual queue $Q_l(t)$ is comparatively small, and hence the convergence rate of the system will be very slow: $R_l(t)$ will be set as $R_l(t) = v_M$ for $t = 1, 2, \ldots$ until $Q_l(t)$ grows large enough such that $R_l(t)(((q_M - \mu_M)/q_M)Q_l(t) - Z_l(t))$ is approximately in the same order as $V f_l(R_l(t))$. The virtual queue evolutions over time are shown in Fig. 4. We observe that a larger V indeed indicates a slower convergence rate of virtual queues. For $V = 100$, the per-link average of virtual queue backlog $Q_l(t)$ and $Z_l(t)$ constantly grow over the observed simulation duration, and the data rate constraints cannot be met within the simulation time of 10^5 time-slots, although a convergence would be achieved as $t \to \infty$. Nevertheless, we observe in Fig. 3 that $V = 20$ and $V = 40$ are sufficient to achieve more than 95% and 99% of the optimal throughput of the network, respectively.

B. Comparative Performance Evaluation

In this section, based on the simulation setup in Section VIII-A, we compare ***ALG*** to two other algorithms: 1) a distributed throughput-optimal algorithm [29], with link weights in the scheduler chosen as $w_l(t) = (\log(U_l(t))/\log(e + \log(1 + U_l(t))))$, which has been shown in simulation to achieve better delay performance than the throughput-optimal algorithms in [25], [28]. Since it is observed in [29] that this algorithm is similar in throughput and delay performance with the Q-CSMA [27], we denote it by *QCSMA*; 2) atraditional CSMA algorithm which employs RTS/CTS handshake to reserve the channel. In the following, we denote this algorithm as *CSMA*.

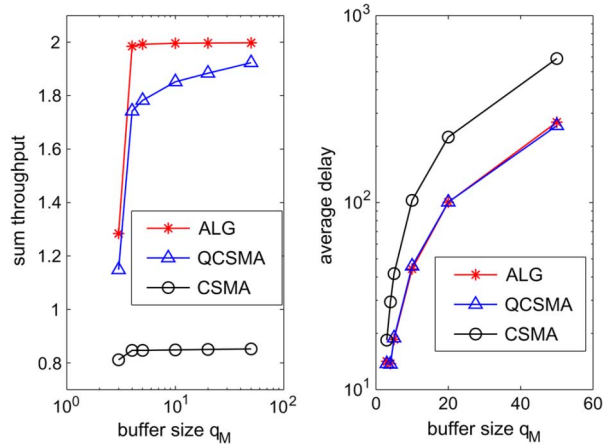


Fig. 5. Algorithm comparison to different buffer sizes.

Note that for both QCSMA and CSMA, we employ the same congestion controller (8) in ***ALG*** with buffer size denoted as q_M for fairness in the comparison.

Fig. 5 shows the throughput and delay performances by varying the buffer size q_M and assuming backlogged sources, where we measure packet delay in time-slots and throughput in packets per time-slot. In the simulation, we choose $V = 50$ and $d_l = 0.1$, $\forall l \in \mathcal{L}$, for ***ALG***. By increasing q_M, i.e., by allowing more packets into the queues, the throughput increases closer to the optimality with a tradeoff of delay performance. For a given q_M value, ***ALG*** and QCSMA have similar delay performances, but the throughput of ***ALG*** significantly outperforms that of QCSMA. As an illustration, with $q_M = 5$, ***ALG*** can achieve a throughput of 1.9924, which is 11.8% more than the throughput of 1.7825 for QCSMA. The remaining unused channel opportunity (the optimality minus the throughput) of QCSMA is 28.6 times that of ***ALG***. In addition, each link can meet the minimum data rate requirement of 0.1 under ***ALG***.

We observe in Fig. 5 that the CSMA algorithm has the worst delay performance since its scheduler does not utilize any queue backlog information, an indicator of congestion level of links. CSMA also has the worst throughput performance since links are not scheduled for transmission in CSMA when their RTS/CTS handshakes collide, which is in sharp contrast with Proposition 2, where we show that the scheduler of ***ALG*** can approach a maximal weight matching with high probability.

In Fig. 3, we compare the delay performances of ***ALG*** and QCSMA as a function of throughput. In the simulation, we set $q_M = 5$ for ***ALG***, and we increase the simulation time to 10^7 to obtain reliable results for QCSMA with $q_M = 100\,000$. We observe that QCSMA requires a much larger buffer size (and hence a significantly larger average packet delay) to achieve comparable throughput levels as ***ALG***. ***ALG*** can operate at higher throughput rates by only adjusting the V parameter, which does not affect the delay performance. Note that a larger value of V in ***ALG*** only affects the convergence rate of virtual queues, and

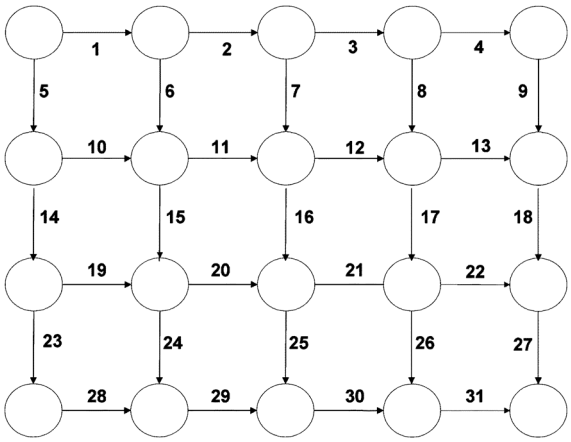


Fig. 6. Grid network topology for simulation comparison.

