Distributed Utility-Optimal Scheduling with Finite Buffers

Dongyue Xue, Robert Murawski, Eylem Ekici

Distributed Utility-Optimal Scheduling with Finite Buffers

Dongyue Xue, Robert Murawski, Eylem Ekici

Department of Electrical and Computer Engineering
Ohio State University, USA
Email: {xued, murawskr, ekici}@ece.osu.edu

***Abstract*—In this paper, we propose a distributed cross-layer scheduling algorithm for 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. 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.**

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 have been proposed for wireless networks with time-varying channels [2]-[4]. Cross-layer algorithms with congestion controllers at transport layer have been proposed in [5]-[8]. Delay issues have been addressed in [9]-[12]. However, the high complexity of these centralized algorithms (NP hard for general interference models [13]) render them impractical for implementation. Low-complexity and distributed alternatives have been proposed in [14]-[17], 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 [23]-[29]. 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, leave room for significant improvements [24]. Moreover, the delay analysis still remains elusive despite some preliminary results [29] 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 [24][26][29]. 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 [23][24], which is justified in [27][28].

Network stability for algorithms such as [29][31] 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 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

978-3-901882-45-6/2012 - Copyright is with IFIP

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 $O(\frac{1}{\epsilon})$ 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. Further extensions to our algorithm are provided in [35], including (1) the construction of individual upper-bounds for average link-based delay; (2) extension to approximate the time-scale separation, which lies at the heart of our algorithm as well as those introduced in [23] [24]; (3) a distributed algorithm for multi-hop wireless networks.

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 present implementation and numerical results in Sections V and VI, respectively. Finally, we conclude our work in Section VII.

II. Related Works

Throughput optimal algorithms have primarily been designed without limitations on buffer sizes. Only recently, throughput/utility-optimal algorithms with finite buffers or worst-case delay guarantees have been proposed in [18]-[22]. Specifically, the multi-hop scheduling algorithms in [18][19] 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 [18][19] is not guaranteed for single-hop transmissions. Virtual queue structures are utilized in [20]-[22] to provide finite buffer size or worst-case delay performance. However, these algorithms are not well suited for distributed implementation due to their centralized nature and high complexity.

It has recently been shown in [29] that random access algorithms employing the Glauber dynamics [24]-[26] 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 [29], $\log U_l$ in [24], $\log\log U_l$ in [25], $\frac{\log U_l}{y(U_l)}$ in [26], 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 [24][26], or to ensure a negative Lyapnov drift [25][29]. 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 multi-hop extension of this model is presented in detail in [35].

Let $G = (\mathcal{L}, \mathcal{E})$ be the interference graph (a.k.a. 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 $\forall j \in \mathcal{N}(l)\backslash\{l\}$, 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.

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][5], 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. 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:

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

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 [30] we have

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

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

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

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 (normalized) 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 $\limsup_{t\to\infty} \frac{1}{t}\sum_{\tau=0}^{t-1}\mathbb{E}\{X(\tau)\} < \infty$. 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:

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

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. In the algorithm proposed in the next section, the decision variables $A_l(t)$ and $\mu_l(t)$ will be determined by a congestion controller and link rate scheduler, respectively.

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 the next section. 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 μ_M on $R_l(t)$, and update queue $Q_l(t)$ *virtually* as follows:

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

with $Q_l(0) = 0$. In the algorithm proposed in the next section, the decision variable $R_l(t)$ will be determined by an $R_l(t)$ regulator. 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:

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

[1]Note that the constantly backlogged packet generator is not necessarily a physical queue. It can represent an application waiting for packet generation and admission, 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.

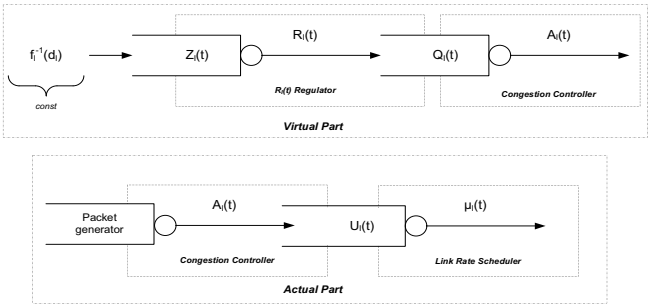


Fig. 1. Queue relationships with decision variables: $R_l(t)$, $A_l(t)$ and $\mu_l(t)$, $\forall l \in \mathcal{L}$.

