Achieving Optimal Throughput Utility and Low Delay with CSMA-like Algorithms: A Virtual Multi-Channel Approach

Po-Kai Huang, *Student Member, IEEE,* and Xiaojun Lin, *Senior Member, IEEE*

***Abstract*—CSMA algorithms have recently received significant interests in the literature for designing wireless control algorithms. CSMA algorithms incur low complexity and can achieve the optimal capacity under certain assumptions. However, CSMA algorithms suffer the starvation problem and incur large delay that may grow exponentially with the network size. In this paper, our goal is to develop a new algorithm that can provably achieve high throughput utility and low delay with low complexity. Towards this end, we propose a new CSMA-like algorithm, called Virtual-Multi-Channel (VMC-) CSMA, that can dramatically reduce delay. The key idea of VMC-CSMA to avoid the starvation problem is to use multiple virtual channels (which emulate a multi-channel system) and compute a good set of feasible schedules simultaneously (without constantly switching/re-computing schedules). Under the protocol interference model and a single-hop utility-maximization setting, VMC-CSMA can approach arbitrarily close-to-optimal system utility with both the number of virtual channels and the computation complexity increasing logarithmically with the network size. Further, once VMC-CSMA converges to the steady-state, we can show that under certain assumptions on the utility functions and the topology, both the expected packet delay and the tail distribution of the HOL (head-of-line) waiting time at each link can be bounded independently of the network size. Our simulation results confirm that VMC-CSMA algorithms indeed achieve both high throughput utility and low delay with low-complexity operations.**

***Index Terms*—Carrier sence multiple access (CSMA), utility maximization, distributed scheduling, Markov chain, starvation, virtual multiple channels.**

I. Introduction

A central problem to the design of wireless control algorithms is how to schedule the link transmissions in the presence of interference. Among the many design goals for wireless scheduling, three of them are perhaps the most important. First, in order to support the increasing amount of the traffic placed on wireless networks, the control algorithm should achieve high capacity. Second, the packet delay should be small to meet the applications' service requirements. Third, for large networks, the control algorithms should have low computational complexity and low communication overhead, and preferably can be implemented in a distributed manner.

Existing algorithms in the literature achieve different tradeoffs among capacity, delay, and complexity. It is well known that max-weight algorithms [2] can attain the largest capacity region of the network, based on which many wireless cross-layer control algorithms have been developed to optimize various performance objectives such as the system throughput, utility/fairness, and power efficiency [3,4]. However, for many problem settings, the max-weight algorithm incurs exponential complexity as the network size increases. Hence, they are infeasible even for medium network size. Further, it is a centralized algorithm that requires global information. There are a number of low-complexity algorithms that can be viewed as approximations to the max-weight algorithm (see [3] and the reference within). However, these algorithms can only guarantee a fraction of the optimal system capacity. Recently, a class of CSMA (Carrier Sense Multiple Access) algorithms were studied in [5,6]. CSMA algorithms are attractive because they are fully distributed and have $\mathcal{O}(1)$ complexity that is independent of the network size. Further, CSMA algorithms can be shown to achieve the optimal capacity under certain assumptions. However, CSMA algorithms have been observed to have large delay that may grow exponentially with the network size [7,8], which makes the usefulness of the capacity gain questionable because most applications (even as simple as web-browsing) require some level of low delay.

In summary, these existing algorithms have been unable to achieve all three goals, i.e., high throughput, low delay and low complexity at the same time. This unsatisfactory state-of-art has led to the conjecture that perhaps there is a fundamental tradeoff among these three dimensions. For example, in an elegant result in [9], the authors show that there exist worst-case topologies such that even to achieve a diminishingly small fraction of the optimal throughput capacity, either the complexity or the delay must grow exponentially with the network size. Based on this impossibility result, it may seem an unattainable endeavor to dramatically reduce the delay of CSMA algorithms without sacrificing their high throughput or low complexity. However, while this impossibility result is theoretically intriguing, it does not explain why for "easier" topologies, one cannot dramatically reduce the delay of CSMA-like algorithms. In fact, for topologies with bounded degree, [10,11] have shown that nearly-optimal throughput can indeed be achieved with complexity and delay that do not grow with the network size. The algorithms in [10,11] are still quite complex. Nonetheless, they do suggest the possibility that for a certain class of network topologies, there may be potential to address the delay problem of CSMA-like algorithms, which may then lead to a low-complexity and distributed algorithm with both provably high throughput and provably low delay.

This work has been partially supported by the National Science Foundation through grant CNS-0643145, CNS-0721484, and CNS-0831999. Earlier version of this paper has appeared in IEEE INFOCOM'13 [1].

Po-Kai Huang and Xiaojun Lin are with the School of ECE, Purdue University (Email: huang113@purdue.edu, linx@purdue.edu).

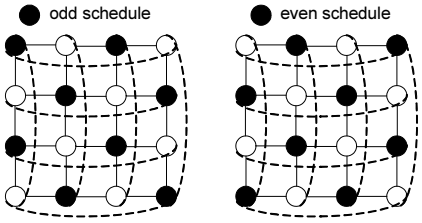


Fig. 1. A torus interference graph with the odd and even schedules.

Unfortunately, improving the delay performance of CSMA remains a challenging problem in spite of a number of recent works [6,8,9,11]–[17]. The main difficulty of reducing the delay of CSMA may be explained by the following example. Consider a n-by-n torus interference graph as shown in Fig. 1, where each circle represents a unit-capacity link, and the edge between two circles means that these two links can not transmit simultaneously. Suppose that the target rates of all links are 0.5. In a typical discrete-time version of the standard CSMA algorithms [6], it is possible to adjust the parameters so that the CSMA algorithms will stay at either the even or the odd schedule with probability close to 0.5 to attain the target rates. However, once CSMA finds either the even or the odd schedule, it will be "locked" to this schedule for a long time before it can switch to the other schedule [8], and the corresponding inactive links will be starved of service in this period. This problem is known as the "starvation" problem for CSMA. Hence, it is difficult for CSMA to attain low delay.

In this paper, we propose a new algorithm, called Virtual-Multi-Channel (VMC-) CSMA, that can achieve both provably high throughput utility and provably low delay with low complexity. The main novelty of our proposed algorithm to significantly reduce delay is to take advantage of multiple (physical- or virtual-) channels. This main idea can be explained as follows. Suppose that there are two separate channels for the same topology in Fig. 1, and each has half the bandwidth of the original system. If one channel uses the odd schedule, and the other channel uses the even schedule, then each link can achieve both the target rate of 0.5 as well as low delay (because each link is served at a constant rate at all time-slots). Note that the key idea here is to compute a set of good schedules for multiple channels at once rather than computing one schedule at a time and constantly switching/recomputing schedules. Thus, the starvation problem is avoided. Although this idea seems to be quite natural, there are three main difficulties to generalize it to arbitrary networks. First, in many systems, we may only have one physical channel. How can we still reduce delay by using multiple channels? Second, even if we have more than one channel, how can we design a low-complexity and distributed algorithm to compute the right multi-channel schedules? Third, the number of channels in the above example is two, but it may not be sufficient for general topologies. Will the number of channels required to approach near-optimal performance be exceedingly large such that the complexity increases to a prohibitive level?

Our proposed VMC-CSMA algorithms precisely address these difficulties. First, for systems with only one physical channel, we introduce the notion of "virtual channels." Specifically, there are C virtual channels, and each virtual channel can have a different schedule. By randomly choosing a virtual channel and using the corresponding schedule at each time slot, we can then emulate the behavior of multiple physical channels. (See Section III-B for the corresponding distributed implementation.) Second, assuming that each link ℓ has a utility function $U_\ell(R_\ell)$ of its rate R_ℓ, our proposed VMC-CSMA algorithms iteratively update the schedules across all virtual channels to optimize the total system utility. In each iteration, VMC-CSMA only requires local information exchange and incurs a low complexity that increases linearly with the number of virtual-channels C. Third, we rigorously quantify the throughput and delay performance of the VMC-CSMA algorithms with respect to C. Specifically, for an arbitrary network topology, let L denote the total number of links. We show that when $\epsilon \leq 0.1$ and $C \geq \frac{2\log L}{3\epsilon^2}$, VMC-CSMA algorithms can allocate an expected rate vector $\vec{R} = [R_\ell]$ to each link such that $\sum_{\ell=1}^{L} U_\ell(R_\ell) \geq (1-\epsilon)\sum_{\ell=1}^{L} U_\ell([R_\ell^* - \epsilon]^+)$, where $\vec{R}^* = [R_\ell^*]$ is the rate vector with maximum system utility. Thus, VMC-CSMA algorithms can achieve close-to-optimal system utility with the number of channels (and hence the corresponding complexity) increasing very slowly ($\mathcal{O}(\log L)$) with the network size. For delay, we show that, once VMC-CSMA algorithms converge to the steady-state, the expected packet delay of link ℓ equals to $1/R_\ell$, and the tail distribution of its head-of-line (HOL) waiting time can be asymptotically bounded (see Lemma 5 for details). More importantly, we show that under the assumption of logarithmic utility functions, as long as some sufficient conditions for the network topology and utility functions are satisfied, the delay of each link will not grow with the network size. Our simulation results confirm that the proposed VMC-CSMA algorithms indeed achieve both high throughput and low delay. Further, it can quickly adapt to network traffic changes. In summary, the proposed VMC-CSMA algorithm achieves both provable close-to-optimal throughput utility and provable low delay (that does not grow with the network size under suitable conditions) with low-complexity operations.

The standard CSMA algorithm has been extended to multi-channel networks in [18]. However, the delay issue was not discussed. There have been a number of recent studies that try to quantify and improve the delay performance of CSMA algorithms [8,11]–[17]. However, none of them can attain the same level of throughput/delay performance and low complexity as we reported here. The work in [15] compares the delay performance as one tunes the parameter of a class of CSMA algorithms. Similarly, the work in [17] introduces a threshold for each link. A link will immediately relinquish transmission opportunity if the queue length drops below the threshold. However, it is unclear whether such modifications will fundamentally and provably alter the exponential order of the starvation time. In [14], CSMA algorithms with deadline constraints are studied under the setting of a complete graph (where each link interferes with every other link). In another work [8], the authors propose to periodically reset or "unlock" the schedule to an initial empty schedule. They show that for

a torus interference graph like Fig. 1, the delay can be made independent of the network size. However, it seems difficulty to generalize the results from [8,14] to more general network topologies. [12,13] show that if the offered load is sufficiently small (as a function of the maximum degree of the interference graph), both the starvation time and the delay of CSMA can be reduced to $\mathcal{O}(\log L)$ [12] or even $\mathcal{O}(1)$ [13]. However, for such results to hold, the offered load must be reduced significantly from the optimal capacity. Similarly, [16] considers using multiple physical channels for CSMA, but under the constraint that each link can occupy at most one channel. Thus, the resulting capacity could also be far from optimal. Our idea of using multiple channels is also inspired by [19]. However, it is difficult to modify the algorithm in [19] to optimize global system utility. To the best of our knowledge, only the algorithms in [10,11] can achieve similar goals as ours, i.e., both provably high throughput and provably low delay with low complexity. Both [10] and [11] partition the network into non-interfering pieces with finite size. In [10], a max-weight algorithm is used in each partition. However, in order to approach the optimal capacity, the size of each partition must be large. Hence, the max-weight algorithm in each partition requires coordination among nodes that are increasingly far apart. In [11], CSMA is run in each partition. However, if the partition size is large, the actual delay in each partition may still be large. Finally, our results do not contradict with the impossibility result of [9] (discussed earlier) because our sufficient conditions on network topology may exclude the worst-case topology of [9]. Further, both our notion of delay (steady-state delay vs. transient delay) and our system setting (with vs. without congestion control) are different (see the end of Section III-D for detailed discussions).

The rest of the paper is organized as follows. The system model is presented in Section II. In Section III, we present the VMC-CSMA algorithm along with the performance analysis. We discuss the implementation issues in Section IV and the simulation results in Section V. Then we conclude.

II. System Model

Consider a wireless network with N nodes and L links, where each node represents a communication device, and each link corresponds to a pair of transmitting node and receiving node. We assume the so-called protocol interference model, i.e., two links interfere with each other if they can not transmit data at the same time.[1] Let $\mathcal{E}_\ell$ be the set of links that interfere with link ℓ. We assume that the link interference relationship is bilateral, i.e., if $\ell_1 \in \mathcal{E}_{\ell_2}$, then $\ell_2 \in \mathcal{E}_{\ell_1}$. Two links ℓ_1 and ℓ_2 are called neighbors if ℓ_1 and ℓ_2 interfere with each other. We consider a time-slotted system, where each slot has unit length. Assume that the wireless network has only one physical channel. The capacity of each link is assumed to be 1, i.e., each link can transmit at most one unit-sized packet in one time slot. To represent a schedule, we will use a L-dimension vector such that the ℓ^{th} element is 1 if link ℓ is included in the schedule and 0 otherwise.