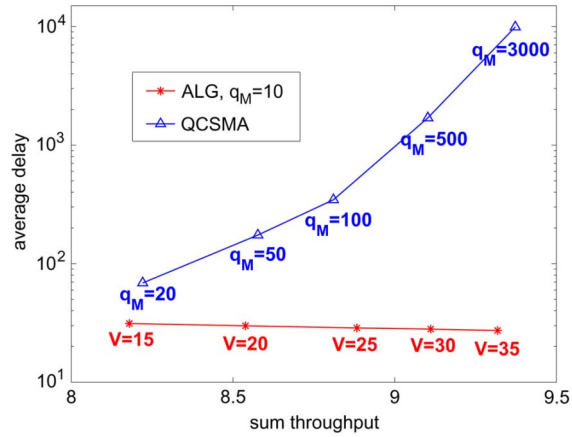


Fig. 7. Simulation comparison for constantly backlogged sources: delay versus throughput in grid topology.

we have shown in Fig. 4 that $V \leq 50$ is a feasible choice for the considered system. Consequently, we have shown that ***ALG*** can achieve close-to-optimal throughput with significantly more favorable delay with respect to QCSMA.

C. Simulation Comparison in Grid Topology

As pointed out in [33], CSMA algorithms such as QCSMA suffer from "temporal starvation" problem,[3] which can lead to large delays in network topologies such as grid and torus ones. In this section, we consider a grid topology shown in Fig. 6 with node-exclusive interference model. We will show that ***ALG*** does not suffer from large delays as much as QCSMA in the grid topology setting. We choose the link weights as $w_l = (0.1/q_M)U_l(t)Q_l(t)$ and fix $v_M = \mu_M = 2$. We consider throughput as the utility metric, i.e., $f_l(x) = x$, $\forall l \in \mathcal{L}$. With the link notations in Fig. 6, we set the minimum throughput constraints as $d_l = 0.2$ for $l = 1, 3, 28, 30$; $d_l = 0$, for $l = 6, 7, 8, 24, 25, 26$; and $d_l = 0.1$ otherwise.

In Fig. 7, we observe that ***ALG*** results in a far better delay performance than QCSMA to achieve comparable throughput levels. Similar to the results discussed in Section VIII-B, a larger value of V improves the throughput performance while not increasing the delay. In addition, the minimum throughput constraints are met for all individual links. As a conclusion, we show through numerical results that ***ALG*** employing virtual queues can reduce delays in grid topologies.

IX. Conclusion and Future Work

In this paper, we proposed a cross-layer utility-optimal scheduling algorithm with finite buffers that guarantees individual link-based/flow-based minimum utility constraints. The finite buffer size of packet queues implies upper-bounded average packet delay in the network and is shown to be of order $O(1/\epsilon)$, where ϵ characterizes the distance between the achieved utility and the optimal value. The distributed algorithm can be implemented via CSMA/CA methods and is shown to achieve close to optimal throughput and good delay performance both in simulation and numerical results. In our future work, we will implement the proposed multihop extension of the algorithm and analyze its performance and scalability.

[3]It may take a long time to switch from one maximal weight schedule to another.

Appendix A
Distributed Link Rate Scheduler

In the following, we present a distributed implementation of the link rate scheduler. This distributed scheduler is a variation of the distributed scheduler in [27], which is based on the RTS/CTS mechanism in the 802.11 standard and is proposed for single-hop wireless networks. Note that the RTS/CTS handshake is only a tool to implement the distributed local interaction of the algorithm, and other alternatives may exist. We assume each node has a single transceiver, i.e., a node cannot transmit and receive at the same time. Let the set of links with the same source node $n \in \mathcal{N}$ be $I(n) = \{(n,i) : (n,i) \in \mathcal{L}\}$, and let $\mu_l(0) = 0$, $\forall l \in \mathcal{L}$.

