Scheduling in Multihop Wireless Networks Without Back-Pressure

Shihuan Liu, *Member, IEEE*, Eylem Ekici, *Senior Member, IEEE*, and Lei Ying, *Member, IEEE*

***Abstract*—This paper focuses on scheduling in multihop wireless networks where flows are associated with fixed routes. The well-known back-pressure scheduling algorithm is throughput-optimal, but requires constant exchange of queue length information among neighboring nodes for calculating the "back-pressure." Moreover, previous research shows that the total queue length along a route increases quadratically as the route length under the back-pressure algorithm, resulting in poor delay performance. In this paper, we propose a self-regulated MaxWeight scheduling, which does not require back-pressure calculation. We prove that the self-regulated MaxWeight scheduling is throughput-optimal (an algorithm is said to be throughput-optimal if it can stabilize any traffic that can be stabilized by any other algorithm). In the simulation part, we show that the self-regulated MaxWeight scheduling has a much better delay performance than the back-pressure algorithm.**

***Index Terms*—Back-pressure, scheduling, throughput-optimal, wireless multihop networks.**

I. Introduction

THIS paper considers the scheduling problem in multihop wireless networks. A widely used algorithm to stabilize multihop flows in wireless networks is the back-pressure algorithm proposed in [2], which can stabilize any traffic load that can be supported by any other routing/scheduling algorithm. We refer to [3] and [4] for a comprehensive survey on the back-pressure algorithm and its variations. A key idea of the back-pressure algorithm is to use the largest queue difference as link weight and schedule the set of links with the largest aggregated weights. Therefore, the back-pressure algorithm requires constant exchange of queue length information among neighboring nodes. Furthermore, under the back-pressure algorithm, the sum of the queue lengths along a route increases quadratically as the length of the route [5], which leads to poor delay performance. The delay performance of the back-pressure algorithm has received much attention recently, and different approaches have been developed to improve the delay performance of the back-pressure [5]–[8]. These algorithms, however, still rely on the *back-pressure calculation* for scheduling, so they still require a constant exchange of queue length information.

In this paper, we consider the following question: *Can the network be stabilized without using the back-pressure?* We address this question in a multihop wireless network with fixed routing. We note that a multihop flow with a fixed route can be broken into multiple single-hop flows, one for each link on the route. A scheduling policy that stabilizes the collection of single-hop flows also provides sufficient service rates for supporting the set of multihop flows. Therefore, assuming each link knows the aggregated rate it needs to carry, an alternative scheduling approach is to let each link generate virtual packets [5], [9] according to the aggregated rate and then let the network schedule the links according to the virtual queues. When a virtual queue is scheduled, real packets are served according to the allocated link rate. This approach is throughput-optimal under the fixed routing assumption, but again requires information exchange in the network. A source needs to estimate the arrival rate of the associated flow and communicate the rate to all nodes along the route of the flow. A directly following question is whether, and under what conditions, it is possible to stabilize a network without explicitly exchanging any information among nodes in the network. In the following, we present an algorithm that can achieve this goal for networks where: 1) the arrival processes satisfy some statistical property; and 2) the routes of flows are fixed.

We propose a self-regulated MaxWeight scheduling algorithm where each node estimates the aggregated link rate locally, i.e., by taking average over the past arrivals on that link. We would like to emphasize that the accuracy of the link-rate estimates relies on the stability of the network because if one queue builds up, it blocks packets to downstream nodes so that those nodes cannot accurately estimate the link rates. On the other hand, the stability of the network relies on the accuracy of the link-rate estimates. This is an interesting paradox that makes the stability of the self-regulated MaxWeight scheduling a nontrivial problem. In this paper, we prove that the self-regulated MaxWeight scheduling is throughput-optimal when the traffic flows are associated with fixed routes and the packet arrivals follow some statistical property. The self-regulated MaxWeight scheduling combined with distributed scheduling algorithms such as the CSMA-based scheduling [10]–[12] provides a scheduling algorithm for multihop wireless networks, which does not require any information exchange in the

Manuscript received July 08, 2012; revised February 26, 2013 and July 29, 2013; accepted August 08, 2013; approved by IEEE/ACM TRANSACTIONS ON NETWORKING Editor U. Ayesta. Date of publication September 05, 2013; date of current version October 13, 2014. This work was supported in part by the NSF under Grants CNS-0831756, CCF-0914912, CNS-1262329, and CNS-1264012 and the DTRA under Grants HDTRA1-08-1-0016 and HDTRA1-09-1-0055. An earlier version of this paper appeared in the Proceedings of the Annual Allerton Conference on Communication, Control, and Computing, Monticello, IL, USA, October 2–4, 2010.

S. Liu is with Qualcomm Incorporated, San Diego, CA 92121 USA (e-mail: shihuanl@qti.qualcomm.com).

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

L. Ying is with the School of Electrical, Computer and Energy Engineering, Arizona State University, Tempe, AZ 85287 USA (e-mail: lei.ying.2@asu.edu).

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.2013.2278840

network. Finally, we would like to comment that the proposed algorithm is motivated by the idea of regulators, which was first proposed for re-entrant lines in [13] and later used for scheduling in wireless networks [14]. Our algorithm, however, does not require any information exchange in network, while the algorithm in [14] requires the mean arrival rates to be communicated to the regulators in the network.

II. Basic Model

We consider a network represented by a graph $\mathcal{G} = (\mathcal{N}, \mathcal{L})$, where $\mathcal{N}$ is the set of nodes and $\mathcal{L}$ is the set of directed links. Let $N = |\mathcal{N}|$ and $L = |\mathcal{L}|$. Denote by (m, n) the link from node m to node n, which implies that node m can communicate with node n. Furthermore, let $\boldsymbol{\mu} = \{\mu_{(m,n)}\}$ denote a link-rate vector such that $\mu_{(m,n)}$ is the transmission rate over link (m, n). A link-rate vector $\boldsymbol{\mu}$ is said to be *admissible* if the link-rates specified by $\boldsymbol{\mu}$ can be achieved *simultaneously*. Define Γ to be the set of all admissible link-rate vectors. It is easy to see that Γ depends on the choice of interference model and might not be a convex set. Furthermore, Γ is time-varying if channels are time-varying. We further assume that there exists $\mu_{\max}$ such that $\mu_{(m,n)} \leq \mu_{\max}$ for all $(m, n) \in \mathcal{L}$ and all admissible $\boldsymbol{\mu}$.

We consider multihop traffic flows with fixed routing in this paper and assume each flow is associated with a single route. Let f denote a flow, and $\mathbf{R}_f$ denote the route associated with flow f. Denote by S_f the source node of flow f, and D_f the destination node of flow f. Furthermore, let $\mu^f_{(m,n)}$ denote the rate at which packets of flow f are served over link (m, n). We use $\mathcal{F}$ to denote the set of all flows in the network, and $F = |\mathcal{F}|$. Assume that time is discretized, and let $X_f(t)$ $(f \in \mathcal{F})$ denote the number of packets injected by flow f at time t. We assume $X_f = \mathbf{E}[X_f(t)]$, which is the arrival rate of flow f. We further assume $X_f(t)$ is upper-bounded, i.e., $X_f(t) \leq X^{\max}$ for all f and t. Denoting by $A_f(t) = \sum_{\tau=1}^{t} X_f(\tau)$ which is the aggregated arrival rate of flow f at time t, we assume that $A_f(t)$ satisfies the following property: Given any $\epsilon > 0$, there exists T_ϵ such that for any $s - t > T_\epsilon$

$$\left| \frac{A_f(s) - A_f(t)}{s - t} - X_f \right| < \epsilon \tag{1}$$

for all f.

Note that if $X_f(t)$ is independent and identically distributed over time, since $A_f(s) - A_f(t) = \sum_{\tau=t+1}^{s} X_f(\tau)$, according to the Strong Law of Large Numbers, $\Pr(\lim_{s-t\to\infty}(A_f(s) - A_f(t)/s - t) = X_f) = 1$. However, condition (1) requires a uniform convergence of all sample paths and is stronger than the sample path convergence of random variables guaranteed by SLLN. We do not assume $X_f(t)$ are i.i.d. across time because condition (1) implies the arrival process is dependent.

III. Necessary Conditions for Stability

In this section, we characterize the necessary conditions of stability. We say a traffic load $\{X_f\}_{f\in\mathcal{F}}$ is supportable if there exists $\{\bar{\mu}^f_{(m,n)}\}_{(m,n)\in\mathcal{L}}$ such that the following two conditions hold.

(i) For any flow f and node $n \neq D_f$

$$X_f \mathbb{I}_{\{S_f=n\}} + \sum_{m:(m,n)\in\mathbf{R}_f} \bar{\mu}^f_{(m,n)} \leq \sum_{b:(n,b)\in\mathbf{R}_f} \bar{\mu}^f_{(n,b)} \tag{2}$$

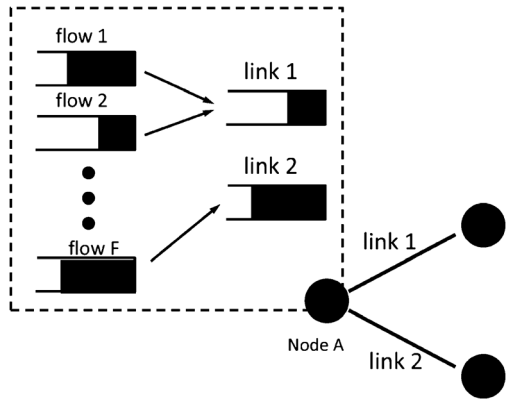

Fig. 1. Two-stage queue architecture.

where $\mathbb{I}_{\{S_f=n\}}$ equals one if $S_f = n$, and zero otherwise.