Associated with each link is a one-hop flow, i.e., packets of the flow will immediately leave the network after it traverses the link. We assume that each flow is infinitely backlogged, i.e., at the transport layer it always has packets to send. Further, each flow at link ℓ has a utility function $U_\ell(\cdot)$ associated with it. If the long-term average rate of link ℓ is R_ℓ, then $U_\ell(R_\ell)$ represents the satisfactory level of the corresponding flow [3]. We assume that each utility function is positive, non-decreasing, strictly concave, and twice differentiable [22]. Further, we assume that it is bounded on the domain $D = [0, 1]$. Let $U(\vec{R}) = \sum_{\ell=1}^{L} U_\ell(R_\ell)$. Let Ω denote the capacity region of the wireless network, which is given by the set of all rate vectors $\vec{R} = [R_\ell]$ such that there exists a control policy that can support the long-term average rate R_ℓ at each link ℓ.

As we have discussed in Section I, we are interested in both high throughput utility and low delay. For high throughput utility, we aim to solve the following optimization problem:

$$\max_{R_\ell \geq 0} \sum_{\ell=1}^{L} U_\ell(R_\ell), \ \ \vec{R} = [R_\ell] \in \Omega. \tag{1}$$

Let $\vec{R}^* = [R_\ell^*]$ be the optimal solution of problem (1). Note that problem (1) is a cross-layer control problem [3] because it involves two control mechanisms. First, the transport layer of the flow at each link ℓ does congestion control and determines how packets can be injected at the long-term average rate R_ℓ^*. Second, at the MAC layer, the system determines how to schedule the link transmissions to support the long-term average rate R_ℓ^* at all links.

Under infinite-backlog setting, the delay of a packet is defined as the difference between the time when the transport layer injects the packet into the buffer and the time when this packet is served by the link. Although this definition seems to be a natural definition of delay, it unfortunately does not fully capture the effect of the possible starvation problem [8]. For example, consider a queue with a single buffer. A new packet is added to the buffer immediately after the old packet is served. Suppose that for every 1000 packets, it takes only one time-slot to serve each of the first 999 packets. However, the 1000^{th} packet suffers starvation for 1000 time-slots. In this case, the expected packet delay (average over all packets) is 1.99. Thus, the effect of the starvation problem is not obvious. Due to this reason, we introduce another notion of delay. Specifically, at each time slot, we study the time that the head-of-line (HOL) packet has waited in the system. In the above example, the expected HOL waiting time across time-slots would be around 250. Hence, the negative impact of the starvation problem is more obvious. In this paper, by "low delay", we mean that both the packet delay and the HOL waiting time should be small. Thus, the goal is to develop low-complexity and distributed algorithms that can provably achieve both high throughput utility and low delay.

Remark: Note that in this paper, we have defined throughput and delay directly based on a utility-maximization setting. This definition may seem somewhat different from many existing

[1]The basic idea of CSMA has been extended to more general interference models, e.g., those based on SINR [20,21], although the algorithms there are likely to suffer similar starvation problem and large delay as in [7,8]. Thus, we expect that the key insights of our work can also be extended to reduce delay under these more general interference models.

works [5,6,8,11]. There, throughput optimality is often defined for a system where packets arrive according to some stochastic process, and the delay is defined as the time from packet arrival to the time when the packet is served at the MAC layer. However, we believe that the infinite-backlog model and the delay definition in this paper are important to study. First, the goal of throughput optimality is to support the largest possible capacity region. In practice, the offered load may fall outside the capacity region, in which case some form of congestion control is required to avoid overloading the system. Further, whenever overload occurs, it is important to maintain some notion of fairness across users. The infinite-backlog utility-maximization formulation is often used to model congestion control and fairness [3,4], and hence is important to study. Second, under such an infinite-backlog model, the delay at the transport layer would have been infinite. Thus, we focus instead on the delay after the packet is injected by the transport layer, which can be interpreted as the service delay at the link's MAC layer. We note that under the traditional CSMA algorithm, this delay remains very large due to the starvation problem (see Fig. 4(a) in Section V). Hence, it is important to study how to reduce such delay. Finally, under certain assumptions, it may be possible to extend the key insights of our work to the setting with packet arrivals. We briefly discuss the main idea and potential challenge in Section III-F.

III. VMC-CSMA Algorithm Design and Analysis

In this section, we propose a new low-complexity Virtual-Multi-Channel (VMC-) CSMA algorithm with provable high throughput utility and provable low delay. Since our algorithm is motivated by the standard CSMA algorithm, we will first briefly describe a discrete time version of the standard CSMA algorithm [3,6] for solving problem (1) and discuss its weakness. We will then introduce the new VMC-CSMA algorithm.

A. The Standard CSMA algorithm

Let $Q_\ell(t)$ be the number of packets of link ℓ at the beginning of time slot t. Let $\vec{S}(t) = [S_\ell(t)]$ be the schedule chosen by the scheduling algorithm at time t. Define a set $\mathcal{S}$ of *decision schedules* such that every element in $\mathcal{S}$ is a feasible schedule. Further, each link must be scheduled by at least one schedule in $\mathcal{S}$. Let $w_\ell(t)$ be an suitable increasing function of $Q_\ell(t)$ as in [6]. For example, $w_\ell(t) = \log(\alpha Q_\ell(t))$, where α is a positive constant. The following CSMA algorithm is known to solve problem (1) under certain assumptions.

CSMA Algorithm: At each time t,

- *Decision Phase:* Choose a decision schedule $\vec{S}^D(t) = [S^D_\ell(t)]$ randomly in $\mathcal{S}$.
- *Scheduling Phase:* For each link ℓ, if $S^D_\ell(t) = 1$ and $S_{\ell'}(t) = 0$ for every link $\ell' \in \mathcal{E}_\ell$, we have: $P(\{S_\ell(t) = x\}) = \frac{\exp(xw_\ell(t))}{1+\exp(w_\ell(t))}$, where $x = 0$ or 1. Otherwise, $S_\ell(t) = S_\ell(t-1)$.
- *Congestion Control:* Each link ℓ injects a random number of packets according to a Poisson distribution with mean $r_\ell = \arg\max_{r\geq 0}\{U_\ell(r) - \beta Q_\ell(t) r\}$, where $\beta > 0$.

It can be shown that under a time-scale separation assumption [6], if β is sufficiently small, $\vec{r} = [r_\ell]$ will converge to values close to the optimal solution of (1) [3]. In practice, the above time-scale separation assumption must be approximated by a small value of α. However, when α and β are small, $Q_\ell(t)$ and $w_\ell(t)$ tend to be large. In that case, the algorithm will likely to be stuck in one schedule for a long time before it switches to another schedule [8] (as we discussed in Section I). In the worst case, the starvation time and the delay may grow exponentially with the network size [7].

B. Virtual-Multi-Channel CSMA Algorithm

We now introduce our proposed VMC-CSMA algorithm that overcomes the difficulties of starvation problem and large delay. As we have mentioned in Section I, the key to achieve both high throughput utility and low delay is to take advantage of C multiple channels and simultaneously compute C feasible schedules over all channels. Once a good set of C feasible schedules is found, we can then avoid constantly switching schedules (and hence the starvation problem). To make this idea feasible in wireless systems with only one physical channel, we introduce the concept of "virtual channels." Specifically, for each link ℓ, there are C virtual channels, and we use $\vec{V}_\ell = [V_{\ell 1}, \cdots, V_{\ell C}]$ to denote its schedule in all virtual channels, where $V_{\ell k} = 1$ if link ℓ is scheduled in the k^{th} virtual channel, and $V_{\ell k} = 0$ otherwise. We use $\vec{V} = [\vec{V}_\ell]$ to denote the global schedules of all virtual channels and all links. When we focus on a specific virtual-channel k, there is a feasible schedule $\vec{S}(\vec{V})_k = [V_{1k}, \cdots, V_{Lk}]$ for the network. Note that given the global schedule $\vec{V}$, the total number of virtual channels used by link ℓ is given by $x_\ell(\vec{V}) = \sum_{k=1}^{C} V_{\ell k}$. To use these schedules, at each time slot t, the network randomly chooses a virtual-channel $k(t)$ uniformly from 1 to C, *i.i.d.* across time-slots. All links in the network then use the feasible schedule $\vec{S}(\vec{V})_{k(t)}$ in this time-slot, i.e., each link ℓ transmits a packet if $V_{\ell,k(t)} = 1$.[2] Note that each link only needs to know its own schedules $\vec{V}_\ell$. Further, the randomization of the virtual-channel $k(t)$ can be achieved in a distributed manner if all links are synchronized, and they have the same random-number generator with a common-seed, which only needs to be agreed upon at the beginning of the system operation. With this implementation, each link ℓ will be scheduled for actual transmission with probability equal to $r_\ell(\vec{V}) = x_\ell(\vec{V})/C$, *i.i.d.* across time-slots. Thus, the long-term average rate of link ℓ is $r_\ell(\vec{V})$, and the average inter-service time is $1/r_\ell(\vec{V})$. Hence, the delay will likely be small as we will show below.

It remains to develop a low-complexity and distributed algorithm for computing the global schedule $\vec{V}$ that leads to high total system utility. Specifically, we seek solution to the following optimization problem, which can be viewed as an approximation to the original optimization problem (1):

$$\begin{aligned} &\max_{\vec{V}} \sum_{\ell=1}^{L} U_\ell(r_\ell(\vec{V})), \\ &\vec{S}(\vec{V})_k \text{ is a feasible schedule, } k = 1, 2, \cdots, C. \end{aligned} \tag{2}$$

[2]We note that the idea of using a random schedule has some similarity to the stationary randomized policy [4] that has been used to analyze the throughput optimality of other algorithms. However, there are no low-complexity methods of computing the right stationary randomized policy.

Our hope is that when C is sufficient large, the optimal solution of (2) will be close to that of (1).

We next describe our proposed low-complexity VMC-CSMA algorithm for solving (2). Later in Section III-C and III-D, we will study its throughput and delay as the number of virtual-channels C varies. Because VMC-CSMA algorithm updates the global schedule $\vec{V}$ iteratively over time, we will use the notation $\vec{V}(t)$, $\vec{V}_\ell(t)$, and $V_{\ell k}(t)$ to denote their corresponding values at time slot t. Similar to the standard CSMA algorithm, we define a set $\mathcal{S}$ of decision schedules such that it satisfies the following three conditions: $(C1)$ each decision schedule is a feasible schedule. $(C2)$ each link ℓ must be scheduled by at least one decision schedule in $\mathcal{S}$. Note that these two conditions are the same as those in the standard CSMA algorithm [6]. $(C3)$ each link scheduled by a decision schedule can broadcast C bits to its neighbors in a time slot. We will discuss in Section IV how to implement the decision schedules at each time slot. Now, we describe our algorithm.

Virtual-Multi-Channel CSMA Algorithm: At each time t,

- *Decision Phase:* Choose a decision schedule $\vec{S}^D(t) = [S^D_\ell(t)]$ randomly in $\mathcal{S}$.
- *Update Phase:* For each link ℓ, if $S^D_\ell(t) = 0$, let $\vec{V}_\ell(t) = \vec{V}_\ell(t-1)$. Otherwise, perform Algorithm 1.

Algorithm 1 Update phase for link ℓ if $S^D_\ell(t) = 1$:

Choose a permutation $(n^\ell_1, n^\ell_2, \cdots, n^\ell_C)$ of the set $\{1, 2, \cdots, C\}$ uniformly at random. Let $f_\ell(x) = \exp(\alpha U_\ell(\frac{x}{C}))$ and $x^1_\ell = \sum_{k=1}^C V_{\ell k}(t-1)$.
for $i = 1$ to C **do**
  if $V_{\ell', n^\ell_i}(t-1) = 1$ for any link $\ell' \in \mathcal{E}_\ell$ **then**
    $V_{\ell, n^\ell_i}(t) = V_{\ell, n^\ell_i}(t-1)$.
  else
    $V_{\ell, n^\ell_i}(t)$ is determined with the random distribution:

$$\begin{aligned} &P(\{V_{\ell,n^\ell_i}(t) = y\}) \\ = \ &\frac{f_\ell(x^i_\ell + y - V_{\ell,n^\ell_i}(t-1))}{f_\ell(x^i_\ell - V_{\ell,n^\ell_i}(t-1)) + f_\ell(x^i_\ell + 1 - V_{\ell,n^\ell_i}(t-1))}, \end{aligned} \tag{3}$$

    where $y = 0$ or 1.
  end if
  $x^{i+1}_\ell = x^i_\ell + V_{\ell,n^\ell_i}(t) - V_{\ell,n^\ell_i}(t-1)$.
end for
Link ℓ broadcasts $\vec{V}_\ell(t)$ to all of its neighbors.

