Implementing Utility-Optimal CSMA

Jinsung Lee, Junhee Lee, Yung Yi, Song Chong*, Alexandre Proutière†, Mung Chiang‡
*Dept. of Electrical Engineering, KAIST, South Korea
Email: {ljs,junhee}@netsys.kaist.ac.kr, {yiyung,song}@ee.kaist.ac.kr
†Microsoft Research Lab, UK
Email: alexandre.proutiere@microsoft.com
‡Dept. of Electrical Engineering, Princeton University, USA
Email: chiangm@princeton.edu

***Abstract*—Hundreds of papers over the last two decades have studied the theory of distributed scheduling in wireless networks, including a number of them on stability or utility maximizing random access. Several publications in 2008 studied an adaptive CSMA that can approach utility optimality without any message passing under a number of assumptions. This paper reports the results from the first deployment of such random access algorithms through an implementation over conventional 802.11 hardware, an on-going effort that started in summer 2009. It shows both a confirmation that Utility Optimal CSMA may work well in the practice even with implementation over legacy equipment, and a wide array of gaps between theory and practice in the field of wireless scheduling. This paper therefore also brainstorms the discovery of and bridging over these gaps, and the implementation-inspired questions on modeling and analysis of scheduling algorithms.**

I. Introduction

Design of distributed scheduling algorithms in wireless networks has been extensively studied under various metrics of efficiency and fairness. In their seminal work [1], Tassiulas and Ephremides developed a centralized scheduling algorithm, Max-Weight scheduling, achieving throughput optimality, i.e., stabilizing any arrival for which there exists a stabilizing scheduler. Since then, there has been a large array of lower-complexity, more distributed scheduling algorithms, using the ideas of randomization (pick-and-compare scheduling), weight approximation (maximal/greedy scheduling), or random access with queue-length exchanges, e.g., in [2]–[11], to achieve large stability region under unsaturated arrivals of traffic at each node in the network. For saturated arrivals, optimizing a utility function, which captures efficiency and fairness at the equilibrium, has been studied for slotted-Aloha random access, e.g., in [12]–[17]. Together with the principle of Layering as Optimization Decomposition, advances in scheduling have also been translated into improvements in joint congestion control, routing, and scheduling over multihop wireless networks, e.g., [18]–[22]. There are many more studies on this topic, as discussed in more detail in surveys such as [23].

A main bottleneck that remains is the need for message passing in the above algorithms. Tradeoffs of the time complexity of message passing with throughput and delay have been studied recently in [6], [7], [24] and [25]. Message passing reduces "effective" performance, is vulnerable to security attacks, and makes the algorithms not fully distributed. This naturally leads to the following question on simplicity-driven design: *Can random access without message passing approach some type of performance optimality?* The answer was suggested to be positive last year, first in [26] for wireless network, with a similar development in a different context in [27]. Convergence proof and tradeoff were presented in [28].

In [28], we extended the algorithm in [26], and developed a rigorous proof of the convergence of these algorithms, without assuming that network dynamics freeze while the CSMA parameters are being updated, for the continuous-time Poisson clock model. New proof techniques are developed to overcome the difficulty of the coupling between the control of CSMA parameters and the queueing network dynamics. We then turned to more realistic discrete-time contention and backoff model, and quantified the effect of collisions. We revealed and characterized the tradeoff between long-term efficiency and short-term fairness: short-term fairness decreases significantly as efficiency loss is reduced. Similarly to other distributed scheduling algorithms, there is a 3-dimensional tradeoff [24]: the price of optimality and zero message passing here is delay experienced by some nodes.

This paper reports the results from the first deployment of such random access algorithms through an implementation over conventional 802.11 hardware, an on-going effort that started in summer 2009. It shows both a confirmation that Utility Optimal CSMA may work well in the practice even with implementation over legacy equipment, and a wide array of gaps between theory and practice in the field of wireless scheduling. This paper therefore also brainstorms the discovery of and bridging over these gaps, and the implementation-inspired questions on modeling and analysis of scheduling algorithms. This is an "interim report" of the ongoing experiments, and further results will be presented in the future.