(ii)

$$\left\{ \sum_{f\in\mathcal{F}} \bar{\mu}^f_{(m,n)} \right\} \in \mathcal{CH}(\Gamma) \tag{3}$$

where $\mathcal{CH}(\Gamma)$ is the convex hull of Γ.

Recall that each flow is associated with a fixed route. It is easy to see the necessary condition (2) is equivalent to the following statement: For any flow f and $(m, n) \in \mathbf{R}_f$, we have

$$\bar{\mu}^f_{(m,n)} \geq X_f. \tag{4}$$

IV. Self-Regulated MaxWeight Scheduling for Multihop Wireless Networks

In this section, we introduce the self-regulated MaxWeight scheduling for multhop wireless networks, which will be proved to be throughput-optimal later.

Two-Stage Queue Architecture: Each node maintains two types of queues: per-flow queues and per-link queues as shown in Fig. 1. An incoming packet is at first buffered at the corresponding per-flow queue and then moved to the per-link queue (the details will be described later). In this paper, we denote by Q^n_f the length of the queue maintained at node n for flow f, and $Q^n_{(n,b)}$ the length of the queue maintained at node n for link (n, b). Clearly, the queue for link (n, b) is maintained only at node n, so we simplify $Q^n_{(n,b)}$ to be $Q_{(n,b)}$ without causing any confusion.

Self-Regulated MaxWeight Scheduling:

- MaxWeight scheduling: Compute an admissible link-rate vector $\mu^*(t)$ such that

$$\mu^*(t) = \arg\max_{\boldsymbol{\mu}(t)\in\Gamma} \sum_{(n,b)\in\mathcal{L}} \mu_{(n,b)}(t) Q_{(n,b)}(t). \tag{5}$$

  Note that here instead of using the back-pressures as link weights, we use the queue lengths of per-link queues as link weights to make scheduling decisions.
- Per-link queue transmission: Node n transmits

$$s_{(n,b)}(t) \triangleq \min\left\{ \mu^*_{(n,b)}(t), Q_{(n,b)}(t) \right\}$$

  packets to node b over link (n, b). The packets are deposited into per-flow queues at node b according to the

flows to which they belong. We let $s^f_{(n,b)}(t)$ denote the number of packets of flow f that are transmitted over link (n,b) at time t. Note that

$$s_{(n,b)}(t) = \sum_f s^f_{(n,b)}(t)$$

always holds.

- Per-flow queue transmission: Denote by $a^n_f(t)$ the number of packets deposited into queue Q^n_f at time-slot t, i.e.,

$$a^n_f(t) = \sum_{(b,n)\in\mathcal{L}} s^f_{(b,n)}(t)\mathbb{I}_{(b,n)\in\mathbf{R}_f}$$

if node n is not the source node of flow f, and

$$a^{S_f}_f(t) = X_f(t)$$

if node n is the source node of flow f.
For each flow f, node n maintains a rate estimate

$$\tilde{X}^n_f(t) = \left(1 + \frac{1}{Q_{(n,n_f)}(t)}\right)\frac{\sum_{\tau=1}^t a^n_f(\tau)}{t} \tag{6}$$

where n_f is the next hop from node n on the route of flow f. If $Q_{(n,n_f)}(t) = 0$, then let

$$\tilde{X}^n_f(t) = (1+\gamma)\frac{\sum_{\tau=1}^t a^n_f(\tau)}{t}$$

where $\gamma > 1$ is a positive constant we can set. At time-slot t, node n moves

$$s^n_f(t) \triangleq \min\left\{\tilde{X}^n_f(t), Q^n_f(t)\right\}$$

packets from queue Q^n_f to queue $Q_{(n,n_f)}$. We further define the packet arrivals of per-link queues

$$a_{(n,b)}(t) \triangleq \sum_{f:(n,b)\in\mathbf{R}_f} s^n_f(t).$$

Remark: Note that we use weight $1+(1/Q_{(n,n_f)}(t))$ instead of $1+\gamma$ in the rate estimate (6). In [1], we proved that the proposed algorithm supports any λ such that $(1+\gamma)\lambda$ is strictly within the network throughput region. By replacing the constant γ with the inverse of the queue length $1/Q_{(n,n_f)}(t)$, the algorithm can support any λ that is strictly within the throughput region. Furthermore, in a light traffic regime, $Q_{(n,n_f)}(t)$ is expected to be small, so $1+(1/Q_{(n,n_f)}(t))$ will not be very close to one. With a large weight in the rate estimate, the algorithm moves packets from per-flow queues to per-link queues quickly so the per-flow queues will not become the bottlenecks, which reduces end-to-end transmission delays.

The intuition of this two-stage queue architecture is that we break each multihop flow into multiple single-hop flows, one for each link on the route. A scheduling policy stabilizing the collection of single-hop flows also provides sufficient service rates for supporting the multihop flow. Therefore, if each link knows the required rate for it to carry, it can simply generate virtual packets [5], [9] according to the required rate and then let the network make scheduling decisions according to the virtual queues (per-link queues). When a virtual queue is scheduled, real packets are served according to the allocated link rate. Note that back-pressure scheduling becomes MaxWeight scheduling here since all flows are single-hop flows. This method, however, requires the source nodes to communicate the data rates to all the links on the routes and thus requires additional communication overhead. To totally remove the communication overhead, we let each node estimate the required link rate locally by averaging over the past arrivals as in (6).

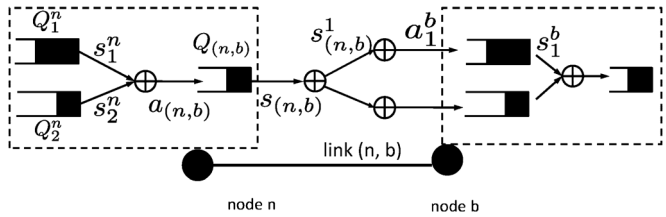

Fig. 2. Notations and the flows.

The notations are illustrated in Fig. 2. From the definition of the self-regulated MaxWeight scheduling algorithm, we can see that the dynamics of queue Q^n_f and $Q_{(n,b)}$ are

$$Q^n_f(t+1) = \left[Q^n_f(t) - s^n_f(t)\right]^+ + a^n_f(t) \tag{7}$$

$$Q_{(n,b)}(t+1) = \left[Q_{(n,b)}(t) - s_{(n,b)}(t)\right]^+ + a_{(n,b)}(t). \tag{8}$$

In the following, we will prove that the self-regulated MaxWeight scheduling algorithm stabilizes any traffic within the stability region of the network. The analysis consists of two steps: We first show that the per-link queues are bounded (Lemma 1), and then using induction to prove that the per-flow queues are bounded as well (Lemma 2).

Lemma 1: Given a set of traffic flows such that $\{(1+\epsilon)X_f\}$ is supportable for some $\epsilon > 0$, there exists a positive constant $C > 0$ such that the lengths of any per-link queue is no more than C for all $t \geq 0$.

Proof: Since we study the stability of the system, we are only concerned about the queue lengths when t is large. From (1) we know that for any given $\delta > 0$, we can find $T(\delta)$, such that when $t > T(\delta)$

$$\frac{A_f(t)}{t} \leq X_f + \delta \qquad \forall f.$$

Therefore, for any node n

$$\sum_{\tau=1}^t a^n_f(\tau) \leq X_f t + \delta t$$

which implies that

$$\tilde{X}^n_f(t) \leq \left(1 + \min\left\{\gamma, \frac{1}{Q_{(n,n_f)}(t)}\right\}\right)(X_f + \delta) \qquad \forall f.$$

Since $\{(1+\epsilon)X_f\}$ is supportable, according to the necessary conditions for stability (refer to the condition), there exists $\bar{\mu}$ such that

$$\bar{\mu}^f_{(m,n)} \geq (1+\epsilon)X_f$$

for any flow f and link (m,n) that is on the route of f. Let $\bar{\mu}_{(m,n)} = \sum_{f:(m,n)\in\mathbf{R}_f} \bar{\mu}^f_{(m,n)}$, then for any link $(m,n) \in \mathcal{L}$, we can conclude that

$$\begin{aligned}
a_{(m,n)}(t) &= \sum_{f:(m,n)\in\mathbf{R}_f} \tilde{X}^m_f(t) \\
&\le \sum_{f:(m,n)\in\mathbf{R}_f} \left(1+\min\left\{\gamma, \frac{1}{Q_{(m,n)}(t)}\right\}\right)(X_f+\delta) \\
&= \sum_{f:(m,n)\in\mathbf{R}_f} (1+\epsilon)X_f \\
&\quad - \sum_{f:(m,n)\in\mathbf{R}_f} \left(\epsilon - \min\left\{\gamma, \frac{1}{Q_{(m,n)}(t)}\right\}\right) X_f \\
&\quad + \sum_{f:(m,n)\in\mathbf{R}_f} \left(1+\min\left\{\gamma, \frac{1}{Q_{(m,n)}(t)}\right\}\right)\delta \\
&\le \sum_{f:(m,n)\in\mathbf{R}_f} \bar{\mu}^f_{(m,n)} \\
&\quad - \sum_{f:(m,n)\in\mathbf{R}_f} \left(\epsilon - \min\left\{\gamma, \frac{1}{Q_{(m,n)}(t)}\right\}\right) X_f \\
&\quad + \sum_{f:(m,n)\in\mathbf{R}_f} \left(1+\min\left\{\gamma, \frac{1}{Q_{(m,n)}(t)}\right\}\right)\delta \\
&= \bar{\mu}_{(m,n)} \\
&\quad - \sum_{f:(m,n)\in\mathbf{R}_f} \left(\epsilon - \min\left\{\gamma, \frac{1}{Q_{(m,n)}(t)}\right\}\right) X_f \\
&\quad + \sum_{f:(m,n)\in\mathbf{R}_f} \left(1+\min\left\{\gamma, \frac{1}{Q_{(m,n)}(t)}\right\}\right)\delta. \quad (9)
\end{aligned}$$