- *Scheduling Phase:* A common virtual-channel $k(t)$ is chosen by all links in the network uniformly at random, and each link ℓ transmits a packet if $V_{\ell,k(t)}(t) = 1$.
- *Congestion Control:* All links ℓ use the window-based flow-control algorithm with window-size 1, i.e., a new packet is injected to link ℓ only if a packet is served.

We note a key difference between the VMC-CSMA and the standard CSMA. Under the VMC-CSMA (correspondingly standard CSMA) algorithm, the random distribution for updating the schedule of link ℓ is a function of its own utility $U_\ell(x_\ell/C)$ (correspondingly its own queue length $Q_\ell(t)$). The significance of this key difference is as follows. Consider (3) and $V_{\ell,n^\ell_i}(t-1) = 0$ for an example, we have

$$P(\{V_{\ell,n^\ell_i}(t) = 1\}) = \frac{f_\ell(x^i_\ell+1)/f_\ell(x^i_\ell)}{1+f_\ell(x^i_\ell+1)/f_\ell(x^i_\ell)} \approx \frac{\exp(\frac{\alpha}{C}U'_\ell(\frac{x^i_\ell}{C}))}{1+\exp(\frac{\alpha}{C}U'_\ell(\frac{x^i_\ell}{C}))} \tag{4}$$

Recall that $U_\ell(\cdot)$ is a strictly concave function, which implies that $U'_\ell(\cdot)$ is a strictly decreasing function. Hence, if a link has larger x^i_ℓ value, it is less likely to activate itself in a new virtual channel, and vice versa. Thus, the schedules across virtual channels are in fact coordinated to reach a state $\vec{V}$ with total system utility close to the optimal value. Hence, VMC-CSMA does not constantly change the schedule and avoid the starvation problem. An additional benefit is that we can use the window-based flow-control as the congestion control component, which further controls the packets in the system and reduces the delay.

C. Utility Optimality

In this subsection, we study the capacity/utility performance when our VMC-CSMA algorithm reaches steady state. Note that under the VMC-CSMA algorithm, the global schedule $\vec{V}(t)$ behaves as a Markov chain. Hence, we will also refer to $\vec{V}(t)$ as the state. Let $\mathcal{V}$ be the state space of the Markov chain. We first introduce a proposition that describes the stationary distribution of the Markov chain.

Proposition 1: The stationary distribution of the Markov chain $\vec{V}(t)$ is given by $P(\vec{V}) = \frac{1}{Z}\exp(\alpha\sum_{\ell=1}^L U_\ell(r_\ell(\vec{V})))$, where Z is a normalization constant for all $\vec{V} \in \mathcal{V}$.

Proof: The proof is given in Appendix A ∎

Proposition 1 implies that the state $\vec{V}$ with larger value of total system utility has a higher chance to be visited. Further, since α appears in the exponent in the stationary distribution, as α increases, the probability of reaching the state with the largest utility, which is the solution to problem (2), will approach 1. However, due to the quantization effect with a finite C, the optimal utility in (2) may be smaller than the optimal utility of the original problem (1). Hence, the key question is how many virtual-channels C we need such that the two optimal-utility values are close. We note that this question is important because the complexity of our algorithm also increases with C. In practice, we would prefer a smaller value of C. A first thought for solving this question is to use the Caratheodory's theorem [23]. Specifically, since the capacity region of the optimization problem in (1) is a convex hull of all the feasible schedules, the Caratheodory's theorem tells us that the optimal solution to (1) can be written as the convex combination of $L+1$ feasible schedules. As a result, we may guess that we need C to be at least $\Theta(L)$ so that the optimal solution to (1) can be approximated by a valid global schedule $\vec{V}$. Somewhat surprisingly, by allowing a small margin ϵ for error and using a novel probabilistic argument, we can reduce the order of C to $\mathcal{O}(\log L)$. This nice result is presented in the following proposition. Recall that R^*_ℓ is the optimal solution of link ℓ in problem (1).

Proposition 2: If $\epsilon \le 0.1$ and $C \ge \frac{2\log L}{3\epsilon^2}$, then there exists a state $\vec{V}^s$ in the state space $\mathcal{V}$ of the Markov chain such that $r_\ell(\vec{V}^s) \ge R^*_\ell - \epsilon$, for all link ℓ.

Proof: The proof is given in Appendix B ∎

Note that since the rate is always larger than zero, it then follows from $r_\ell(\vec{V}^s) \geq R_\ell^* - \epsilon$ that $r_\ell(\vec{V}^s) \geq \max(R_\ell^* - \epsilon, 0) \triangleq [R_\ell^* - \epsilon]^+$. With the help of Proposition 2, we can prove the first main theorem in this paper. Recall that $U(\vec{r}) = \sum_{\ell=1}^L U_\ell(r_\ell)$.

Theorem 3: Under our VMC-CSMA algorithm, for any $\epsilon \leq 0.1$, we can choose $C > \frac{2\log L}{3\epsilon^2}$ and α large enough such that $P\{U(\vec{r}(\vec{V}(t))) \geq \sum_{\ell=1}^L U_\ell([R_\ell^* - \epsilon]^+)\} \geq 1 - \epsilon$, where $\vec{R}^*$ is the optimal solution of problem (1).

Proof: Consider a fixed C. Let $\vec{V}^{\max}$ be the solution to (2). Then $U(\vec{r}(\vec{V}^{\max})) \geq U(\vec{r}(\vec{V}))$, for all $\vec{V} \in \mathcal{V}$. Define the set $A = \{\vec{V} \in \mathcal{V} | U(\vec{r}(\vec{V})) = U(\vec{r}(\vec{V}^{\max}))\}$. Denote $f(\vec{V}) = \exp(\alpha U(\vec{r}(\vec{V})))$. Let $f^{\max} = f(\vec{V}^{\max})$. Note that $f(\vec{V}) = f^{\max}$ for all $\vec{V} \in A$. Since $\mathcal{V}$ is finite for a fixed C, there must exist $\epsilon' > 0$ such that $f(\vec{V}) \leq f^{\max} e^{-\alpha\epsilon'}$ for all $\vec{V} \notin A$. Using Proposition 1 and this inequality, we can then show that

$$\begin{aligned} P(\{\vec{V} \in A\}) &= \frac{\sum_{\{\vec{V}\in A\}} f(\vec{V})}{\sum_{\{\vec{V}\in A\}} f(\vec{V}) + \sum_{\{\vec{V}\notin A\}} f(\vec{V})} \\ &\geq \frac{f^{\max}|A|}{f^{\max}|A| + f^{\max} e^{-\alpha\epsilon'}(|\mathcal{V}| - |A|)}. \end{aligned} \quad (5)$$

This implies that for any ϵ, there exists α such that $P(\{\vec{V} \in A\}) > 1 - \epsilon$. Further, from Proposition 2, for any $\epsilon \leq 0.1$, we can choose a fixed $C \geq \frac{2\log L}{3\epsilon^2}$ such that for $\vec{V} \in A$, $U(\vec{r}(\vec{V})) \geq U(\vec{r}(\vec{V}^s)) \geq \sum_\ell U_\ell([R_\ell^* - \epsilon]^+)$. The results then follows. ∎

Theorem 3 states that as long as α is sufficiently large, in the steady-state the expected service rate achieved by VMC-CSMA and measured at every time-instant will achieve a close-to-optimal utility with high probability. Further, for a given value of ϵ (which characterizes the optimality loss), the number of channels C only needs to grow very slowly, i.e., logarithmically, with the network size. Define $r_\ell^{\min}$ as the worst rate for link ℓ among all global schedules with the maximum utility, i.e.,

$$r_\ell^{\min} = \min_{\{\vec{V}\in\mathcal{V} | U(\vec{r}(\vec{V})) = U(\vec{r}(\vec{V}^{\max}))\}} r_\ell(\vec{V}), \quad (6)$$

where $\vec{V}^{\max}$ is the solution of problem (2). Theorem 3 also leads immediately to the following corollary on the average throughput for each link ℓ and the total system utility.

Corollary 4: Under our VMC-CSMA algorithm, for any $\epsilon \leq 0.1$, we can choose $C > \frac{2\log L}{3\epsilon^2}$ and α large enough such that the time-averaged throughput R_ℓ of link ℓ has a lower bound $R_\ell \geq (1-\epsilon) r_\ell^{\min}$. Further, $U(\vec{R}) \geq (1-\epsilon) U([\vec{R}^* - \epsilon]^+)$.

Proof: To calculate the throughput of link ℓ, we have to consider the real schedule, which is determined both by $\vec{V}(t)$ and the common virtual channel $k(t)$ that we choose at each time slot. Note that $P\{k(t) = k\} = 1/C, k = 1, \cdots, C$. Further, $k(t)$ is i.i.d. across time-slots and is independent of $\vec{V}(t)$. Define the set $A = \{\vec{V} \in \mathcal{V} | U(\vec{r}(\vec{V})) = U(\vec{r}(\vec{V}^{\max}))\}$. Let $P(\vec{V})$ be steady state probability when the state of the Markov chain $\vec{V}(t)$ is $\vec{V}$. From the proof of Theorem 3, we can find α such that $P\{\vec{V}(t) \notin A\} \leq \epsilon$. The average throughput of link ℓ can then be derived as follows.

$$\begin{aligned} R_\ell &= \sum_{\vec{V}\in\mathcal{V}} \sum_{k=1}^C P\{k(t) = k, \vec{V}(t) = \vec{V}\} V_{\ell k} \\ &= \sum_{\vec{V}\in\mathcal{V}} \sum_{k=1}^C \tfrac{1}{C} P\{\vec{V}(t) = \vec{V}\} V_{\ell k} \\ &= \sum_{\vec{V}\in\mathcal{V}} P\{\vec{V}(t) = \vec{V}\} r_\ell(\vec{V}) \\ &\geq \sum_{\vec{V}\in A} P\{\vec{V}(t) = \vec{V}\} r_\ell(\vec{V}) \\ &\geq \sum_{\vec{V}\in A} P\{\vec{V}(t) = \vec{V}\} r_\ell^{\min} \geq (1-\epsilon) r_\ell^{\min} \end{aligned} \quad (7)$$

Now, combining Theorem 3 and the fact that $U(\cdot)$ is a concave function, we can have that

$$U(\vec{R}) \geq \sum_{\vec{V}} P(\vec{V}) U(\vec{r}(\vec{V})) \geq (1-\epsilon) U([\vec{R}^* - \epsilon]^+).$$

This concludes the proof. ∎

Remark: Under suitable time-scale separation assumptions, the long-term time-averaged service rate under the standard CSMA algorithm can also be shown to achieve the optimal utility, which is similar to Corollary 4. However, we emphasize that Theorem 3 is in fact much stronger. Since VMC-CSMA chooses a virtual channel *i.i.d.* across time, even over a shorter time-interval around time t, the average service rate of each link will be close to $r_\ell(\vec{V}(t))$. Thus, Theorem 3 states that the service rate averaged even over shorter intervals will attain close-to-optimal system utility with high probability. In contrast, due to the starvation problem, the short-term service rate of the standard CSMA algorithm could be much more unfair and far away from the optimal utility.

D. Delay Performance

In this subsection, we study the delay performance of VMC-CSMA. A natural definition for packet delay is the number of time-slots from the time when the packet is injected to the buffer of a link by the congestion control component at the corresponding source, to the time when the packet is served by the link. It turns out that this packet delay has a simple characterization. Note that since we use window-based control with window size 1, the number of packets in the buffer at each link ℓ is always 1. Further, the rate that packets arrive to and depart from link ℓ is given by R_ℓ. Hence, by Little's law [24], the expected packet delay (average across packets) at link ℓ must be equal to $1/R_\ell$, which is upper bounded by $\frac{1}{(1-\epsilon) r_\ell^{\min}}$. Thus, a high throughput at link ℓ immediately translates to a low expected packet delay.

However, as we have mentioned in Section II, the above definition of packet delay does not fully capture the effect of starvation problem. A better metric is HOL (head-of-line) waiting time, which is the time that the HOL packet at link ℓ has waited in the system. For example, an extended period of starvation will likely cause large expected HOL waiting time average across time-slots (please refer to the example in Section II). The following lemma shows that under VMC-CSMA algorithm, the tail distribution of the HOL waiting time will decay quickly.

Lemma 5: Under our VMC-CSMA algorithm, for a fixed integer $d > 0$ and any $\epsilon > 0$, there exists sufficiently large α so that for each link ℓ, $P\{\text{HOL waiting time} \geq d\} \leq (1 - r_\ell^{\min})^d + \epsilon$.