The rest of this paper is organized as follows: In Section II, we briefly describe the theoretical foundation of UO-CSMA. Section III presents the experiments and their analysis in implementation, followed by identifying the gaps between theory and practice in Section IV. Implementation-inspired theory and next steps of experimentation are outlined in Sections V and VI. We conclude the paper in Section VII.

II. Theory: Utility Optimal CSMA

In this section, we summarize UO-CSMA (Utility-Optimal CSMA) that approximately achieves optimality in terms of utility. We refer the readers to [28] for details.

A. Network and interference model

We consider a wireless network composed by a set $\mathcal{L}$ of L links. Interference is modeled by a symmetric, boolean matrix $A \in \{0,1\}^{L\times L}$, where $A_{kl} = 1$ if link k interferes link l, and $A_{kl} = 0$ otherwise. Define by $\mathcal{N} \subset \{0,1\}^L$ the set of the N feasible link activation profiles, or schedules. A schedule $m \in \mathcal{N}$ is a subset of non-interfering active links (i.e., for any $m \in \mathcal{N}$, $k, l \in m$, $A_{kl} = 0$). We assume that the transmitters can transmit at a fixed unit rate when active.

B. Scheduling and utility maximization

The network is assumed to handle single-hop data connections. However, the results presented here can be easily extended to multi-hop connections (e.g., using classical *back-pressure* ideas [1]). The transmitter of each link is saturated, i.e., it always has packets to send. A scheduling algorithm decides at each time which links are activated. Denote by $\boldsymbol{\gamma}^s = (\gamma_l^s, l \in \mathcal{L})$ the long-term throughputs achieved by scheduling algorithm s. The throughput vector of any scheduling algorithm has to belong to the *rate region* Γ defined by

$$\Gamma = \{\boldsymbol{\gamma} \in \mathbb{R}_+^L : \exists \boldsymbol{\pi} \in \mathbb{R}_+^N, \\ \forall l \in \mathcal{L}, \gamma_l \leq \sum_{m\in\mathcal{N}:m_l=1} \pi_m, \quad \sum_{m\in\mathcal{N}} \pi_m = 1\}.$$

In the above, for any schedule $m \in \mathcal{N}$, π_m can be interpreted as the proportion of time schedule m is activated. As is a standard in problems with saturated arrivals, the objective is to design a scheduling algorithm maximizing the total network-wide utility. Specifically, let $U : \mathbb{R}^+ \to \mathbb{R}$ be an increasing, strictly concave, differentiable objective function. We wish to design an algorithm to solve the following optimization problem:

$$\begin{aligned} \max \quad & \Sigma_{l\in\mathcal{L}} U(\gamma_l), \\ \text{s.t.} \quad & \boldsymbol{\gamma} \in \Gamma. \end{aligned} \tag{1}$$

We denote by $\boldsymbol{\gamma}^\star = (\gamma_l^\star, l \in \mathcal{L})$ the optimizer of (1). Most distributed schemes proposed in the literature to date to solve (1) make use of a dual decomposition of the problem into a rate control and a scheduling problem: A virtual queue is associated with each link; a rate control algorithm defines the rate at which packets are sent to the virtual queues, and a scheduling algorithm decides, depending on the level of the virtual queues, which schedule to use with the aim of stabilizing all virtual queues. The main challenge reduces to developing a distributed and efficient scheduling algorithm. Many solutions proposed so far are semi-distributed implementations of the max-weight scheduler introduced in [1], and require information about the queues to be passed around among the nodes or links (e.g., see a large set of references in [23]). This signaling overhead increases communication complexity and reduces effective throughput.

C. Efficiency of CSMA

CSMA-based random access is the most popularly used distributed scheduling algorithms in wireless networks. They are based on random back-off algorithms such as the Decentralized Coordinated Function (DCF) in IEEE802.11. The two basic principles behind CSMA schemes are (i) to detect whether the channel is busy before transmitting, and to refrain from starting a transmission when the channel is sensed busy, and (ii) to wait a random period of time before any transmission to limit the probability of collisions.