The dynamic of per-link queues can be rewritten as

$$Q_{(m,n)}(t+1) = Q_{(m,n)}(t) - s_{(m,n)}(t) + u_{(m,n)}(t) + a_{(m,n)}(t)$$

where $u_{(m,n)}(t)$ is the unused service due to the lack of packets in the queue. It is easy to see that when $Q_{(m,n)}(t) > \mu_{\max}$, $u_{(m,n)}(t) = 0$, so $Q_{(m,n)}(t) \times u_{(m,n)}(t) \le \mu^2_{\max}$.

Considering the following Lyapunov function:

$$V(t) = \sum_{(m,n)\in\mathcal{L}} Q^2_{(m,n)}(t).$$

We have

$$\begin{aligned}
&V(t+1) - V(t) \\
&\quad = \sum_{(m,n)\in\mathcal{L}} Q^2_{(m,n)}(t+1) - \sum_{(m,n)\in\mathcal{L}} Q^2_{(m,n)}(t) \\
&\quad = \sum_{(m,n)\in\mathcal{L}} \big(Q_{(m,n)}(t+1) + Q_{(m,n)}(t)\big) \\
&\qquad \times \big(Q_{(m,n)}(t+1) - Q_{(m,n)}(t)\big) \\
&\quad = \sum_{(m,n)\in\mathcal{L}} \big(2Q_{(m,n)}(t) + a_{(m,n)}(t) + u_{(m,n)}(t) - s_{(m,n)}(t)\big) \\
&\qquad \times \big(a_{(m,n)}(t) + u_{(m,n)}(t) - s_{(m,n)}(t)\big) \\
&\quad \le M_1 + \sum_{(m,n)\in\mathcal{L}} 2Q_{(m,n)}(t)\big(a_{(m,n)}(t) - s_{(m,n)}(t)\big) \quad (10) \\
&\quad = M_1 + \sum_{(m,n)\in\mathcal{L}} 2Q_{(m,n)}(t)\big(a_{(m,n)}(t) - \bar{\mu}_{(m,n)}\big) \\
&\qquad + \sum_{(m,n)\in\mathcal{L}} 2Q_{(m,n)}(t)\big(\bar{\mu}_{(m,n)} - s_{(m,n)}(t)\big) \quad (11)
\end{aligned}$$

where $M_1 = 2L\mu^2_{\max} + L[(\max_f X_f + \delta) \times F + \mu_{\max}]^2$.

From inequality (9), we can get

$$\begin{aligned}
&\sum_{(m,n)\in\mathcal{L}} 2Q_{(m,n)}(t) \times \big(a_{(m,n)}(t) - \bar{\mu}_{(m,n)}\big) \\
&\quad \le \sum_{(m,n)\in\mathcal{L}} 2Q_{(m,n)}(t) \\
&\qquad \times \left(\sum_{f:(m,n)\in\mathbf{R}_f} \left(1+\min\left\{\gamma, \frac{1}{Q_{(m,n)}(t)}\right\}\right)\delta \right. \\
&\qquad\quad \left. - \sum_{f:(m,n)\in\mathbf{R}_f} \left(\epsilon - \min\left\{\gamma, \frac{1}{Q_{(m,n)}(t)}\right\}\right) X_f\right) \\
&\quad \le \sum_{(m,n)\in\mathcal{L}} 2Q_{(m,n)}(t) \\
&\qquad \times \left(\min\left\{\gamma, \frac{1}{Q_{(m,n)}(t)}\right\}\left(\sum_{f\in\mathcal{F}} X_f + F\delta\right)\right. \\
&\qquad\quad \left. + F\delta - \sum_{f:(m,n)\in\mathbf{R}_f} \epsilon X_f\right) \\
&\quad \le M_2 + \sum_{(m,n)\in\mathcal{L}} 2Q_{(m,n)}(t)\left(F\delta - \sum_{f:(m,n)\in\mathbf{R}_f} \epsilon X_f\right)
\end{aligned}$$

where $M_2 = 2L(\sum_{f\in\mathcal{F}} X_f + F\delta)$.

Furthermore, from the MaxWeight scheduler (5) and necessary condition (3), we have

$$\begin{aligned}
&\sum_{(m,n)\in\mathcal{L}} 2Q_{(m,n)}(t) \times \big(\bar{\mu}_{(m,n)} - s_{(m,n)}(t)\big) \\
&\quad \le \sum_{(m,n)\in\mathcal{L}} 2Q_{(m,n)}(t) \times \left(\bar{\mu}_{(m,n)} - \mu^*_{(m,n)}(t)\right) + M_3 \\
&\quad \le M_3
\end{aligned}$$

where $M_3 = 2L\mu^2_{\max}$.

Therefore, we conclude that

$$\begin{aligned}
V(t+1) - V(t) &\le M_1 + M_2 + M_3 \\
&\quad + \sum_{(m,n)\in\mathcal{L}} 2Q_{(m,n)}(t)\left(F\delta - \sum_{f:(m,n)\in\mathbf{R}_f} \epsilon X_f\right).
\end{aligned}$$

It is easy to see that when $\{f \in \mathcal{F} : (m,n) \in \mathbf{R}_f\} = \emptyset$, $Q_{(m,n)}(t) = 0$. Thus

$$\begin{aligned}
&\sum_{(m,n)\in\mathcal{L}} 2Q_{(m,n)}(t)\left(F\delta - \sum_{f:(m,n)\in\mathbf{R}_f} \epsilon X_f\right) \\
&\qquad \le \sum_{(m,n)\in\mathcal{L}} 2Q_{(m,n)}(t)(F\delta - \epsilon \min_f X_f).
\end{aligned}$$

If we further choose δ such that $\delta \le (\epsilon \min_f X_f/2F)$, then

$$\sum_{(m,n)\in\mathcal{L}} 2Q_{(m,n)}(t)\left(F\delta - \sum_{f:(m,n)\in\mathbf{R}_f} \epsilon X_f\right) \le \sum_{(m,n)\in\mathcal{L}} 2Q_{(m,n)}(t)\left(-\frac{1}{2}\epsilon \min_f X_f\right).$$

Now note that $\max_{(m,n)\in\mathcal{L}} Q_{(m,n)}(t) \ge \sqrt{V(t)/L}$. Given a positive constant M, when $V(t) > L((M_1+M_2+M_3+M)/\epsilon \min_f X_f)^2$

$$\max_{(m,n)\in\mathcal{L}} Q_{(m,n)}(t) > \frac{M_1+M_2+M_3+M}{\epsilon \min_f X_f}.$$

Thus, we have

$$\begin{aligned} &V(t+1)-V(t)\\ &\quad\le M_1+M_2+M_3\\ &\qquad+\sum_{(m,n)\in\mathcal{L}} 2Q_{(m,n)}(t)\left(-\frac{1}{2}\epsilon \min_f X_f\right)\\ &\quad\le M_1+M_2+M_3-\max_{(m,n)\in\mathcal{L}} Q_{(m,n)}(t)(\epsilon \min_f X_f)\\ &\quad\le -M. \end{aligned}$$

From the above analysis, we can see when $t > T(\delta)$, if $V(t) > L((M_1+M_2+M_3+M/\epsilon \min_f X_f)^2$, the drift of the Lyapunov function is negative. Moreover, when $t \le T(\delta)$, $V(t) \le L(FX^{\max}T(\delta))^2$. Therefore, there exists a constant C_V such that $V(t) \le C_V$ for all t. Hence, $Q_{(m,n)}(t) \le \sqrt{C_V} \triangleq C$ for any $(m,n)\in\mathcal{F}$ and t. □

Next, based on Lemma 1, we will prove that all per-flow queues are bounded as well.

Lemma 2: Given a set of traffic flows $\{X_f\}$ such that $\{(1+\epsilon)X_f\}$ is supportable for some $\epsilon > 0$, under the self-regulated MaxWeight scheduling, there exists a constant $\tilde{C}$ such that the lengths of the per-flow queues are upper-bounded by $\tilde{C}$ for all t.

Proof: First consider the source node S_f of flow f. Suppose n_f^2 is the second hop on the route of flow f. For node S_f, we have the rate estimate

$$\tilde{X}_f^{S_f}(t) = \left(1+\min\left\{\gamma, \frac{1}{Q_{(S_f,n_f^2)}(t)}\right\}\right)\frac{A_f(t)}{t}.$$

From property (1), we know $(A_f(t)/t) \to X_f$ as $t\to\infty$. Furthermore, according to Lemma 1, $Q_{(S_f,n_f^2)}(t) < C$ for all t. Combining these two facts, we can always find t_s such that when $t > t_s$, $\tilde{X}_f^{S_f}(t) > (1+(1/2C))X_f$.