The main difficulty in the distributed implementation is to randomly select an independent link set satisfying condition (9). To achieve this, we employ the RTS/CTS mechanism. Specifically, in each time-slot, we assign a number of T_s control mini-slots before data transmissions. The distributed link scheduler is operated at each node $n \in \mathcal{N}$ for each time-slot t, as illustrated in Fig. 8. At the beginning of each time-slot, the node n selects a link (n, r_n) from $I(n)$ uniformly at random and chooses a random backoff time from the $(T_s - 1)$ mini-slots (Step 1 in Fig. 8). Then, under some conditions specified in Step 2 in Fig. 8, node n initiates a *Request-To-Send–Clear-To-Send* (RTS/CTS) handshake with its destination node r_n when the backoff duration is over. To facilitate the RTS/CTS handshake, each mini-slot is further split into two micro-slots (Steps 2.1 and 2.2 in Fig. 8), dedicated for CTS and RTS transmissions, respectively. After the RTS/CTS handshakes, the distributed link rate scheduler assigns link rates (Step 3 in Fig. 8) following the same assignment in the scheduler with Glauber dynamics described in Section IV.

For narrative clarity, we let $\mathrm{RTS}(n)$ and $\mathrm{CTS}(n)$ denote, respectively, the RTS intended for node n and the CTS intended for node n, with $n \in \mathcal{N}$.

Distributed Link Rate Scheduler at node n in time slot t
Step 1. Initialization:
A node r_n is selected in $I(n)$ uniformly at random;
n chooses a backoff time b_n uniformly at random from the first (T_s-1) mini-slots.
Step 2. RTS-CTS handshake:
for mini-slot $t_s = 1 : T_s$ (i.e., t_s is the current mini-slot)
  Step 2.1. The first micro-slot: CTS transmission
  if n has not overheard in the past any RTS(m), $\forall m \neq n$
    and n has not overheard any collision of RTS before
    and n has not sent out a CTS before
    and n has not received CTS(n) before
    and n received only one RTS(n) in mini-slot (t_s-1)
    do: node n sends out a CTS responding to the RTS(n) received in mini-slot (t_s-1);
  end if
  Step 2.2. The second micro-slot: RTS transmission
  if n has not overheard in the past any CTS(m), $\forall m \neq n$
    and n has not overheard any collision of CTS before
    and n has not sent a CTS before **and** $b_n = t_s$
    do: node n sends out an RTS(r_n) to node r_n;
  end if
end for
Step 3. Schedule for links $\{(n,i):(n,i)\in\mathcal{L}\}$:
$\mu_{ni}(t) = \mu_{ni}(t-1)$, $\forall i \neq r_n : (n,i)\in\mathcal{L}$.
if n received CTS(n) from r_n in Step 2 **do:**
  if $\sum_{j\in\mathcal{N}((n,r_n))\setminus\{(n,r_n)\}}\mu_j(t-1) > 0$
    do: $\mu_{nr_n}(t) = 0$;
  else
  do: $\mu_{nr_n}(t) = 1$ w.p. $p_{(n,r_n)}$ defined in Proposition 2;
    $\mu_{nr_n}(t) = 0$ w.p. $(1-p_{(n,r_n)})$;
  end if
else (i.e., (n,r_n) failed the RTS-CTS handshake)
  do: $\mu_{nr_n}(t) = \mu_{nr_n}(t-1)$;
end if

Fig. 8. Distributed link rate scheduler for networks with single-hop transmissions.

It is not difficult to check that under the distributed link rate scheduler, any link set $\mathbf{x}(t)$ whose links succeed in the RTS-CTS handshake (i.e., Step 2 in Fig. 8) is an independent link set and $\cup_{P(\mathbf{x}(t))>0}\mathbf{x}(t) = \mathcal{L}$, satisfying (9). Thus, the distributed link rate scheduler is equivalent to the scheduler in Section IV.

Appendix B
Proof of Theorem 1

Before we proceed, we present the following lemma that will assist us in proving Theorem 1.

Lemma 1: For any feasible rate vector $(\theta_l)_{l\in\mathcal{L}} \in \Lambda$, there exists a stationary randomized algorithm STAT that stabilizes the network with input rate vector $(A_l^{\mathrm{STAT}}(t))$ and scheduling parameters $(\mu_l^{\mathrm{STAT}}(t))$ independent of queue backlogs, such that the admitted rates are

$$A_l^{\mathrm{STAT}}(t) = \mathbb{E}\left\{\mu_l^{\mathrm{STAT}}(t)\right\} = \theta_l \qquad \forall t, \forall l \in \mathcal{L}.$$

Note that it is not necessary for the randomized algorithm STAT to be distributed or finite-buffered. Similar formulations of STAT and their proofs have been given in [6], [34], and [35], so we omit the proof of Lemma 1 for brevity.

Remark 5: According to the STAT algorithm in Lemma 1, we assign the input rates of the virtual queues $Q_l(t)$ as $R_l^{\mathrm{STAT}}(t) = \theta_l$. Note that (θ_l) can take values such as $(a^*_{l,\epsilon} + (1/2)\epsilon)$ or $(a^*_{l,\epsilon} + \epsilon)$.