Proof: Define the set $A = \{\vec{V} \in \mathcal{V} | U(\vec{r}(\vec{V})) = U(\vec{r}(\vec{V}^{\max}))\}$. From the proof of Theorem 3, we can find α such that $P\{\vec{V}(t) \notin A\} \leq \epsilon/d$. Let E be the event that $\vec{V}(t_1) \in A, t_1 = t-d+1, \cdots, t$. Hence, by the union bound, $P(E^c) \leq d\frac{\epsilon}{d} = \epsilon$. Let $d_\ell(t)$ be the HOL waiting time for link ℓ at time t. We have that $P\{d_\ell(t) \geq d|E\} = P\{\text{link } \ell \text{ is not served for } t-d+1 \text{ to } t|E\} \leq (1-r_\ell^{\min})^d$. Use this inequality, we can then show that $P\{d_\ell(t) \geq d\} = P\{d_\ell(t) \geq d|E\}P\{E\} + P\{d_\ell(t) \geq d|E^c\}P\{E^c\} \leq (1-r_\ell^{\min})^d + \epsilon$. This concludes the proof. ∎

Until now, we have developed bounds on the expected packet delay and the tail distribution of the HOL waiting time. These bounds only depend on $r_\ell^{\min}$ and are otherwise independent of the network size. Given that the delay of CSMA algorithms have been observed to grow exponentially with the network size [7,8]. Our ultimate goal in this section is to develop conditions such that the delay of VMC-CSMA does not grow with the network size. For this statement to hold, we need to bound $r_\ell^{\min}$ away from zero. Unfortunately, without additional restrictions on the network setting, it is still possible to construct an example such that $r_\ell^{\min}$ is close to 0. Due to the space constraints, we refer the readers to [25] for the example. This suggests that without additional restrictions, the value of $r_\ell^{\min}$ may be arbitrarily close to 0. However, this effect has less to do with our VMC-CSMA algorithm. Rather, it is mainly determined by the system setting and the rate-allocation corresponding to the maximum utility. Basically, when the number of links in $\mathcal{E}_\ell$ is large or the utility function of link ℓ has a small derivative, the optimal solution for link ℓ may already be very small. Motivated by the example, we introduce the following conditions. Let $\Delta_\ell = |\mathcal{E}_\ell|$ be the number of links in the neighborhood of link ℓ. Let $\mathcal{K}_\ell$ be the maximum number of links that can be scheduled simultaneously in the neighborhood of link ℓ. In the new condition, we assume that $\max_\ell \Delta_\ell \leq \Delta$ and $\max_\ell \mathcal{K}_\ell \leq \mathcal{K}$ for some constants Δ and $\mathcal{K}$. Note that this condition is very general and will likely hold for a large class of network topologies. Further, we assume that the utility function of all links is of the form $\log(\cdot + h) - \log(h)$. Under these conditions, the next proposition shows that we can have a lower bound for $r_\ell^{\min}$.

Proposition 6: Suppose that the utility functions of all links are of the form $\log(\cdot + h) - \log(h)$. Further, assume that $0 < h < \frac{1}{(\Delta+2)(\mathcal{K}-1)}$ and $\frac{\Delta_\ell+1}{C} < \frac{1}{\mathcal{K}_\ell(\Delta_\ell+2)} - \frac{h(\mathcal{K}_\ell-1)}{\mathcal{K}_\ell}, \forall \ell$, then under the VMC-CSMA algorithm, we have $r_\ell^{\min} \geq \frac{1}{\mathcal{K}_\ell(\Delta_\ell+2)} - \frac{h(\mathcal{K}_\ell-1)}{\mathcal{K}_\ell} - \frac{\Delta_\ell+1}{C}$.

Proof: The proof is given in Appendix C ∎

With Proposition 6, it is then easy to show the following lower bound for $r_\ell^{\min}$ that is independent of the network size.

Corollary 7: Suppose that the utility functions of all links are of the form $\log(\cdot + h) - \log(h)$. If $0 < h \leq \frac{1}{2(\Delta+2)(\mathcal{K}-1)}$ and $C \geq 4\mathcal{K}(\Delta+1)(\Delta+2)$, then under the VMC-CSMA algorithm, we have that $r_\ell^{\min} \geq \frac{1}{4\mathcal{K}(\Delta+2)}$.

Proof: Since $h \leq \frac{1}{2(\Delta+2)(\mathcal{K}-1)}$, we know that $h < \frac{1}{2(\Delta_\ell+2)(\mathcal{K}_\ell-1)}$ for all ℓ. As a result, we have that $\frac{1}{\mathcal{K}_\ell(\Delta_\ell+2)} - \frac{h(\mathcal{K}_\ell-1)}{\mathcal{K}_\ell} \geq \frac{1}{2\mathcal{K}(\Delta+2)}$. Further, from $C \geq 4\mathcal{K}(\Delta+1)(\Delta+2)$, we know that for all ℓ, $C \geq 4\mathcal{K}(\Delta_\ell+1)(\Delta+2)$ and $\frac{\Delta_\ell+1}{C} \leq \frac{1}{4\mathcal{K}(\Delta+2)}$. Hence, $\frac{1}{\mathcal{K}_\ell(\Delta_\ell+2)} - \frac{h(\mathcal{K}_\ell-1)}{\mathcal{K}_\ell} - \frac{\Delta_\ell+1}{C} \geq \frac{1}{4\mathcal{K}(\Delta+2)}$. By Proposition 6, we then have $r_\ell^{\min} \geq \frac{1}{4\mathcal{K}(\Delta+2)}$. ∎

Proposition 6 and Corollary 7 show that as long as the number of interfering neighbors of each link is bounded, by making the utility functions sufficiently steep (i.e., a small h) and by using a sufficiently large (but constant) number of channels C, $r_\ell^{\min}$ can be bounded from below by a constant independent of the network size. Hence, the average delay of each link ℓ does not grow with the network size, and the tail distribution of HOL waiting time is guaranteed to decay quickly.

We note that in [9], the authors show that for open-loop systems, even at an offered load that is a small fraction of the optimal capacity, there exists network topologies such that either the complexity or the delay must grow exponentially with the network size. We note that our result differs from [9] in two important aspects. First, the delay definition is different. We are interested in the steady state delay, i.e., after the Markov chain converges to the stationary distribution. The convergence time (which we did not capture in this paper) may still be exponential in the network size. In contrast, the delay in [9] is the worst-case delay across time and hence also captures the transient phase. Second, we use a closed-loop system (with congestion control) in contrast to an open-loop system in [9]. Hence, our results do not contradict with those of [9].

E. Computational Complexity and Communication Overhead

In this subsection, we discuss the computational complexity and communication overhead of the VMC-CSMA algorithm. For each link, the computational complexity of the VMC-CSMA algorithm is $\mathcal{O}(C)$. This is because all of the computations can be carried out in parallel at all links, and every link only needs to go through the C virtual channels. Further, finding the random permutation in the "update phase" is of complexity $\mathcal{O}(C)$ through the use of the Fisher-Yates shuffle [26]. Regarding the communication overhead, note that each link ℓ can use a C-bit vector to denote its schedule $\vec{V}_\ell$ across all virtual channels. At each time t, a link scheduled by $\vec{S}^D(t)$ may need to broadcast this C-bit vector to all of its neighbors. Hence, the communication overhead is also $\mathcal{O}(C)$. From Theorem 3 and Corollary 4, we can see that $C = \mathcal{O}(\log L)$, which grows very slowly with the network size. Hence, our algorithm is scalable to large networks.[3] In practice, all C bits may be put into a single control packet in one time-slot. For example, if the size of a control packet is 250 bytes, it can accommodate $C = 2000$ virtual channels. Even for a large network with $L = 1000$ links, $C = 2000$ virtual channels are sufficient to reduce ϵ to be 0.05 (see Theorem 3). Hence, the corresponding capacity/utility reduction will be very small. If a data packet of length 2000 bytes (like in 802.11) is transmitted in each time-slot, such a control packet corresponds to a low overhead of 12.5%. In practice, this overhead can be further reduced by reducing the frequency of updates. For example, if the operation in the update phase is performed every 5 time-slots, the communication overhead is further reduced to 2.5%.

[3]In comparison, the complexity of the standard CSMA algorithms is $\mathcal{O}(1)$ and is independent of the network size [5,6]. However, they suffer the large delay problem as mentioned in Section I.

Note that at each time-slot, we can still randomly choose a virtual-channel schedule for transmission. Thus, reducing the frequency of updates will only affect the convergence of the algorithm, but it will not affect the capacity and delay once the Markov chain converges to its stationary distribution.

F. VMC-CSMA under Exogenous Packet Arrivals

Throughout this paper, we have focused on an infinite-backlog setting where the packet injection to the system can be controlled. In this subsection, we briefly discuss how the key idea of VMC-CSMA may potentially be generalized to the setting where packets arrive according to an exogenous stochastic process. The basic idea is as follows. Assume that we know the average packet-arrival rate λ_ℓ of each link, and we know that the vector $\vec{\lambda} = [\lambda_\ell]$ is strictly inside the capacity region Ω, i.e., there exists ϵ_1 such that $\vec{\lambda} + \epsilon_1 \vec{1} \in \Omega$. If we can let the VMC-CSMA algorithm compute a rate-allocation vector $R = [\vec{R}]$ such that $r_\ell^{\min} > \lambda_\ell + \gamma$, where $\gamma < \epsilon_1$, then the end-to-end packet delay may also be small. To see this, assume without loss of generality that the packet arrivals at each link are *i.i.d.* across time. As we can see from the proof of Lemma 5, whenever the event A occurs (whose probability is close to 1), the service at each link stochastically dominates an *i.i.d.* service with probability $r_\ell^{\min}$. Hence, as long as $r_\ell^{\min} > \lambda_\ell + \gamma$, we expect that the end-to-end packet delay at link ℓ will be $\mathcal{O}(1/\gamma)$. The challenges, however, are to design a utility function to compute such vector $\vec{R}$ and to show that even though there is a small probability that the event A does not occur, the expected delay will still be $\mathcal{O}(1/\gamma)$. We refer the readers to [25] for an example that resolves the first challenge, and we leave the second challenge for future work.

IV. Implementation

In this section, we discuss two implementation issues. First, it is well known that for CSMA algorithms, there is a trade-off between optimality and convergence speed depending on the value of α. In practice, we want to choose a suitable α that is not too large to shorten the convergence time. However, when α is not very large, we also observe a common source of performance degradation, which can be explained as follows. Suppose that the Markov chain has found the optimal state $\vec{V}^{\max}$, and a link ℓ is active in virtual channel k. When α is not very large, there is a substantial probability that link ℓ will turn itself off in virtual channel k. If all other links in its neighborhood, i.e., $\mathcal{E}_\ell$, have interfering links that are active in virtual channel k, no links in $\mathcal{E}_\ell$ can be turned on in virtual channel k. Hence, link ℓ may turn itself on again in virtual channel k later, and for link ℓ, the transmission opportunities during this period are lost unnecessarily. Such performance degradation can be easily avoided with the following improved algorithm. We call the schedule computed by VMC-CSMA as the soft schedule. In addition, we now introduce hard schedule as follows. Whenever a link ℓ is turned on in virtual channel k by the soft schedule (i.e. $V_{\ell k} = 1$), we also turn on link ℓ in virtual channel k in the hard schedule. However, even if $V_{\ell k} = 0$, we will only turn off link ℓ in the virtual channel k in the hard schedule when a neighboring link of link ℓ decides to use virtual channel k (i.e., it turns itself on in virtual channel k by the soft schedule). As a result, the hard schedule will give up transmission opportunities until the last minute, and we can avoid the throughput loss due to the reason described above. Note that our earlier throughput/delay results still hold. Now, we formally describe this simple fix. We define an additional hard virtual schedule $H_{\ell k}$ for all virtual-channel schedule $V_{\ell k}$, and we will call $V_{\ell k}$ the soft virtual-channel schedule or simply the virtual schedule. The update of soft virtual schedule $V_{\ell k}$ will remain the same, and the update of hard virtual schedule $H_{\ell k}$ is as follows.

Update Hard Virtual Schedule: At each time t,

- If $V_{\ell k}(t) = 1$, we let $H_{\ell k}(t) = 1$.
- If $V_{\ell k}(t) = 0$, we let $H_{\ell k}(t) = 0$ when link ℓ receives a broadcast from any link ℓ' with $V_{\ell' k}(t) = 1$, and $\ell' \in \mathcal{E}_\ell$. Otherwise, we let $H_{\ell k}(t) = H_{\ell k}(t-1)$.

Now, we modify the scheduling phase as follows.

Scheduling: At time t, a common virtual channel k is chosen by all links in the network uniformly at random, and each link ℓ uses $H_{\ell k}(t)$ as the real schedule at time t.

Note that it is easy to see that each "hard" virtual-channel schedule will eventually become a maximal schedule. (Recall that a maximal schedule must be feasible, and no more links can be added to the schedule without interfering with the links that has been scheduled.) The reason is that for the k^{th} hard virtual schedule of all the links, i.e., $(H_{1k}, H_{2k}, \cdots, H_{Lk})$, once it becomes a maximal schedule, it will remain as a maximal schedule in the rest of the time because each link only relinquishes the transmission opportunity when one of its neighbors is activated by the algorithm. Further, the k^{th} hard virtual schedule of all the links will have a trend to become a maximal schedule starting from time 0. This confirms our claim that we can avoid wasting transmission opportunity when the soft virtual schedule $\vec{V}(t)$ jumps in the Markov chain. It is easy to see that the throughput performance of this improved scheduling algorithm using hard schedule is always better than the algorithm described in Section III-B. Hence, the analytical results in Section III-C and III-D still hold.