We define a "super time-slot" for node S_f, which consists of M_s time-slots, where M_s is a positive number. We index the super time-slot using T. Denote by $a_f^{S_f}(T)$ the number of arrivals of flow f at node S_f during super time-slot T. Due to (1), $(a_f^{S_f}(T)/M_s) \to X_f$ when $M_s\to\infty$. In other words, for a constant η_s such that $0<\eta_s<(X_f/2C)$, we can always find an M_s such that $(a_f^{S_f}(T)/M_s) < X_f+\eta_s$. In other words

$$a_f^{S_f}(T) < X_f M_s + \eta_s M_s.$$

On the other hand, for $M_sT > t_s$, during super time-slot $T+1$, $\tilde{X}_f^{S_f}(t) > (1+(1/2C))X_f$. In other words, the aggregated service rate of $Q_f^{S_f}$ during $T+1$ is at least $(1+(1/2C))M_sX_f$. Since $\eta_s < (X_f/2C)$, we have

$$a_f^{S_f}(T) < X_f M_s + \eta_s M_s < \left(1+\frac{1}{2C}\right)M_sX_f.$$

From that above analysis, we can see that the packets arriving at node S_f during super time-slot T can be completely served during super time-slot $T+1$, if $M_sT > t_s$. Since the number of arrivals to node S_f before time t_s is at most $X^{\max}t_s$, it is easy to see that there exists a constant C_s such that $Q_f^{S_f}(t) < C_s$ for all t.

Next, we use induction to prove all per-flow queues are bounded by a constant. Denote by n_f^i the ith node on the route of flow f. Now assume that

$$Q_f^{n_f^j}(t) \le C_i \text{ for all } t \text{ and } j \le i. \quad \text{(induction assumption)}$$

We next define

$$C_\delta = i(C_i + C).$$

First consider the departures of the per-flow queue $Q_f^{n_f^{i+1}}$ at node n_f^{i+1}. It is easy to see that

$$\sum_{\tau=1}^{t} a_f^{S_f}(\tau) \ge \sum_{\tau=1}^{t} a_f^{n_f^{i+1}}(\tau) \ge \sum_{\tau=1}^{t} a_f^{S_f}(\tau) - C_\delta$$

because C_δ is an upper bound on the number of packets belonging to flow f and queued at nodes n_f^1 to node n_f^i (the upstreaming nodes of node $i+1$). Hence according to (1), we have

$$\frac{\sum_{\tau=1}^{t} a_f^{n_f^{i+1}}(\tau)}{t} \to X_f$$

as $t\to\infty$. Moreover, based on Lemma 1, $Q_{(n_f^{i+1},n_f^{i+2})}(t) < C$ for any t. Therefore, there exists some t_i, such that when $t > t_i$

$$\tilde{X}_f^{n_f^{i+1}}(t) > \left(1+\frac{1}{2C}\right)X_f.$$

Next, consider the arrivals of the per-flow queue $Q_f^{n_f^{i+1}}$ at node n_f^{i+1}. Similarly, we also define a "super time-slot" for node n_f^{i+1}, and each super time-slot consists of M_i time-slots, where M_i is a positive number. We index the super time-slots using T. According to (1), for any $\eta_i > 0$, there exists some large enough M_i such that the total number of arrivals at source node S_f during super time-slot T satisfies

$$a_f^{S_f}(T) \le M_i(X_f+\eta_i).$$

Thus, the total number of arrivals at node n_f^{i+1} during super time-slot T satisfies

$$a_f^{n_f^{i+1}}(T) \le M_i(X_f+\eta_i) + C_\delta.$$

Now we consider the dynamic of $Q_f^{n_f^{i+1}}$. We select large enough M_i and small enough $\eta_i > 0$ such that $M_i((1/2C)X_f - \eta_i) > C_\delta$. For the arrival part

$$a_f^{n_f^{i+1}}(T) \le M_i(X_f + \eta_i) + C_\delta.$$

For $M_i T > t_i$, during super time-slot $T+1$, the rate estimate

$$\tilde{X}_f^{n_f^{i+1}}(t) > \left(1 + \frac{1}{2C}\right) X_f.$$

In other words, the aggregated service rate of $Q_f^{n_f^{i+1}}$ during $T+1$ is at least $(1 + (1/2C))M_i X_f$. Thus, we have

$$a_f^{n_f^{i+1}}(T) \le M_i(X_f + \eta_i) + C_\delta < \left(1 + \frac{1}{2C}\right) M_i X_f.$$

From the above analysis, we know the available service rate for the per-flow queue $Q_f^{n_f^{i+1}}$ during super time-slot $T+1$ is always greater than the arrivals at super time-slot T if $M_i T > t_i$. In other words, under the proposed scheme, the arrival packets at super time-slot T can always be completely served by the end of super time-slot $T+1$. Since the arrival packets to $Q_f^{n_f^{i+1}}$ before time t_i are at most $X^{\max} t_i$, there exists a C_{i+1} value such that $Q_f^{n_f^{i+1}}(t) \le C_{i+1}$ for all t. Then, the lemma follows from the induction principle. □

Based on Lemmas 1 and 2, we directly have the following theorem.

Theorem 3: Given a set of flows with arrival rates $\{X_f\}$ such that $\{(1+\epsilon)X_f\}$ is supportable for some $\epsilon > 0$, all queues under the self-regulated MaxWeight algorithm are bounded.

V. Simulations

In this section, we use simulations to evaluate the performance of the self-regulated MaxWeight algorithm, compared to the well-known back-pressure algorithm.

A. Simulation Settings

In this section, we introduce the simulation settings for Sections V-B and V-C. The network we studied in the simulations is shown in Fig. 3, which is a grid network with 8×8 nodes and some random links. The network topology and node indexes are shown in the figure.

All links are bidirectional links, and each link has capacity 1, i.e., in each time-slot, each link can transmit at most one packet. There are 12 flows in the network, and each of the flows is associated with a fixed route. The routes are shown in Table I. From the table, we can see multiple links are used by 2 or 3 flows. In the simulations, we varied the arrival rates of flows to evaluate the performance of the two algorithms under different traffic loads. Each simulation was executed for 100 000 time-slots. Note that in the self-regulated MaxWeight algorithm, γ is set to be 500.

We studied two performance metrics, the total queue length in the network and the average end-to-end delay. Reference [5] points out that under the back-pressure algorithm, the sum of the queue lengths along a route increases quadratically with the route length, which leads to a poor delay performance. We were interested in whether this phenomenon occurs under the self-regulated MaxWeight algorithm.

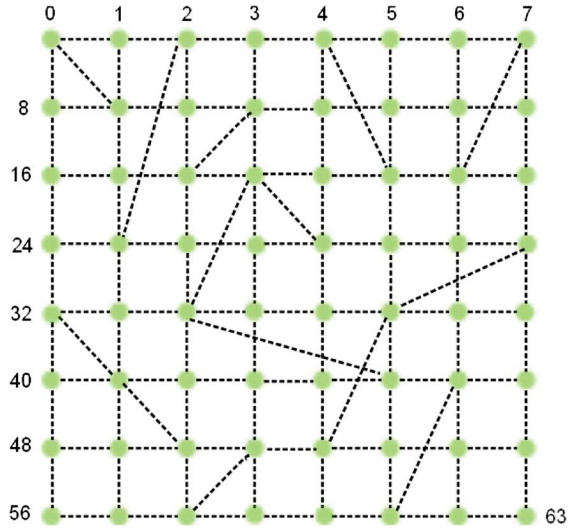

Fig. 3. Network topology.

TABLE I
Flows in the Network

Flow ID	Route
0	$8 \to 9 \to 10 \to 11 \to 12 \to 13 \to 14 \to 6$
1	$17 \to 18 \to 19 \to 28 \to 29 \to 30$
2	$38 \to 37 \to 36 \to 35 \to 34 \to 33 \to 32$
3	$46 \to 45 \to 53 \to 52 \to 60 \to 59 \to 58$
4	$17 \to 16 \to 24 \to 32 \to 41$
5	$19 \to 18 \to 26 \to 34 \to 33 \to 32 \to 40$
6	$43 \to 44 \to 36 \to 28 \to 29 \to 30 \to 22$
7	$55 \to 47 \to 39 \to 31 \to 23 \to 22$
8	$37 \to 45 \to 53 \to 52 \to 60 \to 59 \to 51$
9	$19 \to 27 \to 35 \to 34 \to 33 \to 32$
10	$26 \to 18 \to 19 \to 28 \to 29 \to 30 \to 22$
11	$47 \to 39 \to 38 \to 30 \to 22$

B. Case Where Q-CSMA Is Used for Scheduling

As we mentioned in previous sections, if we combine the self-regulated MaxWeight algorithm with CSMA-based scheduling scheme, the information exchange in the network will be eliminated. In this section, we simulated the case where Q-CSMA is used as the scheduling scheme.