To prove Theorem 1, we first define a network state of $\mathbf{Q}_1(t) = ((U_l(t)),(Q_l(t)),(Z_l(t)))$ and then define the Lyapunov function $L_1(\mathbf{Q}_1(t))$ as follows:

$$L_1\left(\mathbf{Q}_1(t)\right) \triangleq \frac{1}{2}\sum_{l\in\mathcal{L}}\left\{\frac{q_M-\mu_M}{q_M}Q_l(t)^2 + \frac{1}{q_M}U_l(t)^2Q_l(t) + Z_l(t)^2\right\}$$

where we note that different from the traditional quadratic Lyapunov function, there is the (weighted) virtual queue backlog $Q_l(t)/q_M$ multiplied to the quadratic term of actual queue backlogs. We denote the Lyapunov drift by

$$\Delta(t) \triangleq \mathbb{E}\left\{L_1\left(\mathbf{Q}_1(t+1)\right) - L_1\left(\mathbf{Q}_1(t)\right)|\mathbf{Q}_1(t)\right\}.$$

By squaring both sides of the queue dynamics (3), (4), and (6) and through simple algebra, we obtain

$$\begin{aligned}
&\Delta(t) - V\sum_{l\in\mathcal{L}}\mathbb{E}\left\{f_l\left(R_l(t)\right)|\mathbf{Q}_1(t)\right\}\\
&\le B_1 + \frac{\mu_M^2+1}{2q_M}\sum_{l\in\mathcal{L}}Q_l(t) - V\sum_{l\in\mathcal{L}}\mathbb{E}\left\{f\left(R_l(t)\right)|\mathbf{Q}_1(t)\right\}\\
&\quad - \mathbb{E}\left\{\sum_{l\in\mathcal{L}}\frac{1}{q_M}Q_l(t)U_l(t)\left(\mu_l(t)-A_l(t)\right)|\mathbf{Q}_1(t)\right\}\\
&\quad - \mathbb{E}\left\{\sum_{l\in\mathcal{L}}\frac{q_M-\mu_M}{q_M}Q_l(t)\left(A_l(t)-R_l(t)\right)|\mathbf{Q}_1(t)\right\}\\
&\quad - \mathbb{E}\left\{\sum_{l\in\mathcal{L}}Z_l(t)\left(R_l(t)-f_l^{-1}(d_l)\right)|\mathbf{Q}_1(t)\right\}
\end{aligned}$$

the equivalence of which is given as follows:

$$\begin{aligned}
&\Delta(t) - V\sum_{l\in\mathcal{L}}\mathbb{E}\left\{f_l\left(R_l(t)\right)|\mathbf{Q}_1(t)\right\}\\
&\le B_1 + \frac{\mu_M^2+1}{2q_M}\sum_{l\in\mathcal{L}}Q_l(t) + \sum_{l\in\mathcal{L}}f_l^{-1}(d_l)Z_l(t)\\
&\quad + \sum_{l\in\mathcal{L}}\mathbb{E}\left\{g_l\left(R_l(t)\right)|\mathbf{Q}_1(t)\right\}\\
&\quad - \sum_{l\in\mathcal{L}}\frac{1}{q_M}U_l(t)Q_l(t)\mathbb{E}\left\{\mu_l(t)|\mathbf{Q}_1(t)\right\}\\
&\quad - \sum_{l\in\mathcal{L}}\frac{Q_l(t)}{q_M}\mathbb{E}\left\{A_l(t)|\mathbf{Q}_1(t)\right\}\left(q_M-\mu_M-U_l(t)\right).
\end{aligned} \tag{22}$$

Denoting the right-hand side (RHS) of (22) by $RHS(t)$, and taking the expectation of both sides of (22) with respect to the distribution of $\mathbf{Q}_1(t)$, we have

$$\begin{aligned}
&\mathbb{E}\left\{\Delta(t)\right\} - V\sum_{l\in\mathcal{L}}\mathbb{E}\left\{f_l\left(R_l(t)\right)\right\}\\
&\le P\left(\|\mathbf{q}(t)\| > B\right)\mathbb{E}_{\{\mathbf{Q}_1(t)|\|\mathbf{q}(t)\|>B\}}\left\{RHS(t)\right\}\\
&\quad + P\left(\|\mathbf{q}(t)\| \le B\right)\mathbb{E}_{\{\mathbf{Q}_1(t)|\|\mathbf{q}(t)\|\le B\}}\left\{RHS(t)\right\}.
\end{aligned} \tag{23}$$