The network dynamics under CSMA have been extensively studied in the literature. The following popular model is due to Kelly [29], and has been recently revisited by e.g. [30] and [31]. In this model, the transmitter of link l waits an exponentially distributed random period of time with mean $1/\lambda_l$ before transmitting, and when it initiates a transmission, it keeps the channel for an exponentially distributed period of time with mean μ_l. This CSMA algorithm is denoted by CSMA(λ_l, μ_l) in the rest of the paper. Define $\boldsymbol{\lambda} = (\lambda_l, l \in \mathcal{L})$ and $\boldsymbol{\mu} = (\mu_l, l \in \mathcal{L})$. When each link l runs CSMA(λ_l, μ_l), the network dynamics can be captured through a reversible process [32]: If $m^{\boldsymbol{\lambda},\boldsymbol{\mu}}(t)$ denotes the active schedule at time t, then $(m^{\boldsymbol{\lambda},\boldsymbol{\mu}}(t), t \geq 0)$ is a continuous-time reversible Markov chain whose stationary distribution $\boldsymbol{\pi}^{\boldsymbol{\lambda},\boldsymbol{\mu}}$ is given by $\forall m \in \mathcal{N}, \quad \pi_m^{\boldsymbol{\lambda},\boldsymbol{\mu}} = \frac{\prod_{l:m_l=1}\lambda_l\mu_l}{\sum_{n\in\mathcal{N}}\prod_{l:n_l=1}\lambda_l\mu_l}$, where by convention $\prod_{l\in\emptyset}(\cdot) = 1$. It is worth noting that due to the reversibility of the process, the above stationary distribution does not depend on the distributions of the back-off durations or of the channel holding times, provided that they are of mean $1/\lambda_l$ and μ_l, respectively, for link l. This insensitivity property allows us to cover a more realistic scenario with uniformly distributed back-off delays and deterministic channel holding times.

Under the above continuous-time model, collisions are mathematically impossible, leading to tractability as a first step of the study. In practice, however, time is slotted and the back-off periods are multiple of slots, which inevitably causes collisions.

Under the CSMA(λ_l, μ_l)'s algorithms, the link throughputs are given by

$$\forall l \in \mathcal{L}, \quad \gamma_l^{\boldsymbol{\lambda},\boldsymbol{\mu}} = \sum_{m\in\mathcal{N}:m_l=1} \pi_m^{\boldsymbol{\lambda},\boldsymbol{\mu}}.$$

An important result, proved in [26] (Propositions 1 and 2), states that any throughput vector $\boldsymbol{\gamma} \in \Gamma$ can be *approached* using CSMA$(\boldsymbol{\lambda}, \boldsymbol{\mu})$ algorithms. More precisely, we have:

Lemma 1 ([26]): For any $\boldsymbol{\gamma}$ in the interior of Γ, there exist $\boldsymbol{\lambda}, \boldsymbol{\mu} \in \mathbb{R}_+^L$ such that $\forall l \in \mathcal{L}, \quad \gamma_l \leq \gamma_l^{\boldsymbol{\lambda},\boldsymbol{\mu}}$.

The above lemma expresses the *optimality* of CSMA scheduling schemes, and it suggests that for approaching the solution of (1), one may use CSMA algorithms.

D. Continuous time model: Algorithm and performance

We now describe a generic adaptive CSMA-based algorithm to approximately solve (1). The algorithm is an extension of those proposed in [26], and does not require any message

passing. Time is divided into *frames* of fixed durations, and the transmitters of each link update their CSMA parameters (i.e., λ_l, μ_l for link l) at the beginning of each frame. To do so, they maintain a virtual queue, denoted by $q_l[t]$ in frame t, for link l. The algorithm operates as follows:

UO-CSMA

1) During frame t, the transmitter of link l runs CSMA($\lambda_l[t], \mu_l[t]$), and records the amount $S_l[t]$ of service received during this frame;
2) At the end of frame t, it updates its virtual queue and its CSMA parameters according to

$$q_l[t+1] = \left[q_l[t] + \frac{b[t]}{W'(q_l[t])}\big(U'^{-1}(\frac{W(q_l[t])}{V}) - S_l[t]\big)\right]_{q^{\min}}^{q^{\max}},$$

and sets $\lambda_l[t+1]$ and $\mu_l[t+1]$ such that their product is equal to $\exp\{W(q_l[t+1])\}$.