For the interference model, we assume at each time-slot, each node is allowed to transmit to one of its neighbors at most. We further assume the half-duplex interference constraint. In other words, when a link is active, the following three conditions must be satisfied: 1) the receiver of the link is not transmitting; 2) any neighbor of the receiver is not transmitting; 3) the neighbors of the transmitter of the link may be transmitting simultaneously, but cannot take the transmitter as the destination. To schedule the links without violating the interference constraints, we used Q-CSMA as a distributed scheduling scheme [10]–[12]. Fixing the queue lengths, Q-CSMA is a distributed algorithm of finding the MaxWeight schedule. Under Q-CSMA, each node adapts its scheduling decision based on its own queue length and the transmission status of

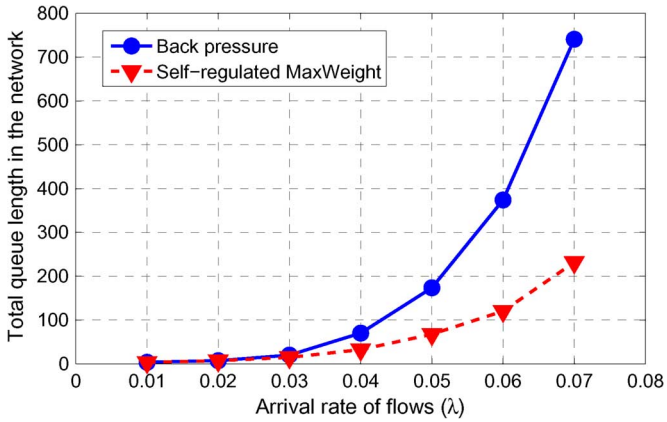

Fig. 4. Total queue lengths in the network under constant arrivals.

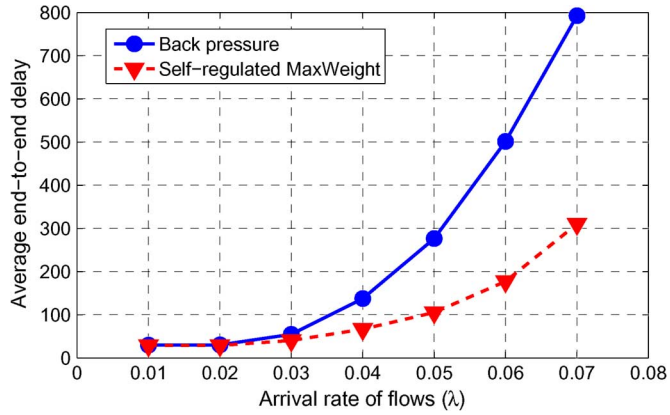

Fig. 5. Average end-to-end delay under constant arrivals.

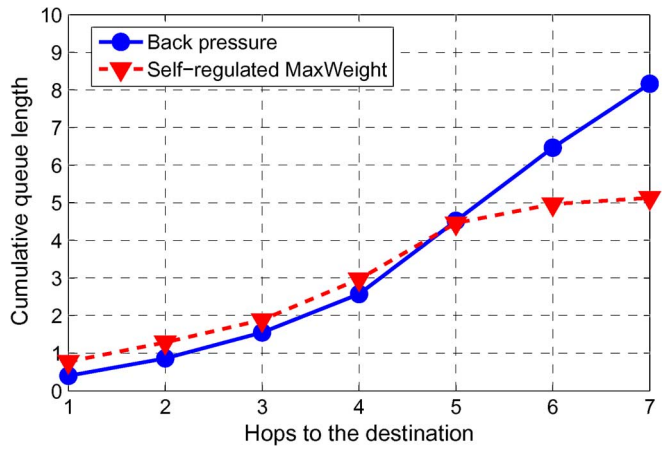

Fig. 6. Cumulative queue lengths versus hops under constant arrivals.

its interfering neighbors, which are learned via carrier sensing. It has been proved that the schedules used under Q-CSMA form a Markov chain, and at the steady state, the Markov chain converges to the MaxWeight schedule. Note the self-regulated MaxWeight algorithm eliminates the communication overhead of exchanging queue lengths among neighboring nodes, and Q-CSMA eliminates the communication overhead of exchanging scheduling decisions among neighboring nodes. Therefore, the joint self-regulated MaxWeight and Q-CSMA completely eliminates the communication overhead and is a fully distributed scheduling algorithm for wireless networks with multihop traffic flows.

We assume the 12 flows have the same packet arrival rate, denoted by λ, i.e., $X_f = \lambda$ for all f. We varied λ from 0.01 to 0.07 to represent different traffic loads.

1) Case of Constant Arrivals: First, we considered the case of constant arrivals. In other words, $X_f(t) = \lambda$ for all t. Note that under this constant arrival rate, property (1) is satisfied.

The overall queue lengths in the network are shown in Fig. 4. As we can observe, when the traffic loads are low, i.e., $\lambda \leq 0.03$, the total queue lengths under both algorithms are similar. However, under medium or high traffic loads, the back-pressure algorithm performs much worse than the self-regulated MaxWeight algorithm.

The average end-to-end delays are shown in Fig. 5. The average end-to-end delay is defined to be the average amount of time needed to transmit a packet from the source to the destination. From the figure, we can see that, similar to the total queue lengths, two algorithms have similar performance under low traffic loads, but with medium or high traffic loads, the self-regulated MaxWeight algorithm outperforms the back-pressure algorithm significantly.

Fig. 6 depicts the cumulative queue lengths versus hops to the destination. On the graph, the x-axis denotes the distance to the destination, and the y-axis is the cumulative queue length from the current node to the destination. We randomly picked a flow that is flow 0 for this simulation and observed its queue lengths under high traffic load, i.e., $\lambda = 0.07$. Under the back-pressure algorithm, the cumulative queue length increases almost quadratically as n increases (e.g., the cumulative queue length at the sixth hop is 6.5, which is close to four times of the cumulative queue length at the third hop, which is 1.6), which is consistent with the observation in [5]. For the self-regulated MaxWeight algorithm, the cumulative queue length increases almost linearly as n. This is what we expected because the algorithm does not need to build up positive "pressure" to pushed the packets to their destinations under the self-regulated MaxWeight algorithm. This is the reason our self-regulated MaxWeight algorithm outperforms the back-pressure algorithm in terms of queue lengths and delays.

2) Case of Poisson Arrivals: Next, we consider the case of Poisson arrivals. We assume the number of arriving packets of each flow f follows a Poisson distribution with expectation λ.

The total queue lengths are shown in Fig. 7. We can observe from the curves that our proposed algorithm has a much better performance than the well-known back-pressure algorithm in terms of total queue lengths, especially in the heavy traffic regime. The reason is that under our algorithm, the network does not need to build up positive queue difference to transmit packets down to the destination, so the number of packets queued in the network is less than that of the back-pressure algorithm.

Furthermore, Fig. 8 shows that the self-regulated MaxWeight algorithm results in a smaller end-to-end delay compared to the back-pressure algorithm. This is a natural result following directly from the situation of total queue lengths and Little's law.

Similar to the case of constant arrivals, Fig. 9 illustrates that the queue length accumulates quadratically from the destination

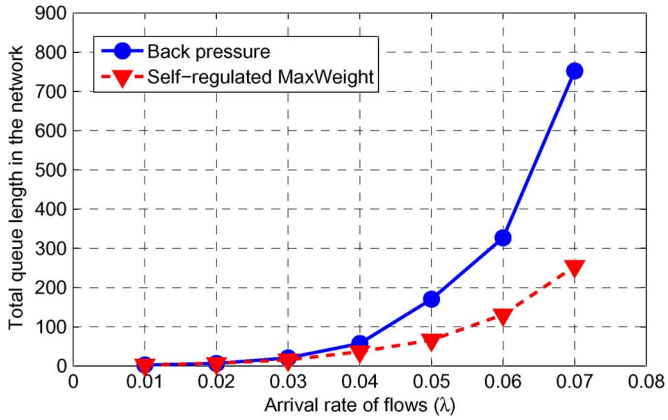

Fig. 7. Total queue lengths in the network under Poisson arrivals.

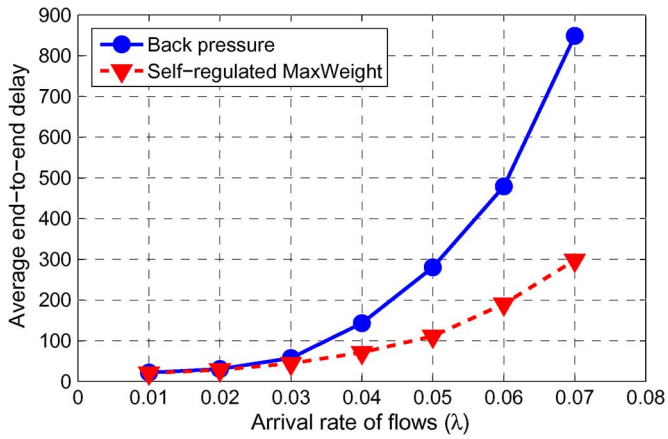

Fig. 8. Average end-to-end delay under Poisson arrivals.

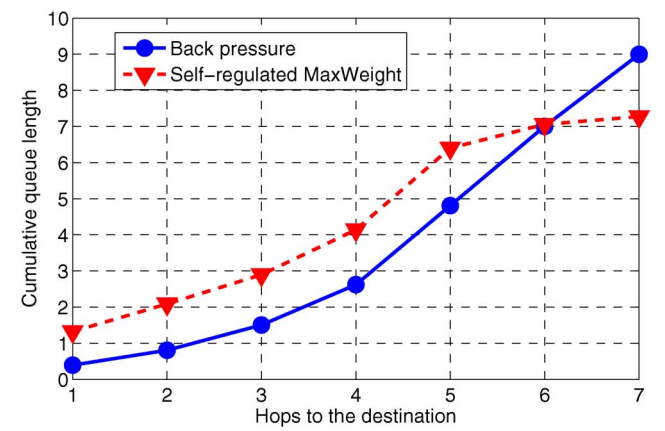

Fig. 9. Cumulative queue lengths versus hops under Poisson arrivals.