Second, we briefly discuss how to choose the decision schedule at each time slot. Recall in Section III-B that at each time slot, we choose a decision schedule $\vec{S}^D$ with a fixed probability distribution from a set $\mathcal{S}$ of decision schedules. Further, the set $\mathcal{S}$ of decision schedules should meet conditions $(C1)$, $(C2)$, and $(C3)$. Note that for the standard CSMA algorithms [6], the authors provide a random backoff algorithm for computing decision schedules that satisfy the first two conditions. There are a number of ways to satisfy the remaining third condition. One possibility is to assume that there is a separate control channel (e.g. using CDMA). On the other hand, if such a separate control channel is not available, the other possibility is to make the decision schedule more sparse. Specifically, two active links in the decision schedule are not neighbors, and the two active links do not share any common neighbors. As a result, if a link is not scheduled by the decision schedule, it will not receive more than one broadcast transmission.

Such a decision schedule can be computed in each time-slot via a random-backoff-based algorithm similar to that of

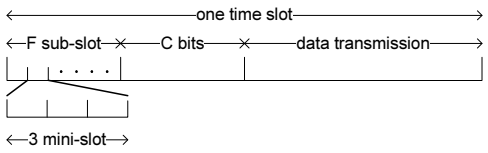


Fig. 2. The composition of a time slot.

[6] with additional signaling messages to resolve conflicts at common neighbors. The details are as follows. We first describe the composition of a time slot as shown in Figure 2. The time slot is composed of three phases. The first phase includes F sub-slot, and each sub-slot has three mini-slots. A decision schedule will be selected distributively in the first phase. The second phase is the time required for each link scheduled by the decision schedule to broadcast the C bits of information for the updated virtual-channel schedules. The third phase is the actual data transmission. The decision schedule can be computed in a distributed fashion as follows. At the beginning of each time slot, we assign each link a mark bit equal to 0. Each link will choose a value from 1 to $F+1$ to attempt to be included in the decision schedule. If a link chooses $F+1$, it will not attempt at all. Suppose that a link ℓ chooses i, $i \leq F$. If link ℓ hears any successful transmission attempts before sub-slot i, it will set its mark bit to 1, and it will not attempt to be included in the decision schedule in this time-slot. On the other hand, if link ℓ does not hear any successful transmission attempts before sub-slot i, the transmitter of link ℓ transmits a RTS in the first mini-slot. This RTS signal indicates to its neighbors that the link ℓ attempts to be included in the decision schedule. The receiver of link ℓ will transmit a CTS in the second mini-slot to confirm the reception of RTS signal. For any other link k that does not choose the i^{th} sub-slot, it sends a signal in the third mini-slot under one of the two conditions: (1) if there is a conflict because its mark bit is 1, and it senses any message transmissions in the first mini-slot, or (2) it senses a conflict due to more than one RTS transmissions in the first mini-slot.

At the end of the i^{th} sub-slot, if the transmitter of link ℓ does not receive a CTS from the receiver for the RTS transmitting in the first mini-slot, which implies that link ℓ collides with other transmissions in the first mini-slot, the attempt fails. Further, if link ℓ hears any signals in the third mini-slot, the attempt fails. Otherwise, link ℓ is included in the decision schedule. This algorithm ensures that the links selected in the decision schedule do not have common neighbors, and hence their neighboring links can receive the C-bit broadcast without interference. Further, each link has a positive probability to be selected in the decision schedule because there is a positive probability that all the neighboring links will not attempt to be included in the decision schedule.

V. Simulation

We evaluate the VMC-CSMA algorithm in two topologies with our C++ simulator. Specifically, we simulate the improved scheduling algorithm and a random backoff-based decision-schedule algorithm discussed in Section IV. For both topologies, there is a one-hop flow associated with each link, and its utility function is given by $\log(10^{-5}+\cdot)-\log(10^{-5})$. Further, the simulation time is 15,000 slots.

We first study a 8-by-8 torus interference graph as described in Section I and compare our algorithm with the standard CSMA algorithm.[4] Note that because of symmetry, the optimal rate R_ℓ^* for each link ℓ under this setting will be 0.5. For the standard CSMA algorithm, the weight is chosen as $w_\ell(t) = \log(0.5Q_\ell(t))$, and the parameter β for congestion control is 0.1. For our VMC-CSMA algorithm, we let $C = 30$ and $\alpha = 29$. The corresponding $\epsilon = 0.3$.[5] We denote the node in the top left corner of the torus as node 0 and the node right next to it as node 1. Note that node 0 is active in the odd schedule, and node 1 is active in the even schedule (See Fig. 1). The average throughput, average delay across packets, and average HOL waiting time across the time-slots for node 0 are presented in the following table.

	throughput	delay	HOL waiting time
CSMA	0.427	159	372.8
VMC-CSMA	0.479	2.09	2.10

The result shows that our algorithm can indeed achieve throughput close to the optimal rate. Further, the delay performance is exactly equal to the inverse of the average throughput. In contrast, both the packet delay and the HOL waiting time of the standard CSMA algorithm are 80 and 170 times larger, respectively.[6] In Figure 3(a), we plot the tail distribution of the HOL waiting time under both algorithms. The HOL waiting time of our algorithm decays quickly, which confirms Lemma 5 and explains why the average HOL waiting time is small. In contract, the HOL waiting time of CSMA decays slowly (see Fig. 3(b)) due to the starvation problem.

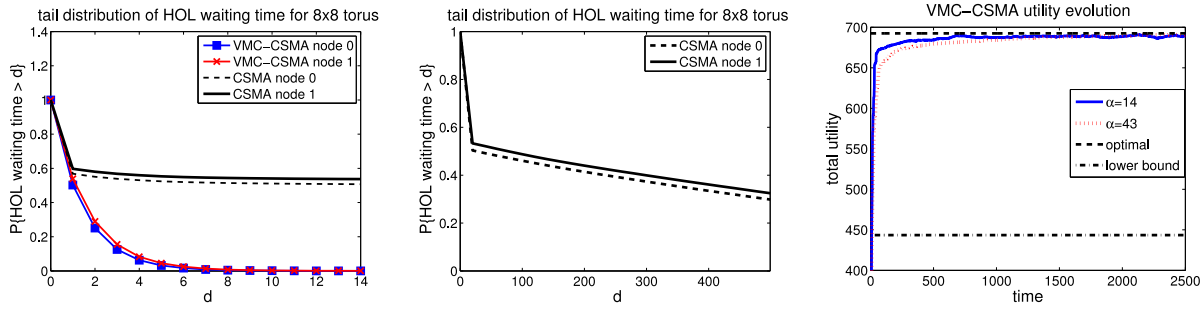
To give a sense of the convergence time of our algorithm, in Fig. 3(c), we plot the time evolution of the instantaneous system utility $\sum_{\ell=1}^{L} U_\ell(r_\ell(\vec{V}(t)))$ under different α. The simulation results show that the utility approaches very close to the optimal utility after 100 time slot. Further, the utility is larger than the lower bound given by $(1-\epsilon)U([\vec{R}^*-\epsilon]^+)$ (Corollary 4) even for a small value of C. Note that when α is larger, the utility value after convergence is also closer to the optimal value of (1). However, larger α also incurs longer convergence time.

Before we proceed to a larger topology, we comment on the choice of α for different values of C. Through our simulation, we observe that a rule of thumb is to choose α to be proportional to C. The reason can be explained by equation (4). For a fixed value of $r_\ell = \frac{x_\ell^i}{C}$, as C increases, in order to maintain the same probability of adding another virtual channel, we should increase α proportional to C. Our simulation studies indicate that as long as the ratio α/C

[4]Note that we use this simple and symmetric topology first because it allows us to easily compare with the optimal solution.

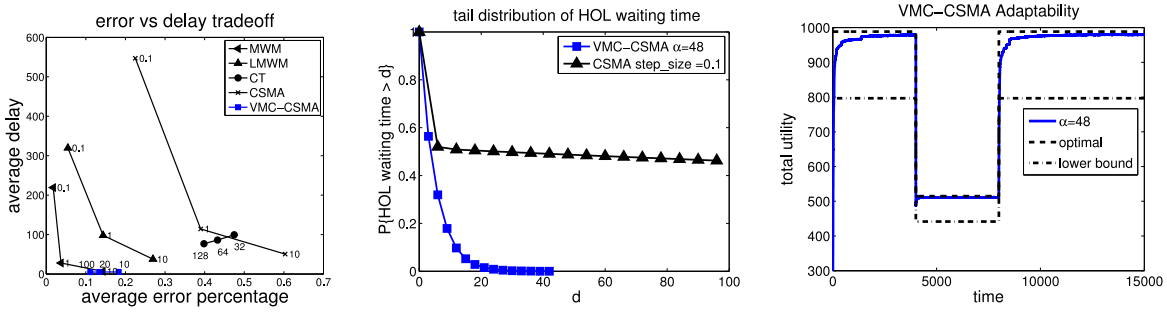
[5]Note that Theorem 3 requires $\epsilon = 0.1$ and a corresponding $C = 277$. In this simulation setting, we intentionally choose a small C to show that in reality we can also use small C to achieve good performance.

[6]Note that the expected packet delay is an average over packets, and the expected HOL waiting time is an average across time. As in the classical Inspector's Paradox [27], these two ways of taking expectation will lead to different values when the inter-service time is not memoryless, which is the case for the standard CSMA algorithm.



(a) (b) (c)

Fig. 3. The simulation results for the 8-by-8 torus graph.



(a) (b) (c)

Fig. 4. The simulation results for a random network with 100 nodes and 100 links.

is fixed, the tradeoff between convergence and optimality is roughly the same for different values of C. As this ratio increases, the optimality is improved at the cost of a longer convergence time. Thus, we will use this rule of thumb in the following simulation results.

Next, we simulate our algorithm under a larger random topology with 100 nodes and 100 links. We set the maximum node degree to be 4, and under this constraint, each link is generated by randomly choosing two nodes.[7] We assume that each link will interfere with the links that are two-hop away (i.e., the two-hop interference model). In addition to the standard CSMA algorithm, we will also compare our algorithm with two other algorithms: the constant-time (CT) distributed algorithm [28] and the well-known maximum weighted matching algorithm (MWM) [3]. However, We caution that since these algorithms incur very different computational complexity and communication overhead, it is difficult to conduct a completely fair comparison. Because our proposed VMC-CSMA algorithm, the standard CSMA algorithm, and the CT algorithm require only one round of local control-message exchange (including channel sensing) in each iteration, we will compare their performance directly. On the other hand, the MWM algorithm is a centralized algorithm that requires the queue length information of all links. Further, its complexity may be exponential under the two-hop interference model. In order to make the comparison slightly more fair, we simulate a version of the MWM algorithm that exchanges queue length information and computes the schedule once every 100 time-slots. We call this algorithm LMWM (Low-frequency MWM). Note that even at the reduced frequency, the LMWM algorithm is still very costly to implement due to its high complexity and the requirement of collecting global queue length information.

In Fig. 4(a), we report the throughput and delay tradeoff of each algorithm. Specifically, the y-axis is the average packet delay, and the x-axis is the average error percentage of the rate vectors computed by these algorithms under various parameter setting. The error percentage for each link is defined by $\max[\text{optimal\_rate} - \text{allocated\_rate}, 0]/\text{optimal\_rate}$. Thus, the most desirable algorithm will correspond to a point close to the origin, where it attains both high capacity and low delay. As we can see in Fig. 4(a), for the VMC-CSMA algorithm, when we vary C (which is labeled next to each point) and set $\alpha = C * 0.48$, the VMC-CSMA consistently achieves low error percentage and low delay. On the other hand, the performance curves for CSMA and LMWM (each point is labeled with the value of β, the step size used in the congestion control) are significantly worse. For CT algorithm, the curve (each point is labeled with the value of the backoff window size) exhibits large error percentage because CT can only achieve a fraction of the capacity region. The performance of MWM algorithm is also included in Fig. 4(a) for a reference. However, MWM incurs much higher complexity. Hence, the VMC-CSMA algorithm is the only one that can attain high throughput utility and low packet delay with low complexity.

Similar to the torus interference graph, we also report the tail distribution of the HOL waiting time. Specifically, we let

[7]We set a maximum on the node degree simply to guarantee a lower bound of $r_\ell^{\min}$ as suggested in Proposition 6 and Corollary 7.