Note that from the $R_l(t)$ regulator and the congestion controller in ***ALG***, the fourth and the last terms of $RHS(t)$ are nonpositive. By employing Proposition 2, the $RHS(t)$ inside the first term of the RHS of (23) satisfies

$$\begin{aligned} &RHS(t) \\ &\le B_1 + \frac{\mu_M^2+1}{2q_M}\sum_{l\in\mathcal{L}} Q_l(t) + \sum_{l\in\mathcal{L}} f_l^{-1}(d_l)Z_l(t) \\ &\quad + \gamma\sum_{l\in\mathcal{L}} \mathbb{E}\left\{g_l\left(R_l(t)\right)|\mathbf{Q}_1(t)\right\} - \gamma\mathbb{E}\left\{w^*(t)|\mathbf{Q}_1(t)\right\} \\ &\quad - \gamma\sum_{l\in\mathcal{L}} \frac{Q_l(t)}{q_M}\mathbb{E}\left\{A_l(t)|\mathbf{Q}_1(t)\right\}\left(q_M-\mu_M-U_l(t)\right). \end{aligned} \tag{24}$$

Similarly, the $RHS(t)$ inside the second term of the RHS of (23) satisfies

$$\begin{aligned} &RHS(t) \\ &\le B_1 + \frac{\mu_M^2+1}{2q_M}\sum_{l\in\mathcal{L}} Q_l(t) + \sum_{l\in\mathcal{L}} f_l^{-1}(d_l)Z_l(t) \\ &\quad + \gamma\sum_{l\in\mathcal{L}} \mathbb{E}\left\{g_l\left(R_l(t)\right)|\mathbf{Q}_1(t)\right\} \\ &\quad - \gamma\sum_{l\in\mathcal{L}} \frac{Q_l(t)}{q_M}\mathbb{E}\left\{A_l(t)|\mathbf{Q}_1(t)\right\}\left(q_M-\mu_M-U_l(t)\right). \end{aligned} \tag{25}$$

The fourth term and the last term of the RHS of (24) and (25) are minimized by the $R_l(t)$ regulator (7) and the congestion controller (8), respectively, over a set of feasible algorithms including the stationary randomized algorithm STAT introduced in Lemma 1. We can substitute into the fourth term of the RHS of (24) and (25) a stationary randomized algorithm STAT with admitted arrival rate vector $(a^*_{l,\epsilon}+(1/2)\epsilon)$ and into the last term with an STAT with admitted arrival rate vector $(a^*_{l,\epsilon}+\epsilon)$. In addition, since $-w^*(t)$ minimizes $-\sum_{l\in\mathcal{L}} U_l(t)Q_l(t)\mu_l(t)$ over all feasible schedulers, we can substitute into the fifth term of RHS of (24) an STAT with admitted arrival rate vector $(a^*_{l,\epsilon}+\epsilon)$. By the above substitutions and rearranging terms, we obtain from (23)

$$\begin{aligned} &\mathbb{E}\left\{\Delta(t)\right\} - V\sum_{l\in\mathcal{L}}\mathbb{E}\left\{f_l\left(R_l(t)\right)\right\} \\ &\le B_1 - \gamma V\sum_{l\in\mathcal{L}} f_l\left(a^*_{l,\epsilon}+\frac{1}{2}\epsilon\right) \\ &\quad - \sum_{l\in\mathcal{L}}\left[\gamma(a^*_{l,\epsilon}+\frac{1}{2}\epsilon) - f^{-1}(d_l)\right]\mathbb{E}\left\{Z_l(t)\right\} \\ &\quad - \left(\frac{(q_M-\mu_M)\gamma\epsilon}{2q_M} - \frac{\mu_M^2+1}{2q_M}\right)\mathbb{E}\left\{\sum_{l\in\mathcal{L}} Q_l(t)\right\} \\ &\quad + P\left(\|\mathbf{q}(t)\|\le B\right) \\ &\quad \times \mathbb{E}\left\{\frac{\gamma}{q_M}\sum_{l\in\mathcal{L}} U_l(t)Q_l(t)\left(a^*_{l,\epsilon}+\epsilon\right)\big|\|\mathbf{q}(t)\|\le B\right\} \end{aligned} \tag{26}$$

which leads to

$$\begin{aligned} &\mathbb{E}\left\{\Delta(t)\right\} - V\sum_{l\in\mathcal{L}}\mathbb{E}\left\{f_l\left(R_l(t)\right)\right\} \\ &\le B_2 - \delta_1\sum_{l\in\mathcal{L}}\mathbb{E}\left\{Q_l(t)+Z_l(t)\right\} - \gamma V\sum_{l\in\mathcal{L}} f_l(a^*_{l,\epsilon}+\frac{1}{2}\epsilon). \end{aligned} \tag{27}$$

We take the time average on $\tau = 0,\ldots,t-1$ of both sides of (27), and take the limsup with respect to t to prove (13). Similarly, we can prove (14) given that the virtual queues $Q_l(t)$ are stable.