In the above algorithm, $b : \mathbb{N} \to \mathbb{R}$ is a step size function; $W : \mathbb{R}^+ \to \mathbb{R}^+$ is a strictly increasing and continuously differentiable function, termed the *weight function*; V, $q^{\min}$, $q^{\max}(> q^{\min})$ are positive parameters, and $[\cdot]_c^d \equiv \min(d, \max(c, \cdot))$. We will later see that proper choice of b ensures convergence. V controls the accuracy of the algorithm, and the function W controls the transient behavior. The impact of $b[t], V$, and W will be demonstrated in the implementation in the next section.

Since the performance of CSMA algorithms depends on the products $\lambda_l\mu_l$ only, we have the choices in UO-CSMA to either update the λ_l's (the transmission intensities) and fix the μ_l's (the transmission durations), or to update the μ_l's and fix the λ_l's, or to update both the λ_l's and μ_l's.

E. Convergence

UO-CSMA may be interpreted as a stochastic approximation algorithm with *controlled Markov noise* as defined in [33]. The main difficulty in analyzing the convergence of UO-CSMA lies in the fact that the updates in the virtual queues, and hence in the CSMA parameters, depend on the random service processes $(S_l[t], t \geq 0)$. The service processes $(S_l[t], l \in \mathcal{L})$ received by the various links in turn depend on the state of the network at the end of frame $t-1$, and on the updated CSMA parameters $(\boldsymbol{\lambda}[t], \boldsymbol{\mu}[t])$.

For any vector $\boldsymbol{q} \in \mathbb{N}^L$, we denote by $\pi^{\boldsymbol{q}}$ the distribution on $\mathcal{N}$ resulting from the dynamics of the CSMA($\boldsymbol{\lambda}, \boldsymbol{\mu}$) algorithms, where for all $l \in \mathcal{L}$, $\lambda_l\mu_l = \exp(W(q_l))$. In other words,

$$\forall m \in \mathcal{N}, \quad \pi_m^{\boldsymbol{q}} = \frac{\exp(\sum_{l\in m} W(q_l))}{\sum_{m'\in\mathcal{N}} \exp(\sum_{l\in m'} W(q_l))}. \tag{2}$$

We also denote by $\boldsymbol{\gamma}[t] = (\gamma_l[t], l \in \mathcal{L})$ the vector representing the cumulative average throughputs of the various links up to frame t, i.e.,

$$\forall l \in \mathcal{L}, \quad \gamma_l[t] = \frac{1}{t}\sum_{n=0}^{t-1} S_l[n].$$

The next theorem states the convergence of UO-CSMA under diminishing step-sizes, towards a point that is arbitrarily close to the utility-optimizer.

Theorem 1: Assume $\sum_{t=0}^{\infty} b[t] = \infty$ and $\sum_{t=0}^{\infty} b[t]^2 < \infty$. For any initial condition $\boldsymbol{q}[0]$, UO-CSMA converges in the following sense:

$$\lim_{t\to\infty} \boldsymbol{q}[t] = \boldsymbol{q}_\star \text{ and } \lim_{t\to\infty} \boldsymbol{\gamma}[t] = \boldsymbol{\gamma}_\star, \text{ almost surely},$$

where $\gamma_\star$ and $\boldsymbol{q}_\star$ are such that $(\gamma_\star, \boldsymbol{\pi}^{q_\star})$ is the solution of the following convex optimization problem (over $\boldsymbol{\gamma}$ and $\boldsymbol{\pi}$):

$$\begin{aligned} \max \quad & V\sum_{l\in\mathcal{L}} U(\gamma_l) - \Sigma_{m\in\mathcal{N}}\pi_m \log \pi_m \\ \text{s.t.} \quad & \gamma_l \leq \sum_{m\in\mathcal{N}:m_l=1} \pi_m, \ \sum_{m\in\mathcal{N}} \pi_m = 1. \end{aligned} \tag{3}$$