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:

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

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 Figure 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. 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 pass through virtual queues.

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 to achieve optimal utility. For instance, in Section V in [35], 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. The Proposed Algorithm for Networks with Single-Hop Transmissions

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. ***ALG*** stabilizes the network and satisfies the minimum per flow 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, which will be 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 ***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 Figure 1).

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

$$\min_{0\leq R_l(t)\leq \mu_M} g_l(R_l(t)), \tag{7}$$

where $g_l(R_l(t)) \triangleq R_l(t)(\frac{q_M-\mu_M}{q_M}Q_l(t) - Z_l(t)) - Vf_l(R_l(t))$ 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 the first term of $g_l(R_l(t))$. We will show later in Theorem 1 that with a large control parameter V, ***ALG*** can approach the optimal network utility. With the numerical results in Section VI-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:

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

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 [24][26][29] 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, $\mu_l(t) = 0$.

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

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 [24] and evolves as a discrete-time Markov chain (DTMC). The distributed implementation of the link rate scheduler is provided in [35] (including the random selection of $\mathbf{x}(t) \in \mathcal{I}$), 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:

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

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

Proposition 2: Let $p_l(t) = \frac{e^{w_l(t)}}{1+e^{w_l(t)}}$, $\forall l \in \mathcal{L}$ 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:

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

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 Proposition 2 in [24] 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.

Remark 1 (Implications of the Propositions): A maximal weight matching scheduler that obtains $w^*(t)$ in each time slot t is usually centralized and computationally prohibitive, but can lead to optimal throughput/utility [1][5][20]. 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 $\frac{U_l(t)}{q_M}$. 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 virtual queues $Q_l(t)$ are large enough. 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 [24], $w_l(t) = \log\log U_l(t)$ proposed in [25], $w_l(t) = \frac{\log U_l(t)}{y(U_l(t))}$ in [26], where $y(\cdot)$ is some function that increases arbitrarily slowly, and $w_l(t)$ is in the form of $U_l(t)$ in [29]. 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 $\gamma > \max_{l\in\mathcal{L}} \frac{f_l^{-1}(d_l)}{a^*_{l,\epsilon}+\frac{1}{2}\epsilon}$, if

$$q_M > \frac{\mu_M^2+1}{\gamma\epsilon} + \mu_M, \tag{12}$$

then ***ALG*** ensures that the virtual queues are stable:

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

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

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

and

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

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

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

Proof: We prove Theorem 1 in Appendix A using Propositions 1 and 2. ■

Remark 2 (Algorithm Performance):

1) *Network Stability and Minimum Utility Constraints:* (10) in Proposition 1 indicates that ***ALG*** stabilizes the actual queue backlogs. As an immediate result, the network is stable. (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 VI-A.

2) *Utility Optimality and Buffer Size of order $O(\frac{1}{\epsilon})$:* The inequality (14) gives the lower-bound of the utility ***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 $\sum_{l\in\mathcal{L}} f_l(a^*_{l,\epsilon}+\frac{1}{2}\epsilon)$ 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 $O(\frac{1}{\epsilon})$ 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 [23][24] and justified in [27][28]. We have introduced approaches to approximate time-scale separation in [35].

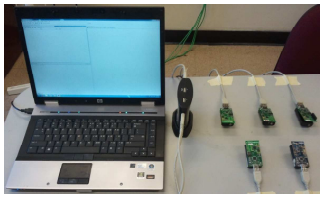
V. Implementation Study

To validate ***ALG***, the proposed algorithm for systems with constantly backlogged sources in Section IV, we compare ***ALG*** with the Q-CSMA algorithm in a testbed. Q-CSMA is a distributed throughput-optimal algorithm proposed in [24], with link weights in the scheduler chosen as $w_l(t) = \log(U_l(t))$. This has been shown in [24][26] to achieve better delay performance than the throughput-optimal algorithms with link weights of the form $U_l(t)$ [23] and $\log\log(U_l(t))$ [25].