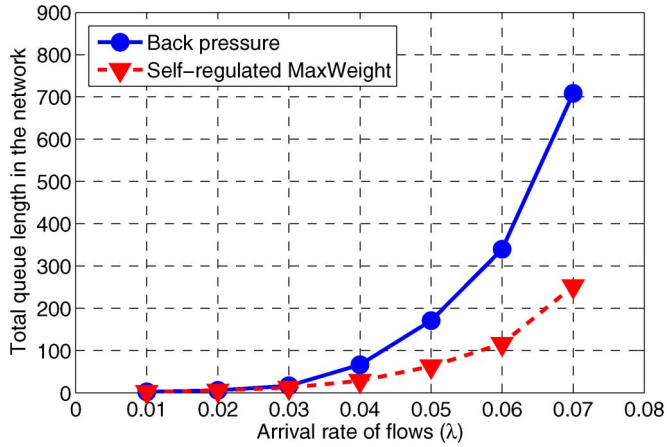

Fig. 10. Total queue lengths in the network under Pareto arrivals.

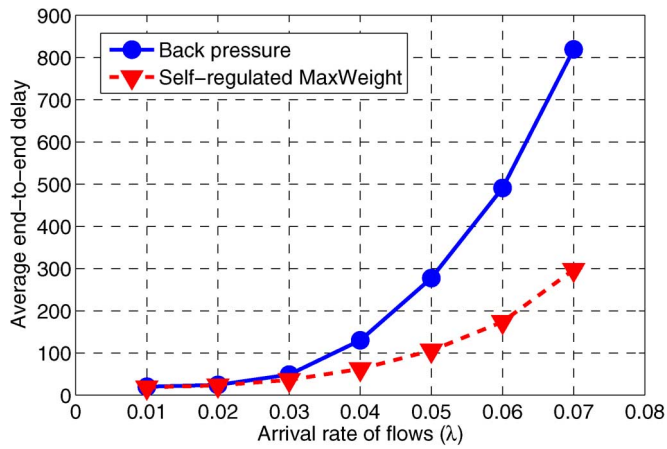

Fig. 11. Average end-to-end delay under Pareto arrivals.

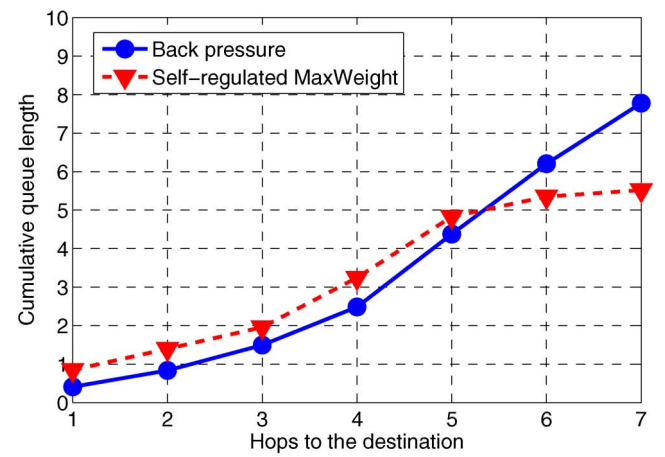

Fig. 12. Cumulative queue lengths versus hops under Pareto arrivals.

to the source under the back-pressure algorithm, while under the self-regulated MaxWeight algorithm, the cumulative queue length increases linearly as n.

3) Case of Pareto Arrivals: We further studied the scenario where the arrivals follow a Pareto distribution with scale parameter x_m and shape parameter α. Note that here the arrival rate $X_f = \alpha x_m/\alpha - 1$. We fixed $x_m = (3/5)X_f$, and correspondingly, we have $\alpha = X_f/(X_f/x_m) = 2.5$.

The total queued packets, average end-to-end delays, and cumulative queue lengths along some routes are illustrated in Figs. 10–12, respectively. From them, we can see that even with Pareto arrivals, we can still observe similar phenomena as in

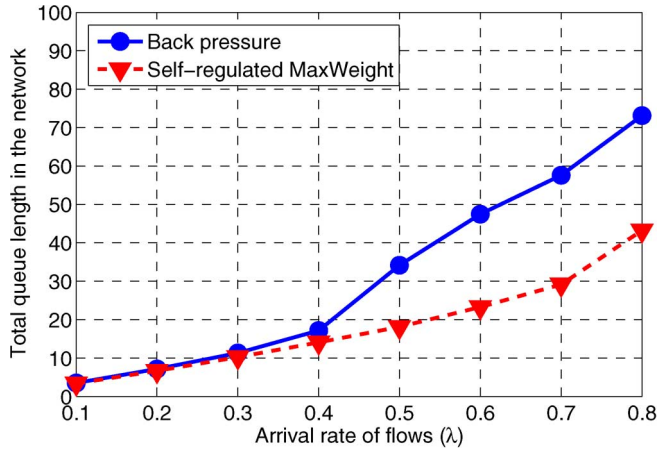

Fig. 13. Total queue lengths in the network under Poisson arrivals.

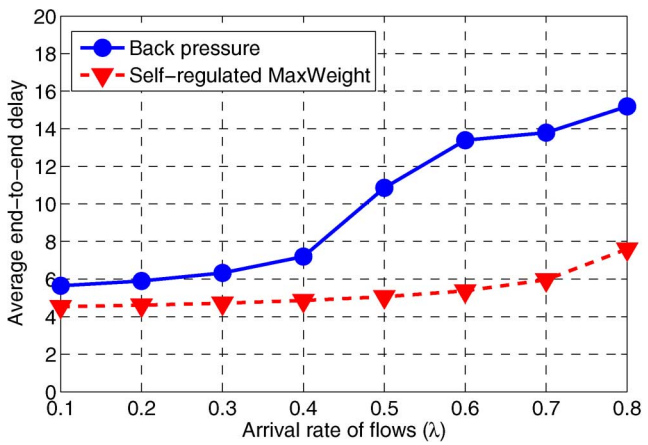

Fig. 14. Average end-to-end delay under Poisson arrivals.

the case of Poisson arrivals. In other words, the self-regulated MaxWeight algorithm still has much better performances than the back-pressure algorithm.

C. Case of Interference-Free Wireless Networks

In this section, we compare the performance of our algorithm to the original back-pressure algorithm without Q-CSMA. Note that, in general, finding the max-weight independent set in a general interference graph is an NP-hard problem. Therefore, we simulated a wireless network where we assume neighboring links use orthogonal frequencies so they do not interfere with each other.

It is easy to see that in this network, the aggregated load on each link should not exceed one since the link capacity is one. We used this criteria to choose flow arrival rates to guarantee the traffic is inside of the capacity region. In particular, for flows 0 and 4, we chose $X_f = \lambda$; for flows 3, 7, 8, and 11, we chose $X_f = \lambda/2$; and for all other flows, we chose $X_f = \lambda/3$. We varied λ from 0.1 to 0.8 to evaluate the performance under different traffic loads. The arrivals follow Poisson distributions.

Figs. 13 and 14 illustrate the total queue lengths and average end-to-end delay under the two algorithms. From the figures, we can see that the self-regulated MaxWeight algorithm still significantly outperforms the back-pressure algorithm. This is because there is no need to build up positive queue differences to push the packets to their destinations under our algorithm.

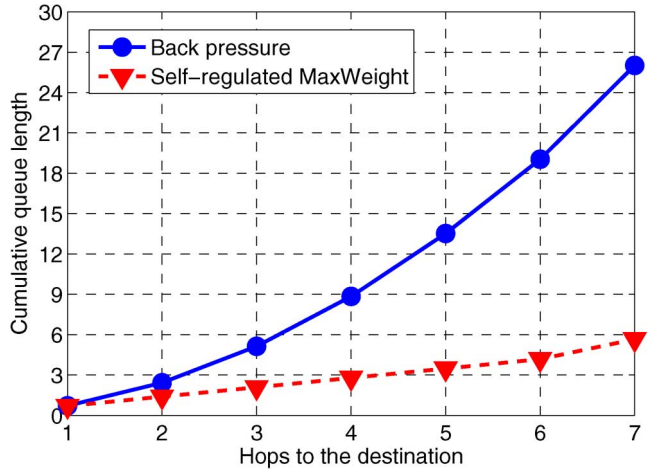

Fig. 15. Cumulative queue lengths versus hops under Poisson arrivals.

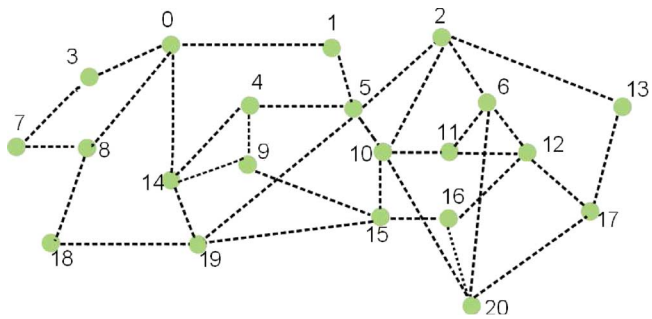

Fig. 16. Random network 1.