Appendix C
Proof of Corollary 2

Let the buffer sizes be chosen as

$$q_l = \frac{\mu_M^2+1}{2\epsilon_l} + \mu_M + \eta f^{-1}(d_l) \qquad \forall l\in\mathcal{L}. \tag{28}$$

To satisfy the minimum utility constraints, from (17) in Theorem 2, we should have

$$\begin{aligned} &\frac{\mu_M^2+1}{2\epsilon_l} + \mu_M + \eta f^{-1}(d_l) \\ &\qquad > \frac{\mu_M^2+1}{2\left[(\gamma-1)f_l^{-1}(d_l)+\gamma\epsilon_l\right]} + \mu_M \qquad \forall l\in\mathcal{L} \end{aligned}$$

which leads to

$$\gamma > \max_{l\in\mathcal{L}} \frac{f_l^{-1}(d_l) + \frac{\left(\mu_M^2+1\right)\epsilon_l}{2\eta f^{-1}(d_l)\epsilon_l+\mu_M^2+1}}{f_l^{-1}(d_l)+\epsilon_l}. \tag{29}$$

Since the RHS of (29) is smaller than 1, as long as γ satisfies (29), the choices of buffer sizes (28) are feasible. Then, the delay upper bounds (19) in Corollary 2 immediately follows from Little's Law.

Appendix D
Proof of Theorem 3

Given time-slot $t = kT+i$, where $0\le i<T$, by analyzing the queue dynamics (3) and (4), we have

$$\begin{aligned} U_l(t)Q_l(t) &\le \left(U_l(kT)+i\mu_M\right)Q_l(t) \\ &\le U_l(kT)Q_l(kT) + i\mu_M q_M + i\mu_M Q_l(t). \end{aligned} \tag{30}$$

Similarly

$$U_l(kT)Q_l(kT) \le U_l(t)Q_l(t) + i\mu_M q_M + iQ_l(t). \tag{31}$$

By employing the inequalities (30) and (31), following the steps in deriving (26) in the proof of Theorem 1, we can obtain

$$\begin{aligned} &\mathbb{E}\left\{\Delta(t)\right\} - V\sum_{l\in\mathcal{L}}\mathbb{E}\left\{f_l\left(R_l(t)\right)\right\} \\ &\le B_2 + (1+\gamma)\mu_M L(T-1) - \gamma V\sum_{l\in\mathcal{L}} f_l\left(a^*_{l,\epsilon}+\frac{1}{2}\epsilon\right) \\ &\quad - \sum_{l\in\mathcal{L}}\left[\gamma\left(a^*_{l,\epsilon}+\frac{1}{2}\epsilon\right) - f^{-1}(d_l)\right]\mathbb{E}\left\{Z_l(t)\right\} \\ &\quad - \mathbb{E}\left\{\sum_{l\in\mathcal{L}} Q_l(t)\left(\frac{\gamma\epsilon(q_M-\mu_M)}{2q_M}\right.\right. \\ &\qquad \left.\left. - \frac{\mu_M^2+1+2(T-1)(1+\gamma\mu_M)}{2q_M}\right)\right\} \end{aligned}$$

which leads to

$$\mathbb{E}\left\{\Delta(t)\right\} - V\sum_{l\in\mathcal{L}}\mathbb{E}\left\{f_l\left(R_l(t)\right)\right\}$$
$$\leq B_4 - \delta_3\sum_{l\in\mathcal{L}}\mathbb{E}\left\{Q_l(t)+Z_l(t)\right\} - \gamma V\sum_{l\in\mathcal{L}} f_l\left(a^*_{l,\epsilon}+\frac{1}{2}\epsilon\right).$$

Following the steps in deriving Theorem 1, we can prove Theorem 3.

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. Autom. Control*, vol. 37, no. 12, pp. 1936–1948, Dec. 1992.

[2] A. Eryilmaz, R. Srikant, and J. Perkins, "Stable scheduling policies for fading wireless channels," *IEEE/ACM Trans. Netw.*, vol. 13, no. 2, pp. 411–424, Apr. 2005.

[3] M. Lotfinezhad, B. Liang, and E. Sousa, "On stability region and delay performance of linear-memory randomized scheduling for time-varying networks," *IEEE/ACM Trans. Netw.*, vol. 17, no. 6, pp. 1860–1873, Dec. 2009.

[4] J. Ryu, L. Ying, and S. Shakkottai, "Back-pressure routing for intermittently connected networks," in *Proc. IEEE INFOCOM Mini-Conf.*, March 2010, pp. 1–5.

[5] M. Neely and E. Modiano, "Dynamic power allocation and routing for time varying wireless networks," *IEEE J. Sel. Areas Commun.*, vol. 23, no. 1, pp. 89–103, Mar. 2005.

[6] L. Georgiadis, M. Neely, and L. Tassiulas, "Resource allocation and cross-Layer control in wireless networks," *Found. Trends Netw.*, vol. 1, no. 1, pp. 1–149, 2006.

[7] X. Liu, E. Chong, and N. Shroff, "Opportunistic transmission scheduling with resource-sharing constraints in wireless networks," *IEEE J. Sel. Areas Commun.*, vol. 19, no. 10, pp. 2053–2065, Oct. 2001.