Furthermore UO-CSMA approximately solves (1) as

$$\big|\sum_{l\in\mathcal{L}} \big(U(\gamma_{\star,l}) - U(\gamma_l^\star)\big)\big| \leq \log|\mathcal{N}|/V. \tag{4}$$

F. Slotted time model: Performance and tradeoff

In practical implementation, time is slotted and collisions may occur. In this section, we briefly summarize the impact of collisions (see [28] more details). We consider the following model for slotted CSMA: The transmitter of link l starts a transmission at the end of a slot with probability p_l if the slot has been sensed to be idle. When a link is active, it can experience either a successful transmission or a collision. When a link is currently successfully transmitting, it releases the channel with probability $1/\mu_l$ at the end of a slot. Collisions are classified into two kinds:

(a) *Short collisions.* Using a channel probing mechanism using a small signaling message, e.g., RTS/CTS, we restrict the length of collisions to a short time interval.
(b) *Long collisions.* Long collisions occur when RTS/CTS-like procedures are not implemented, so that collision time last for a maximum of holding times of links involved with collision.

If we want the resulting link throughputs of UO-CSMA to be close to the solution of (1), the products of the transmission probabilities and the channel holding times need to be very large. In the adaptation of UO-CSMA to the slotted-time scenario, this implies that the channel holding times are very large, since the transmission probabilities must remain very small (to ensure very low collision rates). This further implies that the delay between two successive successful transmissions on a link is very large as well. In other words, to ensure efficiency, we need to sacrifice *short-term fairness*.

Another source of short-term unfairness with UO-CSMA is the fact that if a link is interfering with by a lot of links (compared to other links), before transmitting it needs to wait until all its neighbors become inactive. This waiting time can be very long, especially if these neighbors do not sense each other. When the link finally gets access to the channel, it then needs to hold the channel for a duration that is much larger than the transmission durations of its neighbors, in order to

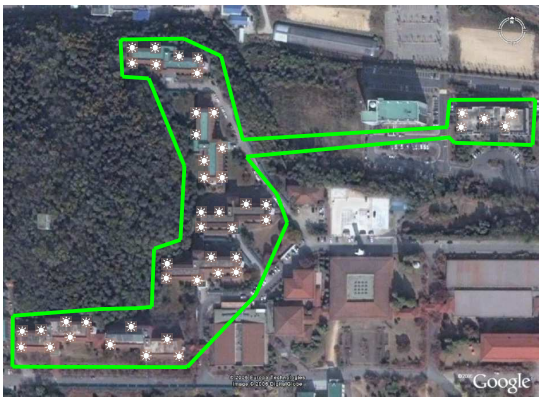


Fig. 1. WiMesh: Campus-wide wireless mesh network testbed at KAIST

achieve throughput fairness. This may considerably exacerbate short-term unfairness.

We now quantify the above two observations. We formally define the short-term fairness index of link l as $1/T_l$ where T_l is the average delay between two successive successful transmissions on this link. To illustrate, consider a simple star network: it is composed of $L+1$ links, where link 1 interferes with by all other links, but link k, $k>1$, interferes with only link 1. Throughout some computations (see [28] for details), it turns out that for a given efficiency loss $\epsilon > 0$, (i) channel holding times for link 1 and link k, $k>1$ scale as $1/\epsilon^{2L}$ and $1/\epsilon^2$, respectively, and (ii) the short-term fairness for all links scales as ϵ^{2L}. This quantifies the tradeoff between efficiency and short-term fairness when implementing UO-CSMA in slotted-time systems, implying that a huge cost of short-term unfairness should be paid for large efficiency.

III. Practice: Implementation of UO-CSMA

This section describes the implementation of UO-CSMA on the 802.11-based conventional hardware platform and presents preliminary results. This implementation provides a proof-of-concept of the theory-driven algorithm, UO-CSMA, and addresses the key challenges to transfer from theory to practice. In Section IV, we elaborate the gaps between theory and practice and the workaround solutions to bridge them.