D. Random Networks With Random Flows

In the previous simulations, we compared the performance of our algorithm and the back-pressure algorithm in a grid network. In this section, we further validate the performance our algorithm in three networks that are randomly generated and have random flows. We used Q-CSMA as the scheduling scheme in the simulations and the same interference model as in Section V-B. Flow arrivals follow a Poisson distribution. In the self-regulated MaxWeight algorithm, γ is set to be 500. Each simulation was executed for 100 000 iterations. For each of the these three random networks, we compared the performance of the self-regulated MaxWeight algorithm and the original MaxWeight algorithm and evaluated both the total queue lengths and the average end-to-end delays.

1) Random Network 1: The first random network is shown in Fig. 16, which consists of 21 nodes. We randomly created six traffic flows in the network, as listed in Table II. The arrival rate of flows 0 and 1 is 0.5λ, the arrival rate of flows 2 and 3 is λ, and the arrival rate of flows 4 and 5 is 2λ. We varied λ from 0.007 to 0.056 in the simulations. The performance comparison is in Figs. 17 and 18.

2) Random Network 2: The topology of the second random network is shown in Fig. 19. It consists of 16 nodes. We created six random traffic flows as listed in Table III. The arrival rate of flows 0 and 1 is chosen to be 0.5λ, the arrival rate of flows 2

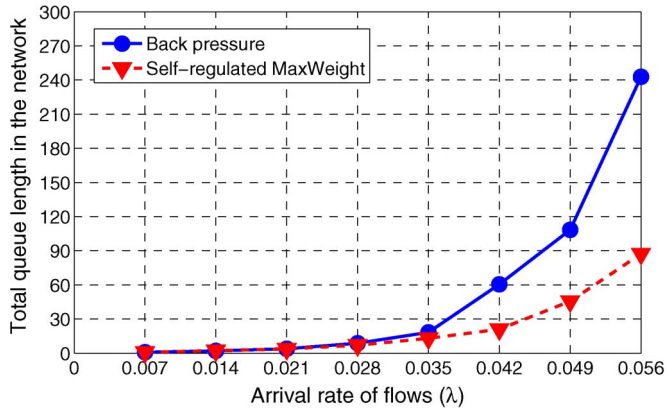

Fig. 17. Total queue lengths in random network 1.

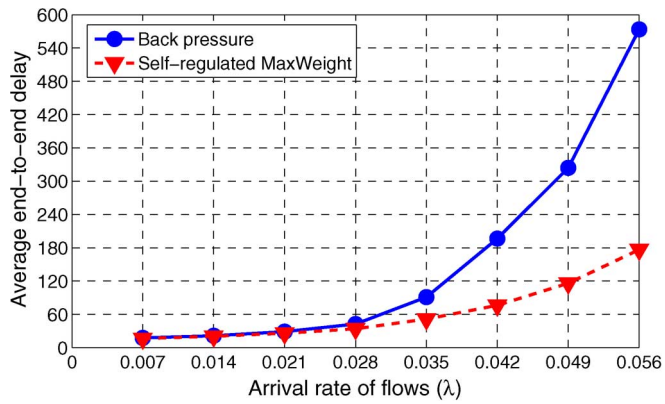

Fig. 18. Average end-to-end delays in random network 1.

TABLE II
FLOWS IN RANDOM NETWORK 1

Flow ID	Route
0	$0 \to 1 \to 5 \to 10 \to 20$
1	$7 \to 8 \to 18 \to 19 \to 5 \to 10 \to 11 \to 12$
2	$13 \to 2 \to 10 \to 15 \to 9 \to 14$
3	$20 \to 6 \to 11 \to 10 \to 5 \to 4 \to 14$
4	$20 \to 16 \to 12 \to 17 \to 13 \to 2$
5	$7 \to 3 \to 0 \to 1$

and 3 is chosen to be λ, and the arrival rate of flows 4 and 5 is chosen to be 2λ. The simulation results are in Figs. 20 and 21.

3) Random Network 3: Fig. 22 shows the topology of the third random network, which contains 18 nodes. The traffic flows are listed in Table IV. The arrival rate of flows 0 and 1 is 2λ, the arrival rate of flows 2 and 3 is λ, and the arrival rate of the rest of the flows is 0.5λ. The simulation results are summarized in Figs. 23 and 24.

As we can see from these simulation results, the self-regulated MaxWeight algorithm significantly outperforms the back-pressure algorithm, especially when traffic load is high.

VI. LIMITATIONS AND DISCUSSION

In this section, we discuss the limitations of our model and possible extensions.

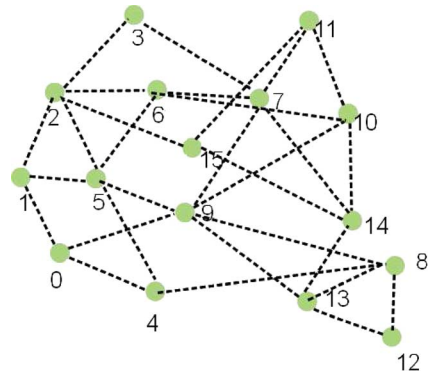

Fig. 19. Random network 2.

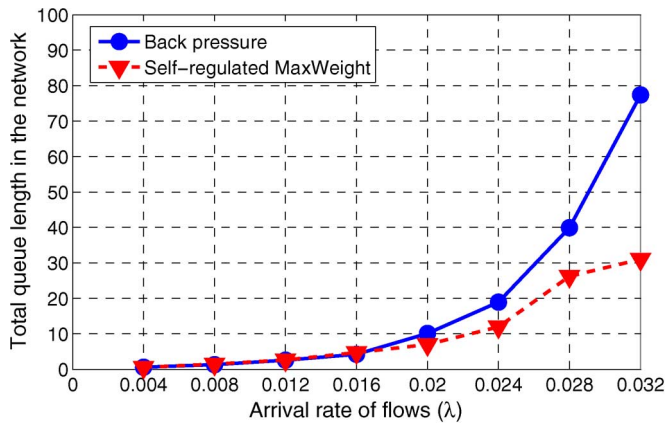

Fig. 20. Total queue lengths in random network 2.

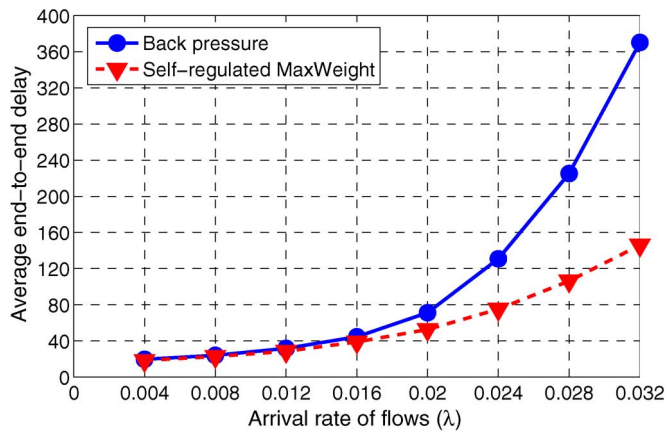

Fig. 21. Average end-to-end delays in random network 2.

TABLE III
FLOWS IN RANDOM NETWORK 2

Flow ID	Route
0	$0 \to 1 \to 2 \to 6 \to 7$
1	$12 \to 13 \to 9 \to 5 \to 6 \to 7$
2	$2 \to 1 \to 5 \to 6 \to 10 \to 11$
3	$13 \to 8 \to 9 \to 0 \to 1$
4	$14 \to 7 \to 9 \to 5 \to 1$
5	$1 \to 2 \to 6 \to 10 \to 14 \to 13$

A. Elastic Flows and Fairness

In this paper, we focused on inelastic flows and the stability of the network. An interesting open problem is to see whether

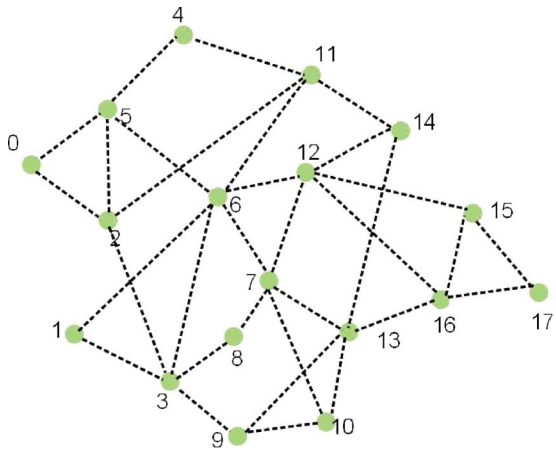

Fig. 22. Random network 3.

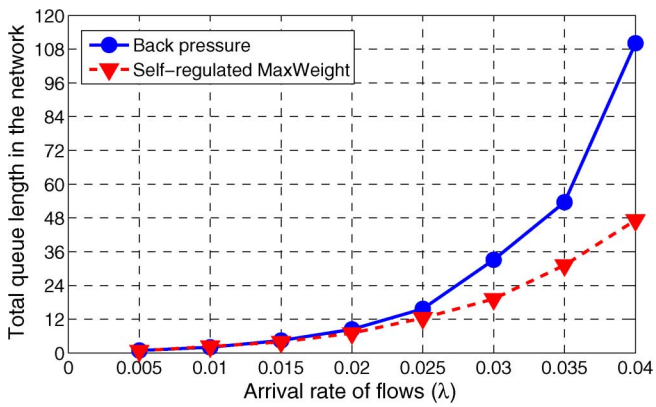

Fig. 23. Total queue lengths in random network 3.