[8] M. Neely, "Energy optimal control for time varying wireless networks," *IEEE Trans. Inf. Theory*, vol. 52, no. 7, pp. 2915–2934, Jul. 2006.

[9] A. Eryilmaz and R. Srikant, "Joint congestion control, routing and MAC for stability and fairness in wireless networks," *IEEE J. Sel. Areas Commun.*, vol. 24, no. 8, pp. 1514–1524, Aug. 2006.

[10] V. Venkataramanan, X. Lin, L. Ying, and S. Shakkottai, "On scheduling for minimizing end-to-end buffer usage over multihop wireless networks," in *Proc. IEEE INFOCOM*, Mar. 2010, pp. 1–9.

[11] L. Bui, R. Srikant, and A. Stolyar, "Novel architectures and algorithms for delay reduction in back-pressure scheduling and routing," in *Proc. IEEE INFOCOM Mini-Conf.*, Apr. 2009, pp. 2936–2940.

[12] L. Ying, S. Shakkottai, and A. Reddy, "On combining shortest-path and back-pressure routing over multihop wireless networks," in *Proc. IEEE INFOCOM*, Apr. 2009, pp. 1674–1682.

[13] H. Xiong, R. Li, A. Eryilmaz, and E. Ekici, "Delay-aware cross-layer design for network utility maximization in multi-hop networks," *IEEE J. Sel. Areas Commun.*, vol. 29, no. 5, pp. 951–959, May 2011.

[14] G. Sharma, R. Mazumdar, and N. Shroff, "On the complexity of scheduling in wireless networks," in *Proc. 12th Annu. MobiCom*, 2006, pp. 227–238.

[15] X. Wu, R. Srikant, and J. Perkins, "Scheduling efficiency of distributed greedy scheduling algorithms in wireless networks," *IEEE Trans. Mobile Comput.*, vol. 6, no. 6, pp. 595–605, Jun. 2007.

[16] P. Chaporkar, K. Kar, and S. Sarkar, "Throughput guarantees through maximal scheduling in wireless networks," in *Proc. 43rd Annu. Allerton Conf. Commun., Control, Comput.*, Sep. 2005.

[17] X. Lin and S. Rasool, "A distributed joint channel-assignment, scheduling and routing algorithm for multi-channel ad-hoc wireless networks," in *Proc. IEEE INFOCOM*, May 2007, pp. 1118–1126.

[18] P. Huang, X. Lin, and C. Wang, "A low-complexity congestion control and scheduling algorithm for multihop wireless networks with order-optimal per-flow delay," in *Proc. IEEE INFOCOM*, Apr. 2011, pp. 2588–2596.

[19] X. Lin and N. Shroff, "The impact of imperfect scheduling on cross-layer congestion control in wireless networks," *IEEE/ACM Trans. Netw.*, vol. 14, no. 2, pp. 302–315, Apr. 2006.

[20] P. Giaccone, E. Leonardi, and D. Shah, "Throughtput region of finite-buffered networks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 18, no. 2, pp. 251–263, Feb. 2007.

[21] L. Le, E. Modiano, and N. Shroff, "Optimal control of wireless networks with finite buffers," in *Proc. IEEE INFOCOM*, Mar. 2010, pp. 1–9.

[22] D. Xue and E. Ekici, "Delay-guaranteed cross-layer scheduling in multihop wireless networks," *IEEE/ACM Trans. Netw.*, vol. 21, no. 6, pp. 1696–1707, Dec. 2013.

[23] M. Neely, "Delay-based network utility maximization," in *Proc. IEEE INFOCOM*, Mar. 2010, pp. 1–9.

[24] M. Neely, "Opportunistic scheduling with worst case delay guarantees in single and multi-hop networks," in *Proc. IEEE INFOCOM*, Apr. 2011, pp. 1728–1736.

[25] L. Jiang and J. Walrand, "a distributed algorithm for throughput and utility maximization in wireless networks," *IEEE/ACM Trans. Netw.*, vol. 18, no. 3, pp. 960–972, Jun. 2010.

[26] L. Jiang, D. Shah, J. Shin, and J. Walrand, "Distributed random access algorithm: scheduling and congestion control," *IEEE Trans. Inf. Theory*, vol. 56, no. 12, pp. 6182–6207, Dec. 2010.

[27] J. Ni, B. Tan, and R. Srikant, "Q-CSMA queue-length based CSMA/CA algorithms for achieving maximum throughput and low delay in wireless networks," in *Proc. IEEE INFOCOM Mini-Conf.*, Apr. 2010, pp. 1–5.

[28] S. Rajagopalan, D. Shah, and J. Shin, "Network adiabatic theorem: an efficient randomized protocol for contention resolution," in *Proc. 11th SIGMETRICS*, 2009, pp. 133–144.

[29] J. Ghaderi and R. Srikant, "On the design of efficient CSMA algorithms for wireless networks," in *Proc. IEEE CDC*, Dec. 2010, pp. 954–959.