A. WiMesh Network and Common Code Architecture

We implemented UO-CSMA on a campus-scale mesh network testbed at KAIST, Korea, *WiMesh testbed* [34]. The WiMesh has 56 mesh routers in an office building and in 6 undergraduate dormitory buildings over 1 km^2 area, as shown in Figure 1. Each node is typically equipped with two 802.11a/b/g based wireless interface cards as well as one Ethernet interface. The WiMesh has been designed for a research testbed that is open to researchers who want to test and realize their conceptual ideas on top of real hardwares.

One of the unique features in WiMesh is *Common Code Architecture (CCA)*, a programming environment designed to reduce cost and effort of testing a protocol by providing a way of using one code for both simulation and real experiment. A typical way to validate new algorithms or protocols is to first make simulation programs at e.g., [35], [36] and then implement them in real hardwares. However, implementing in the real testbed is significantly different from simulations and much more challenging, as it requires understanding of complicated real network protocol stacks, skills of system programming, and even modification of proprietary hardware through work-around hacking. Deploying protocols designed from a substantially new angle such as UO-CSMA is particularly hard compared to simple modifications of existing protocols, e.g., 802.11 DCF with a new backoff scheme. CCA reduces the transition time from theory to simulation and then to implementation.

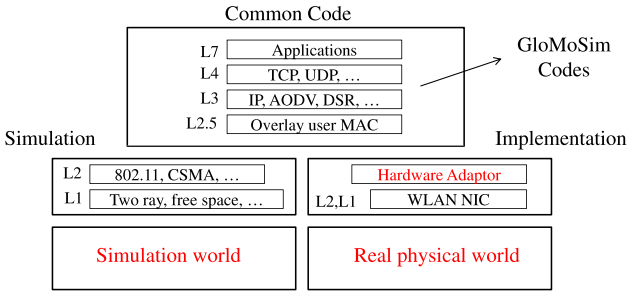


Fig. 2. Common Code Architecture: GloMoSim simulation codes are reusable in WiMesh without modification.

In CCA, we first make a simulation code, where CCA at WiMesh uses the protocol stack of GloMoSim simulator [36]. We can make various simulation scenarios to test a developed protocol. Simulation results give useful feedback to the original protocol, so that we can have chances to modify the original theory and protocol easily. Once simulation tests verify the developed protocol, we load and run the same simulation codes on top of CCA. Most of functions of CCA are run at user-level space which is possible by making interfaces connecting between user-level processes and wireless hardwares. Figure 2 depicts the structure of CCA. We refer the readers to [37] for more details.

B. Setup of Simulation and Implementation

To evaluate performance, we compare UO-CSMA with the optimal benchmark and the 802.11 DCF in both simulation and implementation. In simulation, we implement UO-CSMA by changing the CSMA in GloMoSim, where we mainly modified the part that sets backoff counters. We used two-ray path-loss model, SNR bounded packet reception. Note that we disabled ACK operation. When collision occurs, it lasts for the corresponding holding time. The network is slotted with 1.6ms timeslot and 5Mbps link capacity, and the packet size is set to be 1000 bytes. Using the feature of CCA, we can use the same GloMoSim code to experiment UO-CSMA in real hardware. The setup and hardware specification is shown in Table I.

In 802.11, contention resolution scheme operates based on the contention window, CW, where a back-off counter randomly chosen in the range $[0, CW-1]$. We modify the mechanism of setting CW appropriately so that 802.11 DCF can be turned into a basis for implementing UO-CSMA.

TABLE I
ENVIRONMENTAL SETUP FOR EXPERIMENT

WLAN device	Atheros 5212 chipset
PHY	802.11a, 5.745GHz band, 6Mbps rate
Flow	Single-hop session
Traffic	Saturated
Utility function	$U(x) = \log(x)$
Performance metrics	Total utility (or, throughput), short-term fairness
Holding time	$20, 100, 500$
V parameter	$20, 100, 500$
Weight function	$W(x) = x$ and $\log\log(x)$

(a) *Per-link* CW. In 802.11, CW is maintained at each node, not each link, i.e., one contention window per one interface card. In UO-CSMA, backoff counters should be installed per link. We implemented per-link CW, denoted by CW_l, at GloMoSim, and associate per-link CW_l to the per-node CW, whenever link l is activated.