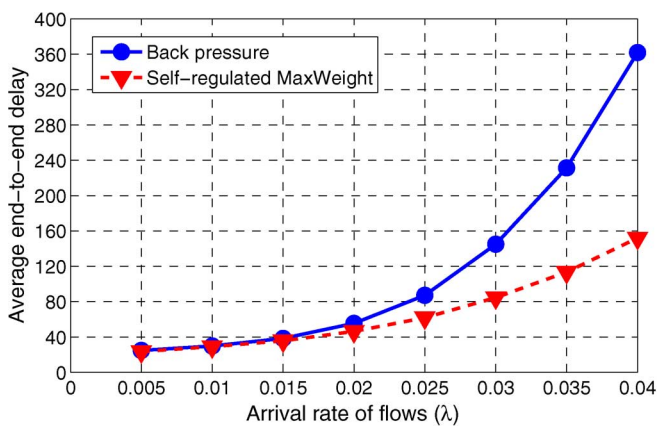

Fig. 24. Average end-to-end delays in random network 3.

TABLE IV
FLOWS IN RANDOM NETWORK 3

Flow ID	Route
0	$0 \to 5 \to 4 \to 11 \to 14 \to 13$
1	$1 \to 3 \to 8 \to 7 \to 12 \to 15 \to 16$
2	$11 \to 2 \to 3 \to 9 \to 13 \to 10$
3	$5 \to 6 \to 7 \to 13 \to 16$
4	$17 \to 16 \to 12 \to 6 \to 5 \to 0$
5	$13 \to 16 \to 12 \to 14 \to 11 \to 4$
6	$9 \to 3 \to 2 \to 0 \to 5 \to 4$

a similar algorithm can be derived for elastic flows. For elastic flows, the aggregated load on each link varies over time and is controlled by the congestion control algorithm. It has been shown in [15]–[17] that joint congestion control and back-pressure routing/scheduling can solve the network utility maximization problem and achieves a fair resource allocation. One of our future research problems is to extend the proposed algorithm to elastic flows. Note that in the back-pressure algorithm, the queue at a source node reflects the congestion level of the network due to the back-pressure nature of the algorithm. Therefore, the source rate should be chosen to be inversely proportional to the queue length at the source. Under our algorithm, the queue at the source does not reflect the network congestion, so new congestion control algorithms need to be derived.

B. Routing

Our model assumes a single fixed route for each flow. The celebrated back-pressure algorithm [2] simultaneously tackles routing and scheduling and achieves throughput optimality. While our result shows that it is possible to design a throughput-optimal scheduling algorithm without exchanging queue length information among neighboring nodes, it is yet clear whether a similar approach works for multipath routing or dynamic routing. In multipath routing or dynamic routing, the aggregated load on a link depends on the routing decisions, so it is not fixed. Therefore, estimating the required link rate locally is not as easy. In [18], the authors proposed an approach to generate routing probabilities based on historical data and then to use this information in the back-pressure routing. The approach, however, still requires queue information exchange for computing the routing probabilities. It remains to be an interesting research problem to see whether the communication overhead can be *completely* eliminated in multipath routing or when routes are not fixed.

C. Traffic

In this paper, we assume that arrivals satisfy condition (1). We conjecture that the algorithm remains to be throughput-optimal for more general traffic types. Our simulation results confirmed that the algorithm performs well with Poisson arrivals and Pareto arrivals. It is one of our future research problems to prove the throughput optimality of the proposed algorithm for general arrivals.

D. Queue Complexity

The two-stage queue architecture requires both per-flow queues and per-link queues at each node. Thus, the number of queues needed at each node can be very large when there are many flows in the network. A recent paper [19] has shown that a similar algorithm with per-hop queues can also achieve throughput optimality.

VII. CONCLUSION

In this paper, we considered the scheduling problem in multihop wireless networks and proposed the self-regulated MaxWeight scheduling that does not require the exchange of queue length information among neighboring nodes, hence completely eliminating the communication overhead when

combined with recent CSMA-based scheduling algorithms. The new algorithm is proved to be throughput-optimal and has a much better performance compared to the well-known back-pressure algorithm.

REFERENCES

[1] S. Liu, E. Ekici, and L. Ying, "Scheduling in multihop wireless networks without back-pressure," in *Proc. Annu. Allerton Conf. Commun., Control, Comput.*, Monticello, IL, USA, 2010, pp. 686–690.

[2] 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.

[3] X. Lin, N. Shroff, and R. Srikant, "A tutorial on cross-layer optimization in wireless networks," *IEEE J. Sel. Areas Commun.*, vol. 24, no. 8, pp. 1452–1463, Aug. 2006.

[4] L. Georgiadis, M. J. Neely, and L. Tassiulas, "Resource allocation and cross-layer control in wireless networks," *Found. Trends Netw.*, vol. 1, no. 1, pp. 1–144, 2006.

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

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

[7] L. Huang and M. Neely, "Delay reduction via Lagrange multipliers in stochastic network optimization," *IEEE Trans. Autom. Control*, vol. 56, no. 4, pp. 842–857, Apr. 2011.

[8] B. Ji, C. Joo, and N. B. Shroff, "Delay-based back-pressure scheduling in multi-hop wireless networks," in *Proc. IEEE INFOCOM*, Shanghai, China, 2011, pp. 2579–2587.

[9] L. Bui, R. Srikant, and A. L. Stolyar, "Optimal resource allocation for multicast flows in multihop wireless networks," *Phil. Trans. Royal Soc.*, vol. 366, no. 1872, pp. 2059–2074, 2008.

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

[11] 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," *IEEE/ACM Trans. Netw.*, vol. 20, no. 3, pp. 825–836, Jun. 2012.

[12] S. Rajagopalan, D. Shah, and J. Shin, "Network adiabatic theorem: An efficient randomized protocol for contention resolution," in *Proc. Annu. ACM SIGMETRICS Conf.*, 2009, pp. 133–144.

[13] C. Humes, "A regulator stabilization technique: Kumar-seidman revisited," *IEEE Trans. Autom. Control*, vol. 39, no. 1, pp. 191–196, Jan. 1994.

[14] X. Wu and R. Srikant, "Regulated maximal matching: A distributed scheduling algorithm for multi-hop wireless networks with node-exclusive spectrum sharing," in *Proc. IEEE CDC*, Seville, Spain, 2005, pp. 5342–5347.

[15] A. Stolyar, "Maximizing queueing network utility subject to stability: Greedy primal-dual algorithm," *Queueing Syst.*, vol. 50, no. 4, pp. 401–457, Aug. 2005.

[16] A. Eryilmaz and R. Srikant, "Fair resource allocation in wireless networks using queue-length-based scheduling and congestion control," *IEEE/ACM Trans. Netw.*, vol. 15, no. 6, pp. 1333–1344, Dec. 2007.

[17] M. J. Neely, E. Modiano, and C.-P. Li, "Fairness and optimal stochastic control for heterogeneous networks," *IEEE/ACM Trans. Netw.*, vol. 16, no. 2, pp. 396–409, Apr. 2008.

[18] E. Athanasopoulou, L. Bui, T. Ji, R. Srikant, and A. Stoylar, "Back-pressure-based packet-by-packet adaptive routing in communication networks," *IEEE/ACM Trans. Netw.*, vol. 21, no. 1, pp. 224–257, Feb. 2013.

[19] B. Ji, C. Joo, and N. B. Shroff, "Throughput-optimal scheduling in multihop wireless networks without per-flow information," *IEEE/ACM Trans. Netw.*, vol. 21, no. 2, pp. 634–647, Apr. 2013.

Shihuan Liu (S'10–M'13) received the B.E. degree in electronics engineering from Tsinghua University, Beijing, China, in 2008, and the Ph.D. degree in electrical and computer engineering from Iowa State University, Ames, IA, USA, in 2012.

His research interest is in the area of resource allocation in wireless networks.

Dr. Liu won the Research Excellence Award from Iowa State University in 2012.

Eylem Ekici (S'99–M'02–SM'11) received the B.S. and M.S. degrees in computer engineering from Boazii 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 cognitive radio networks, vehicular communication systems, and nano communication systems with a focus on modeling, optimization, 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 *Mobile Computing and Communications Review*.

Lei Ying (M'08) received the B.E. degree from Tsinghua University, Beijing, China, in 2001, and the M.S. and Ph.D. degrees in electrical and computer engineering from the University of Illinois at Urbana–Champaign, Urbana, IL, USA, in 2003 and 2007, respectively.

He currently is an Associate Professor with the School of Electrical, Computer and Energy Engineering, Arizona State University, Tempe, AZ, USA. He was the Northrop Grumman Assistant Professor with the Department of Electrical and Computer Engineering, Iowa State University, Ames, IA, USA, from 2010 to 2012. His research interest is broadly in the area of stochastic networks, including big data and cloud computing, P2P networks, social networks, and wireless networks.

Dr. Ying is an Associate Editor of the IEEE/ACM TRANSACTIONS ON NETWORKING. He won the Young Investigator Award from the Defense Threat Reduction Agency (DTRA) in 2009 and the NSF CAREER Award in 2010.