[30] A. Proutiere, Y. Yi, T. Lan, and M. Chiang, "Resource allocation over network dynamics without timescale separation," in *Proc. IEEE INFOCOM Mini-Conf.*, Mar. 2010, pp. 1–5.

[31] L. Jiang and J. Walrand, "Convergence and stability of a distributed CSMA algorithm for maximal network throughput," in *Proc. IEEE CDC*, Dec. 2009, pp. 4840–4845.

[32] L. Jiang, M. Leconte, J. Ni, R. Srikant, and J. Walrand, "Fast mixing of parallel Glauber dynamics and low-delay CSMA scheduling," in *Proc. IEEE INFOCOM Mini-Conf.*, Apr. 2011, pp. 371–375.

[33] M. Lotfinezhad and P. Marbach, "Throughput-optimal random access with order-optimal delay," in *Proc. IEEE INFOCOM*, Apr. 2011, pp. 2867–2875.

[34] M. Neely, "Dynamic power allocation and routing for satellite and wireless networks with time varing channels," Ph.D. dissertation, Massachusetts Institution of Technology (MIT), Cambridge, MA, USA, 2003.

[35] M. Lotfinezhad, B. Liang, and E. Sousa, "Optimal control of constrained cognitive radio networks with dynamic population size," in *Proc IEEE INFOCOM*, Mar. 2010, pp. 1–9.

[36] A. Eryilmaz, R. Srikant, and J. Perkins, "Stable scheduling policies for fading wireless channels," *IEEE/ACM Trans. Netw.*, vol. 13, no. 2, pp. 411–424, Apr. 2005.

[37] D. Shah, D. Tse, and J. Tsitsiklis, "Hardness of low delay network scheduling," *IEEE Trans. Inf. Theory*, vol. 57, no. 12, pp. 7810–7817, Dec. 2011.

[38] Texas Instruments, Austin, TX, USA, "CC2420," 2013 [Online]. Available: http://focus.ti.com/docs/prod/folders/print/cc2420.html

[39] Texas Instruments, Austin, TX, USA, "MSP430 microcontroller," 2013 [Online]. Available: http://focus.ti.com/docs/prod/folders/print/msp430f1611.html

[40] E. Felemban, R. Murawski, E. Ekici, S. Park, K. Lee, J. Park, and Z. Hameed, "SAND: Sectored-antenna neighbor discovery protocol for wireless networks," in *Proc. IEEE SECON*, Jun. 2010, pp. 1–9.

[41] D. Xue and E. Ekici, "Distributed utility-optimal scheduling with finite buffers," in *Proc. WiOpt*, May 2012, pp. 278–285.

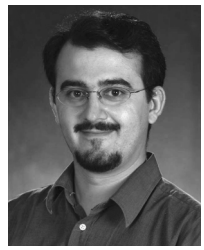
[42] D. Xue and E. Ekici, "Distributed utility-optimal scheduling with finite buffers," The Ohio State University, Columbus, OH, USA, Tech. Rep., 2013 [Online]. Available: http://www.ece.osu.edu/~xued/distributed.pdf

Dongyue Xue (S'11) received the B.S. degree in information engineering from Shanghai Jiaotong University, Shanghai, China, in 2009, and the M.S. and Ph.D. degrees in electrical and computer engineering from The Ohio State University, Columbus, OH, USA, in 2013.

His research interests include cross-layer scheduling in wireless networks and dynamic resource allocation in cognitive radio networks.

Robert Murawski (S'07–M'11) received the B.S. and M.S. degrees from Cleveland State University, Cleveland, OH, USA, in 2003 and 2004, respectively, and the Ph.D. degree in electrical engineering from The Ohio State University, Columbus, OH, USA, in 2011.

He is currently working with Vantage Partners, LLC, as a contractor for the NASA Glenn Research Center, Cleveland, OH, USA. He is currently working under the Space Communication and Navigation (SCaN) program at NASA, with a focus on space communications and the integration of the three NASA space communication networks.



Eylem Ekici (S'99–M'02–SM'11) received the B.S. and M.S. degrees in computer engineering from Bogazici University, Istanbul, Turkey, in 1997 and 1998, respectively, and the Ph.D. degree in electrical and computer engineering from the Georgia Institute of Technology, Atlanta, GA, USA, in 2002.

Currently, he is an Associate Professor with the Department of Electrical and Computer Engineering, The Ohio State University, Columbus, OH, USA. His current research interests include wireless networks, vehicular communication systems, cognitive radio networks, nano communication systems, with an emphasis on resource management, and analysis of network architectures and protocols.

Dr. Ekici is an Associate Editor of the IEEE/ACM TRANSACTIONS ON NETWORKING, IEEE TRANSACTIONS ON MOBILE COMPUTING, *Computer Networks*, and *ACM Mobile Computing and Communications Review*.