(b) *From access probability to contention window.* Our theory is developed based on access probability p_l. Thus, we need a way of converting p_l to CW_l, for which we use: $CW_l = 2/p_l$, where '2' is needed since the actual contention window size is selected randomly from $[0, CW_l - 1]$.

(c) CW_{min} *and* CW_{max}. In 802.11, there exist two backoff related values. In short, the real CW is first set to be CW_{min} and then doubles whenever there is a collision. The doubled CW value is used when a collided packet is retransmitted. We disable this feature by setting $CW_{min} = CW_{max}$, so that retransmitted packets are not specially treated.

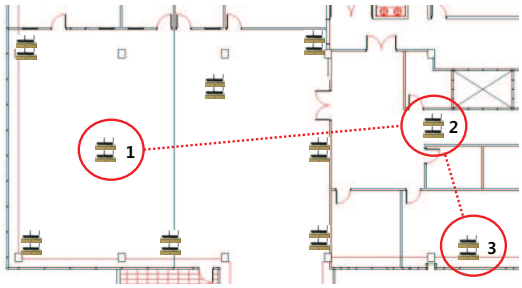
Note that in our experiments, both UO-CSMA and 802.11 DCF are implemented on top of CCA, implying there is no gap due to Common Code between them. To facilitate packet-by-packet parameter control with less overhead, instead of using (indirect) interfaces such as user-level commands, e.g., `iwpriv`, provided by the device driver, we directly instill the target parameter into the so-called TX descriptor used by the firmware to decide action.

C. Results of 3-link Experiment

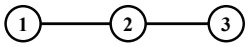
Now, we present the preliminary results in a simple 3 link topology, as shown in Figure 3, where flows 1 and 2 (2 and 3, respectively) are interfering with each other, but flows 1 and 3 are interference-free. In physical reality, interference among links cannot be modeled by a simple graph, since channels are time-varying and interference is very often asymmetric. We tried various placements of wireless nodes in WiMesh network, so that we obtain the desired inteference relationship in Figure 3. This small experiment allows us to get a deep view on many aspects of theory prediction, including performance, transient behavior, and parameter setting. Ongoing work will extend these to large topologies.

Total utility and throughput deviation

We first performed multiple per-link throughput measurement without any interference to figure out effective link capacity without MAC overhead. In our hardware setup,



(a) A network topology with three links and six nodes in experiment on WiMesh testbed



(b) Interference graph: We map a link to a node, where two nodes are if they are interfering with each other.

Fig. 3. Network topology in our experiment

the average per-link capacity amounts to about 5Mbps. We henceforth use this value as a link capacity.

In theory, it is easy to compute the optimal throughput by solving (1) with log utility function, where $\gamma_2^* = 1/3 \times 5 \approx 1.67$Mbps, and $\gamma_1^* = \gamma_3^* = 2/3 \times 5 \approx 3.33$Mbps. In simulation, we implemented UO-CSMA with adaptive backoff scheme on top of CSMA. The simulation almost match the protocol over the slotted model in Section II, except that access probability is replaced by contention windows.

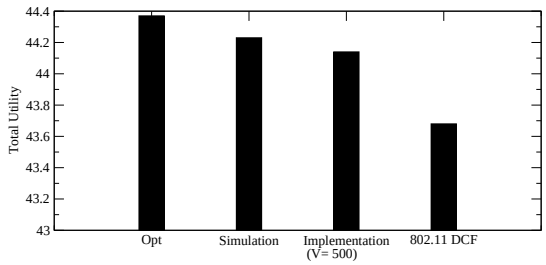
Figure 4 summarizes the results, where we show the total utilities over links as well as deviation from the optimal values. The deviation is computed by normalizing the total throughput difference from optimal one for each case with the optimal solution.

Figure 4 shows that UO-CSMA in simulation has a good match with that in theory, where a small gap is mainly due to the difference between the ideal continuous without collision and the slotted model with collision. We can also observe that UO-CSMA implementation works well with 6.6% throughput deviation from theory, whereas about 14.9% throughput deviation is observed in 802.11 DCF.