$C = 100$ and choose a node where the optimal allocated rate is 0.2177. As shown in Fig. 4(b), the tail distribution under our algorithm again decays quickly. In contrast, the tail distribution of the HOL waiting time under CSMA algorithm does not decay even after 100. Finally, we simulate our algorithm under the possibility of traffic changes and observe how our algorithm adapts with the change. Specifically, we let $C = 100$ and $\alpha = 48$. Further, we turn off the traffic on half of the links after 4000 time-slots and turn on the traffic after 8000 time-slots. Then, we record the evolution of the total utility compared with the optimal utility computed offline. The result is shown in Fig. 4(c). We can see that our algorithm adapts well to the traffic changes, and the instantaneous utility quickly approaches to the optimal utility.

VI. Conclusion

In this paper, we propose a virtual-multi-channel (VMC-) CSMA algorithm that can provably achieve high throughput utility and low delay with low complexity for wireless networks under the protocol interference model and a single-hop utility-maximization setting. The key idea of VMC-CSMA is to resolve the starvation problem of standard CSMA algorithms by emulating a multi-channel system with C virtual channels and computing multiple feasible schedules simultaneously. VMC-CSMA inherits the distributed nature of CSMA, and the complexity of each link grows linearly with C. We use a novel probabilistic argument to show that VMC-CSMA can approach arbitrarily close to the optimal total system utility with C (and hence the complexity) increasing logarithmically with the network size. Further, we derive sufficient conditions for the network topology and utility functions under which the expected packet delay and the distribution of its head-of-line (HOL) waiting time can be bounded independently of the network size. Our simulation results show that VMC-CSMA algorithm indeed achieves both high throughput utility and low delay with low-complexity operations. In the future work, we will study how to extend this novel idea to the wireless networks with arrivals and multi-hop traffic.

Appendix A
Proof of Proposition 1

It is easy to verify that the Markov Chain is irreducible and aperiodic. The reason is that every feasible state $\vec{V}$ can communicate with state $\vec{0}$, and the period of state zero is 1. By [27, Theorem 4.3.3], if we can find a stationary distribution of the Markov chain, this stationary distribution will be the unique distribution. Now, we will show that $P(\vec{V}) = \frac{1}{Z}\exp(\sum_{\ell=1}^{L} U_\ell(\frac{x_\ell(\vec{V})}{C}))$ is the correct distribution. We will demonstrate the correctness by verifying the local balance equations. Consider two states $\vec{V}^1$ and $\vec{V}^2$. If $\vec{V}^1$ has a transition to $\vec{V}^2$, it implies that there exists a decision schedule $\vec{S}^D$ such that for every link ℓ that is scheduled by $\vec{S}^D$, it is feasible to change the state from $\vec{V}_\ell^1$ to $\vec{V}_\ell^2$. Further, if link ℓ is not scheduled by $\vec{S}^D$, then $\vec{V}_\ell^1 = \vec{V}_\ell^2$. Otherwise, there is no transition from $\vec{V}^1$ to $\vec{V}^2$. Now, based on these conditions, we also know that there is a transition from $\vec{V}^2$ to $\vec{V}^1$. The reason is that for every link ℓ that is scheduled by the same decision schedule $\vec{S}^D$, it is feasible to change the state from $\vec{V}_\ell^2$ to $\vec{V}_\ell^1$. Further, if link ℓ is not scheduled by $\vec{S}^D$, then $\vec{V}_\ell^2 = \vec{V}_\ell^1$. Note that there may be multiple decision schedules that can be used to change the state from $\vec{V}^1$ to $\vec{V}^2$. We will call these decision schedules "feasible decision schedules". Further, for any link ℓ that is scheduled by the decision schedule, any permutation can be used to change the state from $\vec{V}_\ell^1$ to $\vec{V}_\ell^2$. Now, let $P(\vec{S}^D)$ be the probability that decision schedule $\vec{S}^D$ is chosen. Also, let $\vec{n}^\ell = (n_1^\ell, \cdots, n_C^\ell)$ be a vector represent a permutation of the set $\{1, 2, \cdots, C\}$, and let $\vec{n}'^\ell = (n_C^\ell, \cdots, n_1^\ell)$. Note that $\vec{n}'^\ell$ is the permutation $\vec{n}^\ell$ in reverse order. To verify the local balance equations, we only need to show that

$$\begin{aligned} & P(\vec{V}^1) \sum_{\{\text{feasible } \vec{S}^D\}} P(\vec{S}^D) \prod_{\{\ell \in \vec{S}^D\}} \Big(\sum_{\vec{n}^\ell} \frac{P_{\ell,n_1^\ell} \cdots P_{\ell,n_C^\ell}}{C!}\Big) \\ = \; & P(\vec{V}^2) \sum_{\{\text{feasible } \vec{S}^D\}} P(\vec{S}^D) \prod_{\{\ell \in \vec{S}^D\}} \Big(\sum_{\vec{n}'^\ell} \frac{P'_{\ell,n_C^\ell} \cdots P'_{\ell,n_1^\ell}}{C!}\Big). \end{aligned} \tag{8}$$

Note that P_{ℓ,n_i^ℓ} is the probability of changing $V_{\ell,n_i^\ell}^1$ to $V_{\ell,n_i^\ell}^2$, which is the i^{th} iteration of updating the virtual-channel schedules from $\vec{V}_\ell^1$ to $\vec{V}_\ell^2$ with permutation $\vec{n}^\ell$. Similarly, P'_{ℓ,n_i^ℓ} is the probability of changing the state from $V_{\ell,n_i^\ell}^2$ to $V_{\ell,n_i^\ell}^1$, which is the $(C+1-i)^{th}$ iteration of updating the virtual-channel schedules from $\vec{V}_\ell^2$ to $\vec{V}_\ell^1$ with permutation $\vec{n}'^\ell$. Let $f_\ell(x) = \exp(\alpha U_\ell(\frac{x}{C}))$. Also, let $x_\ell^2 = x_\ell(\vec{V}^2)$, and $x_\ell^1 = x_\ell(\vec{V}^1)$. Recall that for any ℓ that is not scheduled by $\vec{S}^D$, we have that $\vec{V}_\ell^2 = \vec{V}_\ell^1$. It implies that $x_\ell^2 = x_\ell^1$ if ℓ is not scheduled by $\vec{S}^D$. We can plug in $P(\vec{V}^i) = \frac{1}{Z}\exp(\alpha\sum_{\ell=1}^{L} U_\ell(\frac{x_\ell^i}{C})), i = 1, 2$, and rewrite equation (8) to the following equation:

$$\begin{aligned} & \sum_{\{\text{feasible } \vec{S}^D\}} \frac{P(\vec{S}^D)}{Z} \prod_{\{\ell \in \vec{S}^D\}} \Big(\sum_{\vec{n}^\ell} \frac{P_{\ell,n_1^\ell} \cdots P_{\ell,n_C^\ell}}{C!}\Big) f_\ell(x_\ell^1) \\ = \; & \sum_{\{\text{feasible } \vec{S}^D\}} \frac{P(\vec{S}^D)}{Z} \prod_{\{\ell \in \vec{S}^D\}} \Big(\sum_{\vec{n}'^\ell} \frac{P'_{\ell,n_C^\ell} \cdots P'_{\ell,n_1^\ell}}{C!}\Big) f_\ell(x_\ell^2). \end{aligned} \tag{9}$$

If we can show that given any permutation $\vec{n}^\ell = (n_1^\ell, \cdots, n_C^\ell)$, we have that

$$f_\ell(x_\ell^1) P_{\ell,n_1^\ell} \cdots P_{\ell,n_C^\ell} = P'_{\ell,n_C^\ell} \cdots P'_{\ell,n_1^\ell} f_\ell(x_\ell^2), \tag{10}$$

then equation (9) will be true, and we could finish the proof. Now, we show that equation (10) is true.

By equation (3), P_{ℓ,n_i^ℓ} and P'_{ℓ,n_i^ℓ} can be described by fractions, and we will show that the denominator (resp. numerator) on the left hand side of equation (10) is equal to the denominator (resp. numerator) on the right hand side of equation (10). We first focus on at the numerator. Note that the number of virtual channels activated by link ℓ, i.e., x_ℓ, determines the probability of updating the schedule of the virtual channel. Further, note that the product operation on the left hand side of equation (10) implies that we proceed the operation with the order depending on the permutation vector $(n_1^\ell, \cdots, n_C^\ell)$. Similarly, for the right hand side of equation (10), we proceed the product operation with the order depending on the permutation vector $(n_C^\ell, \cdots, n_1^\ell)$. Hence, we

know that for the left hand side of equation (10), the value of x_ℓ right before updating the schedule of virtual channel n_k^ℓ is

$$x_\ell^1 + \sum_{i=1}^{k-1}(V_{\ell,n_i^\ell}^2 - V_{\ell,n_i^\ell}^1). \tag{11}$$

Similarly, for the right hand side of equation (10), the value of x_ℓ right before updating the schedule of virtual channel n_k^ℓ is

$$x_\ell^2 + \sum_{i=C}^{k+1}(V_{\ell,n_i^\ell}^1 - V_{\ell,n_i^\ell}^2). \tag{12}$$

Also, we have the equality between x_ℓ^1 and x_ℓ^2

$$x_\ell^2 = x_\ell^1 + \sum_{i=1}^{C}(V_{\ell,n_i^\ell}^2 - V_{\ell,n_i^\ell}^1). \tag{13}$$

From equations (3), (11), and (12), we know that the numerator on the left hand side of equation (10) is equal to

$$f_\ell(x_\ell^1)\prod_{k=1}^{C} f_\ell(x_\ell^1 + \sum_{i=1}^{k}(V_{\ell,n_i^\ell}^2 - V_{\ell,n_i^\ell}^1)). \tag{14}$$

Similarly, we know that the numerator on the right hand side of equation (10) is equal to

$$f_\ell(x_\ell^2)\prod_{k=C}^{1} f_\ell(x_\ell^2 + \sum_{i=C}^{k}(V_{\ell,n_i^\ell}^1 - V_{\ell,n_i^\ell}^2)). \tag{15}$$

It then follows from equation (13) that equation (14) equals to equation (15). Hence, the numerator on the left hand side of equation (10) equals to the numerator on the right hand side of equation (10). Now, we focus on the denominator of equation (10). We will show that the denominator of P_{ℓ,n_k^ℓ} equal to that of P'_{ℓ,n_k^ℓ} for all $1 \le k \le C$. Specifically, from equations (3), (11), and (12), we know that the denominator of P_{ℓ,n_k^ℓ} is

$$\begin{aligned} & f_\ell(x_\ell^1 + \textstyle\sum_{i=1}^{k-1}(V_{\ell,n_i^\ell}^2 - V_{\ell,n_i^\ell}^1)) \\ + \; & f_\ell(x_\ell^1 + \textstyle\sum_{i=1}^{k-1}(V_{\ell,n_i^\ell}^2 - V_{\ell,n_i^\ell}^1) \\ & \quad +1_{\{V_{\ell,n_k^\ell}^1=0\}} - 1_{\{V_{\ell,n_k^\ell}^1=1\}}), \end{aligned} \tag{16}$$

and the denominator of P'_{ℓ,n_k^ℓ} is

$$\begin{aligned} & f_\ell(x_\ell^2 + \textstyle\sum_{i=C}^{k+1}(V_{\ell,n_i^\ell}^1 - V_{\ell,n_i^\ell}^2)) \\ + \; & f_\ell(x_\ell^2 + \textstyle\sum_{i=C}^{k+1}(V_{\ell,n_i^\ell}^1 - V_{\ell,n_i^\ell}^2) \\ & \quad +1_{\{V_{\ell,n_k^\ell}^2=0\}} - 1_{\{V_{\ell,n_k^\ell}^2=1\}}). \end{aligned} \tag{17}$$

Let $y = x_\ell^1 + \sum_{i=1}^{k-1}(V_{\ell,n_i^\ell}^2 - V_{\ell,n_i^\ell}^1)$. We can rewrite equation (16) as

$$f_\ell(y) + f_\ell(y + 1_{\{V_{\ell,n_k^\ell}^1=0\}} - 1_{\{V_{\ell,n_k^\ell}^1=1\}}). \tag{18}$$

By equation (13), we can also rewrite equation (17) as

$$\begin{aligned} & f_\ell(y + V_{\ell,n_k^\ell}^2 - V_{\ell,n_k^\ell}^1) \\ + \; & f_\ell(y + V_{\ell,n_k^\ell}^2 - V_{\ell,n_k^\ell}^1 + 1_{\{V_{\ell,n_k^\ell}^2=0\}} - 1_{\{V_{\ell,n_k^\ell}^2=1\}}). \end{aligned} \tag{19}$$

It is then easy to show that under all four possible conditions, i.e, $(V_{\ell,n_k^\ell}^2, V_{\ell,n_k^\ell}^1) = (0,0),(0,1),(1,0)$, or $(1,1)$, equation (18) is equal to equation (19). Hence, the denominator of P_{ℓ,n_k^ℓ} is equal to that of P'_{ℓ,n_k^ℓ}.

Now, we have proved that equation (10) is true. This implies that equation (8) is true, and we have verified the local balance equations. This concludes the proof.