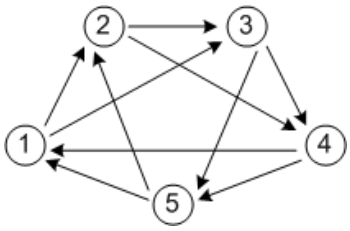
To guarantee fairness in the comparison, we construct the same congestion controller (8) for Q-CSMA and ***ALG***. It is easy to check that queue backlogs $U_l(t)$ are also bounded by q_M in our implementation of Q-CSMA. Therefore, q_M can be taken as the buffer size.

We implement the proposed algorithm and Q-CSMA in hardware on the Crossbow Telos-B platform as shown in Figure 2(a). Each node is equipped with a CC2420 802.15.4 wireless transceiver [32] and a programmable MSP430 processor [33]. For each protocol implementation, a time-slotted structure was used to coordinate the distributed link rate scheduler as proposed in [35]. 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 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 [34].

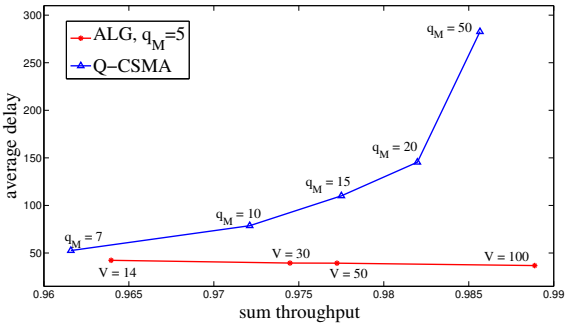
We test both ***ALG*** and Q-CSMA on the topology shown in Figure 2(b) with 5 nodes and 10 directional links. In this test, all nodes are within communication range of each other and therefore are in the same interference set, i.e., we employ the two-hop interference model. Under this model, at most 1 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



(a) Hardware testbed setup



(b) network topology for implementation



(c) Test results: delay versus throughput

Fig. 2. Implementation results

and throughput optimization instead of utility constraint and utility optimization. We choose the link rate weights as $w_l = \frac{0.1}{q_M} U_l(t) Q_l(t)$, $\forall l \in \mathcal{L}$, as proposed in Section VI in [35]. The maximal data admission rate is $\mu_M = 2$.

Figure 2(c) shows the delay performance comparison between ***ALG*** and Q-CSMA as a function of network throughput, where packet delay is measured in time slots and throughput in packets per time slot. 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 10000 time slots. To ensure a fair comparison with Q-CSMA, we do not impose minimum utility/throughput constraints on individual links for both ***ALG*** and Q-CSMA. By increasing V, the network throughput approaches the optimal throughput as stated in Remark 2, while *the delay performance was not sacrificed* under ***ALG***. This might sound counter-intuitive, but can be explained by the Little's Law. With an increased 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 larger 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 the 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 grow over the course of the 10000 time slot tests. A more detailed explanation of the effect of V on the system convergence time is given through simulation results in Section VI.