APPENDIX B
PROOF OF PROPOSITION 2

We first prove one lemma that is required for proving Proposition 2. We omit the proofs due to the space constraints. Details can be found in the technical report [25].

Lemma 8: If $0 < \epsilon \le 0.1$ and $\epsilon < r < 1$,

$$(1-r+\epsilon)\log(\frac{1-r+\epsilon}{1-r}) + (r-\epsilon)\log(\frac{r-\epsilon}{r}) > \frac{3}{2}\epsilon^2$$

Now, we can prove Proposition 2.

Proof of Proposition 2: Recall that $\vec{R}^* = [R_\ell^*]$ is the optimal rate allocation of problem (1). By Caratheodory's Theorem [23], $\vec{R}^*$ can be represented as a convex combination of $L+1$ feasible schedules $\vec{S}_1, \vec{S}_2, \cdots, \vec{S}_{L+1}$, i.e., $\vec{R}^* = \sum_{i=1}^{L+1}\alpha_i\vec{S}_i$, where $\alpha_i \ge 0$ and $\sum_{i=1}^{L+1}\alpha_i = 1$. Now, consider the following random system. There are C trials. For each trial $k = 1, \cdots, C$, choose a $\vec{S}$ from $\vec{S}_1, \cdots, \vec{S}_{L+1}$ based on the probability distribution $P(\{\vec{S} = \vec{S}_i\}) = \alpha_i$, for all i, independently across k. Let r_ℓ^s be the number of times link ℓ is scheduled by this random set of C schedules divided by C. Then, in order to show that there exists $\vec{V}^s$ such that $r_\ell(\vec{V}^s) \ge R_\ell^* - \epsilon$, for all link ℓ, it is sufficient to show that the following event occurs with a positive probability: $R_\ell^* - r_\ell^s \le \epsilon$ for all links ℓ. Note that by union bound, we have $P(\bigcap_\ell\{R_\ell^* - r_\ell^s \le \epsilon\}) \ge 1 - \sum_{\ell=1}^{L}P(R_\ell^* - r_\ell^s > \epsilon)$. Hence, it is sufficient to show that

$$P(R_\ell^* - r_\ell^s > \epsilon) < \frac{1}{L}, \text{ for all links } \ell. \tag{20}$$

Consider a fixed link ℓ. Intuitively, since the probability that link ℓ is active in each trial is equal to R_ℓ^*, *i.i.d.* across trials, (20) should hold when C is sufficiently large. Precisely, note that (20) holds trivially if $R_\ell^* \le \epsilon$. Further, if $R_\ell^* = 1$, then link ℓ will be activated by all schedules $\vec{S}_i$, and $P(R_\ell^* - r_\ell^s > \epsilon) = 0$. Hence, we only need to consider the possibility that

$$\epsilon < R_\ell^* < 1. \tag{21}$$

Let h be a random variable that represents the number of times that link ℓ is activated by the series of C random schedules, and $r_\ell^s = \frac{h}{C}$. Note that h is a Binomial random variable, and its moment generating function is $E[e^{-\frac{ht}{C}}] = (R_\ell^* e^{-\frac{t}{C}} + 1 - R_\ell^*)^C$. Using the Chernoff bound, we then have, for all $t > 0$,

$$\begin{aligned} P(R_\ell^* - r_\ell^s > \epsilon) &= P(-\tfrac{h}{C} > \epsilon - R_\ell^*) \\ &\le e^{t(R_\ell^* - \epsilon)}E[e^{-\frac{ht}{C}}] = e^{f(t)}, \end{aligned} \tag{22}$$

where $f(t) = t(R_\ell^* - \epsilon) + C\log(R_\ell^* e^{-\frac{t}{C}} + 1 - R_\ell^*), \forall t > 0$. To get the best bound, we solve the minimum of

$$f(t) = t(R_\ell^* - \epsilon) + C\log(R_\ell^* e^{-\frac{t}{C}} + 1 - R_\ell^*), \forall t > 0.$$

We first observe that

$$f''(t) = \frac{\frac{R_\ell^*}{C}e^{-\frac{t}{C}}(1 - R_\ell^*)}{(R_\ell^* e^{-\frac{t}{C}} + 1 - R_\ell^*)^2} > 0, \forall t > 0,$$

which follows from equation (21). Therefore, $f(t)$ is strictly convex, and there is one global minimum. Hence, we find the minimum by solving

$$\begin{aligned} & f'(t) = (R_\ell^* - \epsilon) + \frac{-R_\ell^* e^{-\frac{t}{C}}}{R_\ell^* e^{-\frac{t}{C}} + 1 - R_\ell^*} = 0 \\ & \Leftrightarrow e^{-\frac{t}{C}} = \frac{(1-R_\ell^*)(R_\ell^* - \epsilon)}{R_\ell^*(1-R_\ell^*+\epsilon)}. \end{aligned} \tag{23}$$

Substituting (23) into (22), we can then show that

$$\begin{aligned} & P(R_\ell^* - r_\ell^s > \epsilon) \\ \leq & \left[\left(\frac{R_\ell^* - \epsilon}{R_\ell^*} \right)^{-(R_\ell^* - \epsilon)} \left(\frac{1-R_\ell^*}{1-R_\ell^*+\epsilon} \right)^{1-R_\ell^*+\epsilon} \right]^C. \end{aligned} \tag{24}$$

Hence, a sufficient condition of (20) to hold is

$$C > \frac{\log L}{(1-R_\ell^*+\epsilon)\log(\frac{1-R_\ell^*+\epsilon}{1-R_\ell^*}) + (R_\ell^* - \epsilon)\log(\frac{R_\ell^* - \epsilon}{R_\ell^*})}. \tag{25}$$

It then follows from $C \geq \frac{2\log L}{3\epsilon^2}$, Lemma 8, equation (25), and equation (24) that $P(R_\ell^* - r_\ell^s > \epsilon) < \frac{1}{L}$ if $\epsilon < R_\ell^* < 1$. Hence, equation (20) is true, and we finish the proof. ∎

APPENDIX C
PROOF OF PROPOSITION 6

We first prove one lemma that is required for proving Proposition 6. In the following lemma, we assume that $0 < h < \frac{1}{(\Delta+2)(\mathcal{K}-1)}$. We omit the proofs due to the space constraints. Details can be found in the technical report [25].

Lemma 9: Consider link ℓ and n links $\ell_1, \ell_2, \cdots, \ell_n$, where link ℓ interferes with all n links, but these n links do not interfere with each other. Suppose that the rate of link ℓ is r_ℓ, and the rate of link ℓ_i is r_{ℓ_i}. Further, suppose that $n \leq \mathcal{K}_\ell$, and the utility function of all links is $\log(\cdot + h) - \log(h)$.

1) Suppose that $\frac{1}{C} < \frac{1}{(\Delta_\ell+2)(1+\mathcal{K}_\ell)} - \frac{(\mathcal{K}_\ell-1)h}{\mathcal{K}_\ell+1}$ and $\frac{1}{C} \leq \frac{1}{\Delta_\ell+2}$. If $r_\ell = 0$ and $r_{\ell_i} \geq \frac{1}{\Delta_\ell+2}$, for all i, then $U_\ell(r_\ell + \frac{1}{C}) + \sum_{i=1}^n U_{\ell_i}(r_{\ell_i} - \frac{1}{C}) > U_\ell(r_\ell) + \sum_{i=1}^n U_{\ell_i}(r_{\ell_i})$.
2) Suppose that $\frac{1}{C} < \frac{1}{\mathcal{K}_\ell(\Delta_\ell+2)} - \frac{h(\mathcal{K}_\ell-1)}{\mathcal{K}_\ell}$. If $0 < r_\ell < \frac{1}{\mathcal{K}_\ell(\Delta_\ell+2)} - \frac{h(\mathcal{K}_\ell-1)}{\mathcal{K}_\ell} - \frac{\Delta_\ell+1}{C}$ and $r_{\ell_i} \geq \frac{1}{\Delta_\ell+2} + \frac{1}{C}$, then $U_\ell(r_\ell + \frac{1}{C}) + \sum_{i=1}^n U_{\ell_i}(r_{\ell_i} - \frac{1}{C}) > U_\ell(r_\ell) + \sum_{i=1}^n U_{\ell_i}(r_{\ell_i})$.

Proof of Proposition 6: We prove this proposition by contradiction. Suppose that there is a link ℓ with $r_\ell^{\min} < \frac{1}{\mathcal{K}_\ell(\Delta_\ell+2)} - \frac{h(\mathcal{K}_\ell-1)}{\mathcal{K}_\ell} - \frac{\Delta_\ell+1}{C}$. Let $I_\ell = \mathcal{E}_\ell \bigcup \{\ell\}$. This implies that there is a $\vec{V}$ such that $\vec{V} \in \mathcal{V}$, $U(\vec{r}(\vec{V})) = U(\vec{r}(\vec{V}^{\max}))$, i.e., $U(\vec{r}(\vec{V})) \geq U(\vec{r}(\vec{V}'))$, for all $\vec{V}' \in \mathcal{V}$, and $r_\ell(\vec{V}) < \frac{1}{\mathcal{K}_\ell(\Delta_\ell+2)} - \frac{h(\mathcal{K}_\ell-1)}{\mathcal{K}_\ell} - \frac{\Delta_\ell+1}{C}$. In the sequel, we refer to the rate $r_\ell(\vec{V})$ as the optimal rate of each link ℓ. We will construct another solution $\vec{V}' \in \mathcal{V}$ such that $U(\vec{r}(\vec{V}')) > U(\vec{r}(\vec{V}))$, which then leads to a contradiction.

From the definition of $\vec{r}(\vec{V})$, $\vec{r}(\vec{V})$ can be described by the convex combination of C schedules, where each schedule has L dimensions. Denote these schedules as $\vec{S}_1, \cdots, \vec{S}_C$. Let a_i be the coefficient of $\vec{S}_i$. Note that $a_i = 1/C$ for all i. Let $\mathcal{S} = \{\vec{S}_i, 1 \leq i \leq C\}$. Now, we can truncate the schedule $\vec{S}_i$ to the set I_ℓ, i.e., we only keep the elements of $\vec{S}_i$ that correspond to links in I_ℓ. Let $\vec{S}_i'$ be the truncated version of $\vec{S}_i$. We then know that the optimal rate for all links ℓ in I_ℓ can also be described by the convex combination of C schedules, i.e., $\vec{S}_i', 1 \leq i \leq C$, with dimension $\Delta_\ell + 1$. Let $\mathcal{S}' = \{\vec{S}_i', 1 \leq i \leq C\}$. Next, we will show that the optimal rate for all links ℓ in I_ℓ can actually be described by the convex combination of k nonempty schedules from $\mathcal{S}'$ with positive coefficient, where k is at most $\Delta_\ell + 2$. First, $\vec{S}_i'$ is not empty for each i. Otherwise, we can include link ℓ in $\vec{S}_i$, and we have another global schedule with larger utility. Second, we can follow the proof of the Caratheodory's theorem to describe the optimal rate in I_ℓ as the convex combination of $\Delta_\ell + 2$ schedules in $\mathcal{S}'$ with positive coefficients. Without loss of generality, let these schedules be $\vec{S}_1', \cdots, \vec{S}_k', k \leq \Delta_\ell + 2$. Let a_i' be the weight of $\vec{S}_i', 1 \leq i \leq k$. We know that $\sum_{i=1}^k a_i' = 1, a_i' > 0$. Note that for each $\vec{S}_i'$, there exists at least one corresponding untruncated schedule $\vec{S}_i$ with a positive coefficient, i.e., $a_i = 1/C > 0$, and its truncation in I_ℓ equal to $\vec{S}_i'$. Consider the following two cases.

Case 1: $r_\ell(\vec{V}) = 0$. In this case, we know that all these k schedules do not schedule link ℓ. Further, since $\sum_{i=1}^k a_i' = 1$, we can conclude that one of the k schedules must have weight no smaller than $\frac{1}{\Delta_\ell+2}$. Without loss of generality, let this schedule be $\vec{S}_1'$, and we have that $a_1' \geq \frac{1}{\Delta_\ell+2}$. This implies that the links scheduled by $\vec{S}_1'$ have optimal rate no smaller than $\frac{1}{\Delta_\ell+2}$. Recall that $\vec{S}_1$ is the corresponding untruncated schedule with coefficient $a_1 = 1/C$. Now, construct a schedule $\vec{S}$ such that its schedule for link ℓ is the same as $\vec{S}_1$ if $\ell \notin I_\ell$, and it only schedules link ℓ in I_ℓ. From the condition $\frac{\Delta_\ell+1}{C} < \frac{1}{\mathcal{K}_\ell(\Delta_\ell+2)} - \frac{h(\mathcal{K}_\ell-1)}{\mathcal{K}_\ell}$, we can show that $\frac{1}{C} < \frac{1}{(\Delta_\ell+2)(1+\mathcal{K}_\ell)} - \frac{(\mathcal{K}_\ell-1)h}{\mathcal{K}_\ell+1}$ and $\frac{1}{C} \leq \frac{1}{\Delta_\ell+2}$.