VI. Further Numerical Results

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

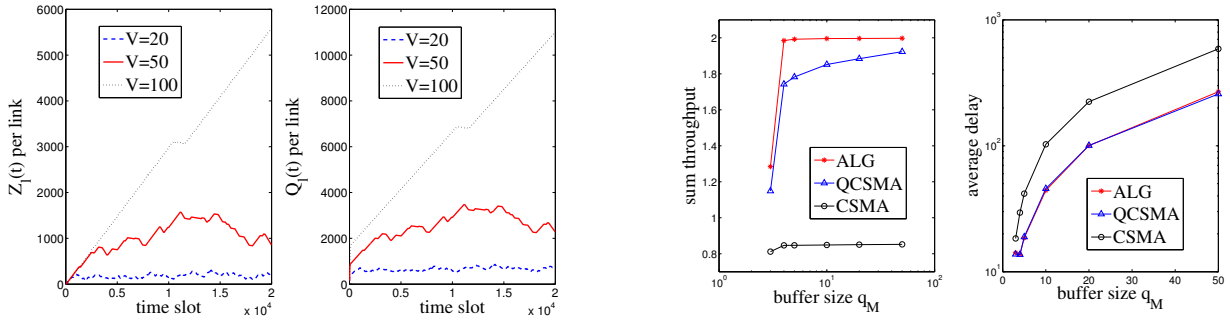
A. Effect of Parameter V

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

We first illustrate that a large value of V, which we recall is a control parameter in the $R_l(t)$ regulator (7) proposed for the backlogged source system, has a negative effect on the convergence rates of virtual queues. The virtual queue evolutions over time are shown in Figure 3(a), where we fix the buffer size as $q_M = 5$ and the utility constraint (equivalent to data rate constraint) as $d_l = 0.1$, $\forall l \in \mathcal{L}$. 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 [35] 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 subsection, based on the simulation setup in Section VI-A, we compare ***ALG*** with two other algorithms:
(i) A distributed throughput-optimal algorithm [26], with link weights in the scheduler chosen as $w_l(t) = \frac{\log(U_l(t))}{\log(e+\log(1+U_l(t)))}$, which has been shown in simulation to achieve better delay performance than the throughput-optimal algorithms in [23][25]. Since it is observed in [26] that this algorithm is similar in throughput and delay performance with the Q-CSMA [24], we denote it by *QCSMA*.
(ii) A traditional CSMA algorithm which employs RTS/CTS



(a) Virtual queue evolutions

(b) Comparison with different buffer sizes

Fig. 3. Simulation results

handshake to reserve the channel. In the following, we denote this algorithm as *CSMA*.

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.

Figure 3(b) 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 Figure 3(b) 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.

Similar delay performance comparisons as in Figure 2(c) have been observed in [35], which shows that ***ALG*** can achieve close-to-optimal throughput with significantly more favorable delay with respect to QCSMA. Further simulations in [35] are carried out in a large grid topology to show that ***ALG*** does not suffer from temporal starvation [29] (It may take a long time to switch from one maximal weight schedule to another in CSMA algorithms such as Q-CSMA.) as much as Q-CSMA in the grid topology setting and can still achieve far better delay performance.

VII. Conclusions 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 $O(\frac{1}{\epsilon})$, where ϵ characterizes the distance between the achieved utility and the optimal value. The distributed algorithm can be implemented via CSMA methods with RTS/CTS handshake and is shown to achieve close to optimal throughput and good delay performance both in simulation and testbed implementations. In our future work, we will investigate more efficient implementation methods of the proposed algorithm. Moreover, we will extend the proposed algorithm to multi-hop flows with the express intent of implementation feasibility.

References

[1] L. Tassiulas and A. Ephremides, "Stability properties of constrained queueing systems and scheduling policies for maximum throughput in multihop radio networks", in *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", in *IEEE/ACM Transactions on Networking*, vol. 13, no. 2, pp. 411-424, April 2005.

[3] J. Ryu, L. Ying and S. Shakkottai, "Back-pressure routing for intermittently connected networks", in *IEEE INFOCOM 2010 Mini-Conference*, March 2010.

[4] M. Neely and E. Modiano, "Dynamic power allocation and routing for time varying wireless networks", in *IEEE Journal on Selected Area in Communications*, vol.23, no.1, pp. 89-103, March 2005.

[5] L. Georgiadis, M. Neely and L.Tassiulas, "Resource allocation and cross-Layer control in wireless networks", in *Foundations and Trends in Networking*, pp. 1-149, 2006.

[6] X. Liu and E. Chong, and N. Shroff, "Opportunistic transmission scheduling with resource-sharing constraints in wireless networks", IEEE Journal on Selected Areas in Communications, vol. 19, no. 10, pp. 2053-2065, October, 2001.