Consider another solution $\vec{V}'$ such that $\vec{r}(\vec{V}') = \sum_{i=2}^C \frac{1}{C}\vec{S}_i + \frac{1}{C}\vec{S}$. Suppose that there are n links scheduled in $\vec{S}_1'$. Denote these n links as ℓ_i, $i = 1, 2, \cdots, n$. Note that $n \leq \mathcal{K}_\ell$. Then $U(\vec{r}(\vec{V}')) - U(\vec{r}(\vec{V})) = U_\ell(r_\ell(\vec{V}) + \frac{1}{C}) - U_\ell(r_\ell(\vec{V})) - (\sum_{i=1}^n U_{\ell_i}(r_{\ell_i}(\vec{V})) - U_{\ell_i}(r_{\ell_i}(\vec{V}) - \frac{1}{C}))$. For each link ℓ_i scheduled in $\vec{S}_1'$, we know that $r_{\ell_i}(\vec{V}) \geq \frac{1}{\Delta_\ell+2}$. By Lemma 9 (1), we then know that $U(\vec{r}(\vec{V}')) - U(\vec{r}(\vec{V})) > 0$. This implies that the utility of $\vec{r}(\vec{V}')$ is larger than the utility of $\vec{r}(\vec{V})$, which contradicts with the assumption that $U(\vec{r}(\vec{V})) = U(\vec{r}(\vec{V}^{\max}))$.

Case 2: $r_\ell(\vec{V}) > 0$. We know that every feasible schedule that schedules at least one link in $\mathcal{E}_\ell$ can not schedule link ℓ because every link in $\mathcal{E}_\ell$ interferes with link ℓ. Further, every schedule that schedules link ℓ can not schedule any links in $\mathcal{E}_\ell$. Hence, $r_\ell(\vec{V})$ is equal to the sum of the coefficient of the k schedules that only schedule link ℓ, and there exists at lease one schedule (from the k schedules) that schedules link ℓ. We then know that there are at most $\Delta_\ell + 1$ schedules (from the k schedules) that do not schedule link ℓ. (Note that this step does not hold in Case 1.) Since $r_\ell(\vec{V}) < \frac{1}{\mathcal{K}_\ell(\Delta_\ell+2)} - \frac{h(\mathcal{K}_\ell-1)}{\mathcal{K}_\ell} - \frac{\Delta_\ell+1}{C}$, we know that the sum of the weight of the rest k' schedules, $0 < k' \leq \Delta_\ell + 1$, is larger than $\frac{\mathcal{K}_\ell(\Delta_\ell+2)-1}{\mathcal{K}_\ell(\Delta_\ell+2)} + \frac{h(\mathcal{K}_\ell-1)}{\mathcal{K}_\ell} + \frac{\Delta_\ell+1}{C}$. Hence, we can conclude that one of the remaining k' schedules must have weight larger than $\frac{\mathcal{K}_\ell(\Delta_\ell+2)-1}{\mathcal{K}_\ell(\Delta_\ell+2)(\Delta_\ell+1)} + \frac{h(\mathcal{K}_\ell-1)}{\mathcal{K}_\ell(\Delta_\ell+1)} + \frac{1}{C}$, which can be shown to be no smaller than $\frac{1}{\Delta_\ell+2} + \frac{1}{C}$. Without loss of generality, let this schedule be $\vec{S}_1'$, and we have that $a_1' > \frac{1}{\Delta_\ell+2} + \frac{1}{C}$. This implies

that the links scheduled by $\vec{S}_1'$ have rate larger than $\frac{1}{\Delta_\ell+2}+\frac{1}{C}$. Recall that $\vec{S}_1$ is the corresponding untruncated schedule with coefficient $a_1 = 1/C$. Now, construct a schedule $\vec{S}$ such that its schedule for link ℓ is the same as $\vec{S}_1$ if $\ell \notin I_\ell$, and it only schedules link ℓ in I_ℓ. Suppose that there are n links scheduled in $\vec{S}_1'$. Denote these n links as ℓ_i, $i = 1, 2, \cdots, n$. Note that $n \leq \mathcal{K}_\ell$. Recall that $0 < r_\ell(\vec{V}) < \frac{1}{\mathcal{K}_\ell(\Delta_\ell+2)} - \frac{h(\mathcal{K}_\ell-1)}{\mathcal{K}_\ell} - \frac{\Delta_\ell+1}{C}$, and from $a_1' > \frac{1}{\Delta_\ell+2}+\frac{1}{C}$, we have that $\frac{1}{\Delta_\ell+2}+\frac{1}{C} \leq r_{\ell_i}(\vec{V})$ for any link ℓ_i scheduled in $\vec{S}_1'$.

Consider another solution $\vec{V}'$ such that $\vec{r}(\vec{V}') = \sum_{i=2}^{C} \frac{1}{C}\vec{S}_i + \frac{1}{C}\vec{S}$. Then $U(\vec{r}(\vec{V}')) - U(\vec{r}(\vec{V})) = U_\ell(r_\ell(\vec{V}) + \frac{1}{C}) - U_\ell(r_\ell(\vec{V})) - (\sum_{i=1}^{n} U_{\ell_i}(r_{\ell_i}(\vec{V})) - U_{\ell_i}(r_{\ell_i}(\vec{V}) - \frac{1}{C}))$. By Lemma 9 (2), we then know that $U(\vec{r}(\vec{V}')) - U(\vec{r}(\vec{V})) > 0$. This implies that the utility of $\vec{r}(\vec{V}')$ is larger than the utility of $\vec{r}(\vec{V})$, which contradicts with the assumption that $U(\vec{r}(\vec{V})) = U(\vec{r}(\vec{V}^{\max}))$.

Since for both cases, the assumption $r_\ell(\vec{V}) < \frac{1}{\mathcal{K}_\ell(\Delta_\ell+2)} - \frac{h(\mathcal{K}_\ell-1)}{\mathcal{K}_\ell} - \frac{\Delta_\ell+1}{C}$ leads to a contradiction, we conclude that $r_\ell^{\min} \geq \frac{1}{\mathcal{K}_\ell(\Delta_\ell+2)} - \frac{h(\mathcal{K}_\ell-1)}{\mathcal{K}_\ell} - \frac{\Delta_\ell+1}{C}$. This concludes the proof. ■

REFERENCES

[1] P.-K. Huang and X. Lin, "Improving the Delay Performance of CSMA Algorithms: A Virtual Multi-Channel Approach," in *IEEE INFOCOM*, 2013, pp. 2598 – 2606.

[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, 1992.

[3] X. Lin, N. B. 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, 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. 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, 2010.

[6] 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, 2012.

[7] C. Borgs, J. T. Chayes, A. Frieze, J. H. Kim, P. Tetali, E. Vigoda *et al.*, "Torpid mixing of some Monte Carlo Markov chain algorithms in statistical physics," in *IEEE FOCS*, 1999, pp. 218–229.

[8] M. Lotfinezhad and P. Marbach, "Throughput-Optimal Random Access with Order-Optimal Delay," in *IEEE INFOCOM*, 2011, pp. 2867–2875.

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

[10] S. Sarkar and S. Ray, "Arbitrary Throughput Versus Complexity Tradeoffs in Wireless Networks using Graph Partitioning," *IEEE Trans. Autom. Control*, vol. 53, no. 10, pp. 2307–2323, 2008.

[11] D. Shah and J. Shin, "Delay optimal queue-based CSMA," in *ACM SIGMETRICS Performance Evaluation Review*, vol. 38, no. 1, 2010, pp. 373–374.

[12] L. Jiang, M. Leconte, J. Ni, R. Srikant, and J. Walrand, "Fast Mixing of Parallel Glauber Dynamics and Low-Delay CSMA Scheduling," *IEEE Trans. Inf. Theory*, vol. 58, no. 10, pp. 6541 – 6555, 2012.

[13] V. G. Subramanian and M. Alanyali, "Delay performance of CSMA in networks with bounded degree conflict graphs," in *IEEE ISIT*, 2011, pp. 2373–2377.

[14] B. Li and A. Eryilmaz, "Optimal Distributed Scheduling under Time-Varying Conditions: A Fast-CSMA Algorithm with Applications," *IEEE Trans. Wireless Commun.*, vol. 12, no. 7, pp. 3278 – 3288, 2013.

[15] C.-H. Lee, D. Y. Eun, S.-Y. Yun, and Y. Yi, "From Glauber Dynamics To Metropolis Algorithm: Smaller Delay in Optimal CSMA," in *IEEE ISIT*, 2012, pp. 2681–2685.

[16] K.-K. Lam, C.-K. Chau, M. Chen, and S.-C. Liew, "Mixing Time and Temporal Starvation of General CSMA Networks with Multiple Frequency Agility," in *IEEE ISIT*, 2012, pp. 2676–2680.

[17] D. Xue and E. Ekici, "On Reducing Delay and Temporal Starvation of Queue-Length-Based CSMA Algorithms," in *Proc. 50th Annual Allerton Conference on Communication, Control, and Computing*, 2012, pp. 754–761.

[18] A. Proutiere, Y. Yi, T. Lan, and M. Chiang, "Resource Allocation over Network Dynamics without Timescale Separation," in *IEEE INFOCOM*, 2010, pp. 1–5.

[19] Y. Yi, G. de Veciana, and S. Shakkottai, "Learning contention patterns and adapting to load/topology changesin a mac scheduling algorithm," in *IEEE WiMesh*, 2006, pp. 23–32.

[20] D. Qian, D. Zheng, J. Zhang, N. B. Shroff, and C. Joo, "Distributed CSMA Algorithms for Link Scheduling in Multihop MIMO Networks Under SINR Model," *IEEE/ACM Trans. Netw.*, vol. 21, no. 3, June 2013.

[21] S. Zhou, X. Wu, and L. Ying, "Distributed Power Control and Coding-Modulation Adaptation in Wireless Networks using Annealed Gibbs Sampling," in *IEEE INFOCOM*, 2012, pp. 3016 – 3020.

[22] S. Boyd, *Convex Optimization*. Cambridge University Press, 2004.

[23] D. P. Bertsekas, A. Nedić, and A. E. Ozdaglar, *Convex Analysis and Optimization*. Athena Scientific Belmont, 2003.

[24] D. Gross and C. Harris, *Fundamentals of Queueing Theory*, 4th ed. John Wiley & Sons, 2008.

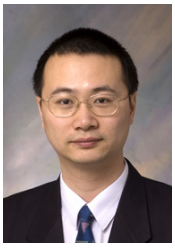
[25] P.-K. Huang and X. Lin, "Achieving Optimal Throughput Utility and Low Delay with CSMA-like Algorithms: A Virtual Multi-Channel Approach," *Technical Report, Purdue University*, 2014. [Online]. Available: http://docs.lib.purdue.edu/ecetr/455/

[26] D. E. Knuth, *Art of Computer Programming Volume 2: Seminumerical Algorithms*, 3rd ed. Addison-Wesley Publishing Company, 1997.

[27] S. M. Ross, *Stochastic Processes*, 2nd ed. New York: John Wiley & Son, 1996.

[28] A. Gupta, X. Lin, and R. Srikant, "Low-Complexity Distributed Scheduling Algorithms for Wireless Networks," *IEEE/ACM Trans. Netw.*, vol. 17, no. 6, pp. 1846–1859, 2009.

Po-Kai Huang received his B.S. and M.S. in Electrical and Computer Engineering from National Tsing Hua University, Taiwan, in 2004 and 2006, respectively. He received his Ph.D. degree in Electrical and Computer Engineering from Purdue University, Indiana, in 2013. He is currently working as system engineer at Intel corporation. His research interests are in the area of wireless networks including wireless cross-layer control, resource allocation, and delay performance analysis.



Xiaojun Lin (S'02 / M'05 / SM'12) received his B.S. from Zhongshan University, Guangzhou, China, in 1994, and his M.S. and Ph.D. degrees from Purdue University, West Lafayette, Indiana, in 2000 and 2005, respectively. He is currently an Assistant Professor of Electrical and Computer Engineering at Purdue University.

Dr. Lin's research interests are in the analysis, control and optimization of wireless and wireline communication networks. He received the IEEE INFOCOM 2008 best paper award and 2005 best paper of the year award from *Journal of Communications and Networks*. His paper was also one of two runner-up papers for the best-paper award at IEEE INFOCOM 2005. He received the NSF CAREER award in 2007. He was the Workshop co-chair for IEEE GLOBECOM 2007, the Panel co-chair for WICON 2008, the TPC co-chair for ACM MobiHoc 2009, and the Mini-Conference co-chair for IEEE INFOCOM 2012. He is currently serving as an Area Editor for (Elsevier) Computer Networks journal, and has served as a Guest Editor for (Elsevier) Ad Hoc Networks journal.