[7] M. Neely, "Energy Optimal Control for Time Varying Wireless Networks", in *IEEE Transactions on Information Theory*, vol. 52, no. 7, pp. 2915-2934, July 2006.

[8] A. Eryilmaz and R. Srikant, "Joint congestion control, routing and MAC for stability and fairness in wireless networks", in *IEEE Journal on Selected Areas in Communications*, vol. 24, no. 8, pp. 1514-1524, August 2006.

[9] 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 2010*, March 2010.

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

[11] L. Ying, S. Shakkottai, and A. Reddy, "On combining shortest-path and back-pressure routing over mult-hop wireless networks", in *Proc. IEEE INFOCOM'09*, April 2009.

[12] H. Xiong, R. Li, A. Eryilmaz, and E. Ekici, "Delay-Aware Cross-Layer Design for Network Utility Maximization in Multi-hop Networks", in *IEEE Journal on Selected Areas in Communications*, vol. 29, no. 5, pp. 951-959, May 2011.

[13] G. Sharma, R. Mazumdar and N. Shroff, "On the complexity of scheduling in wireless neworks", in *Proc. of the 12th Annual International Conference on Mobile Computing and Networking (MobiCom'06)*, pp. 227-238, 2006.

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

[15] P. Chaporkar, K. Kar and S. Sarkar, "Throughput guarantees through maximal scheduling in wireless networks", in *Proc. 43rd Annual Allerton Conference on Communicastions, Control and Computing*, 2005.

[16] 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'07*, May 2007.

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

[18] P. Giaccone, E. Leonardi, and D. Shah, "Throughtput region of finite-buffered networks", in *IEEE Transactions on Parallel and Distributed Systems*, vol. 18, no. 2, pp. 251-263, Feb. 2007.

[19] L. Le, E. Modiano, and N. Shroff, "Optimal control of wireless networks with finite buffers", in *Proc. IEEE INFOCOM 2010*, March 2010.

[20] D. Xue and E. Ekici, "Delay-guaranteed cross-layer scheduling in multi-hop wireless networks", Technical Report, 2011. Available: http://arxiv.org/abs/1009.4954.

[21] M. Neely, "Delay-based network utility maximization", in *Proc. IEEE INFOCOM 2010*, March 2010.

[22] M. Neely, "Opportunistic scheduling with worst case delay guarantees in single and multi-hop networks", in *Proc. IEEE INFOCOM 2011*, April 2011.

[23] L. Jiang and J. Walrand,"a distributed algorithm for throughput and utility maximization in wireless networks", in *IEEE/ACM Transactions on Networking*, vol. 18, no.3, pp. 960-972, June 2010.

[24] 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 2010 Mini-Conference*, April 2010.

[25] S. Rajagopalan, D. Shah, and J. Shin, "Network adiabatic theorem: an efficient randomized protocol for contention resolution", in *SIGMETRICS'09: Proceeding of the eleventh international joint conference on measurement and modeling of computer systems*, pp. 133-144, 2009.

[26] J. Ghaderi and R. Srikant, "On the design of efficient CSMA algorithms for wireless networks", in *Proc. IEEE conference on Decision and Control (CDC 2010)*, December, 2010.

[27] A. Proutiere, Y. Yi, T. Lan, and M. Chiang, "Resource allocation over network dynamics without timescale separation", in *Proc. IEEE INFOCOM 2010 Mini-Conference*, March 2010.

[28] L. Jiang and J. Walrand, "Convergence and stability of a distributed CSMA algorithm for maximal network throughput", in *IEEE CDC'09*, 2009.

[29] L. Jiang, M. Leconte, J. Ni, R. Srikant, and J. Walrand, "Fast mixing of parallel Glauber dynamics and low-delay CSMA scheduling", in *IEEE INFOCOM 2011 Mini-Conference*, April 2011.

[30] M. Neely, "Dynamic power allocation and routing for satellite and wireless networks with time varing channels", Ph.D. dissertation, Mass. Inst. Technol. (MIT), Cambridge, MA, 2003.

[31] A. Eryilmaz, R. Srikant, and J. Perkins, "Stable scheduling policies for fading wireless channels", in *IEEE/ACM Transactions on Networking*, pp. 411-424, Vol. 13, No.2 ,April 2005.

[32] Texas Instruments, CC2420: http://focus.ti.com/docs/prod/folders/print/cc2420.html

[33] Texas Instruments, MSP430 Microcontroller: http://focus.ti.com/docs/prod/folders/print/msp430f1611.html

[34] 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 SECON 2010, 21-25 2010, pp. 1 - 9.

[35] D. Xue and E. Ekici, "Distributed utility-optimal scheduling with finite buffers", Technical Report, Ohio State University, available: http://www.ece.osu.edu/∼xued/distributed.pdf

APPENDIX A
PROOF OF THEOREM 1

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:

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

where we note that different from the traditional quadratic Lyapunov function, there is the (weighted) virtual queue backlog $\frac{Q_l(t)}{q_M}$ multiplied to the quadratic term of actual queue backlogs. We denote the Lyapunov drift by $\Delta(t) \triangleq \mathbb{E}\{L_1(\mathbf{Q}_1(t+1)) - L_1(\mathbf{Q}_1(t))|\mathbf{Q}_1(t)\}$.

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

$$\begin{aligned}&\Delta(t) - V\sum_{l\in\mathcal{L}}\mathbb{E}\{f_l(R_l(t))|\mathbf{Q}_1(t)\}\\ \leq& B_1 + \frac{\mu_M^2+1}{2q_M}\sum_{l\in\mathcal{L}}Q_l(t) + \sum_{l\in\mathcal{L}}f_l^{-1}(d_l)Z_l(t)\\ &+\sum_{l\in\mathcal{L}}\mathbb{E}\{g_l(R_l(t))|\mathbf{Q}_1(t)\}\\ &-\sum_{l\in\mathcal{L}}\frac{1}{q_M}U_l(t)Q_l(t)\mathbb{E}\{\mu_l(t)|\mathbf{Q}_1(t)\}\\ &-\sum_{l\in\mathcal{L}}\frac{Q_l(t)}{q_M}\mathbb{E}\{A_l(t)|\mathbf{Q}_1(t)\}(q_M-\mu_M-U_l(t)).\end{aligned} \tag{15}$$

Denoting the RHS of (15) by $RHS(t)$, and taking the expectation of both sides of (15) with respect to the distribution of $\mathbf{Q}_1(t)$, we have:

$$\begin{aligned}&\mathbb{E}\{\Delta(t)\} - V\sum_{l\in\mathcal{L}}\mathbb{E}\{f_l(R_l(t))\}\\ \leq& P(||\mathbf{q}(t)|| > B)\mathbb{E}_{\{\mathbf{Q}_1(t)|||\mathbf{q}(t)||>B\}}\{RHS(t)\}\\ &+P(||\mathbf{q}(t)|| \leq B)\mathbb{E}_{\{\mathbf{Q}_1(t)|||\mathbf{q}(t)||\leq B\}}\{RHS(t)\}.\end{aligned} \tag{16}$$

The fourth term and the last term of the $RHS(t)$ are minimized by the $R_l(t)$ regulator (7) and the congestion controller (8), respectively, over a set of feasible algorithms including a set of stationary randomized algorithms (STATs) introduced in [35]. From (16) and following the analysis in [35] by substituting a set of STATs, we can obtain

$$\begin{aligned}&\mathbb{E}\{\Delta(t)\} - V\sum_{l\in\mathcal{L}}\mathbb{E}\{f_l(R_l(t))\}\\ \leq& B_2 - \delta_1\sum_{l\in\mathcal{L}}\mathbb{E}\{Q_l(t)+Z_l(t)\} - \gamma V\sum_{l\in\mathcal{L}}f_l(a^*_{l,\epsilon}+\frac{1}{2}\epsilon).\end{aligned} \tag{17}$$

We take the time average on $\tau = 0, ..., t-1$ of both sides of (17), 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.