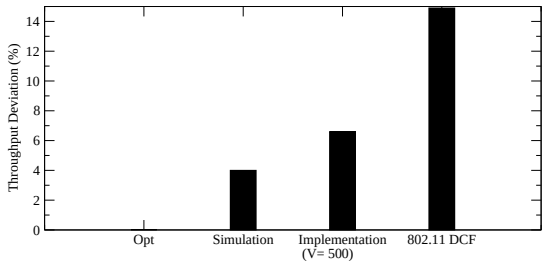
One interesting fact from our measurement in the real testbed is that throughput deviations in simulation and practice differ. The throughput loss in simulation is mainly due to collisions. Note that we use a graph-based interference model with perfectly synchronized nodes in simulation. In the testbed experiment, we indirectly measured the number of collisions by assuming that all data loss is due to collisions, which seems to be suitable since two nodes over a link are very close. The number of collisions in practice just amounts to about 20 packets out of 30000 packets, and thus almost zero collisions. It implies that throughput deviation in practice is due to other reasons such as imperfect execution of holding time etc. The small number of collisions is likely due to the collision avoidance feature of 802.11 and device asynchronization.

Holding time: Efficiency and short-term fairness

Section II shows that increasing holding times lead to

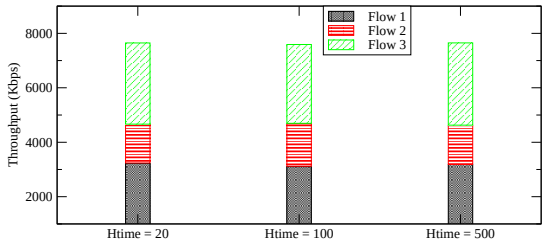


(a) Total utility for theory, simulation and experiment of UO-CSMA, and 802.11 DCF

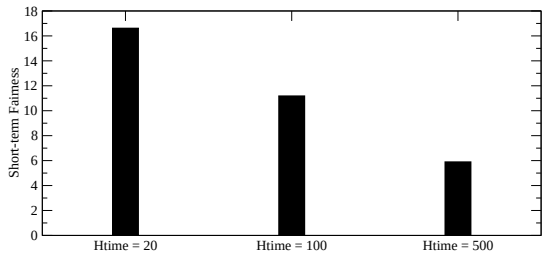


(b) Throughput deviations of simulation and practice from theory

Fig. 4. Total utility and throughput deviation



(a) Throughput for different holding times

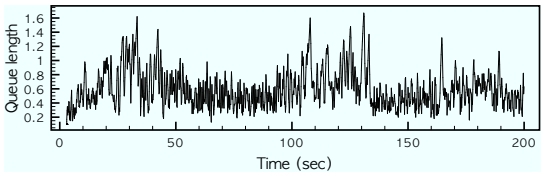


(b) Short-term fairness for different holding times

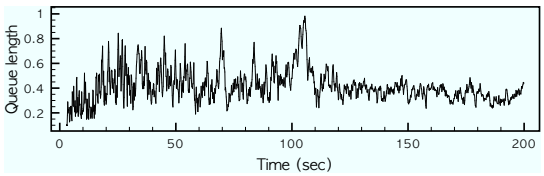
Fig. 5. Throughput and short-term fairness

increase in total throughput, since collisions can decrease at the cost of short-term fairness. We define short-term fairness as the inverse of longest starvation time, where starvation time is the interval between two successful transmission. We measured such tradeoff between throughput and short-term fairness and show it in Figure 5.

From Figure 5, we observe that as holding time increases, the short-term fairness decreases, whereas throughput remains to be the same. To figure out why, we performed the following measurements: By modifying the Madwifi device driver [38], we disabled retransmission. Again, we assume that there is no packet loss due to weak wireless signal, since each sender-receiver pair is very close. Similar to the earlier experiment in



(a) Queue trace of flow 2 with fixed step size



(b) Queue trace of flow 2 with diminishing step size

Fig. 6. Flow 2 converges to 0.4 in both cases.

total utility, for different holding times, we still get a similar number of collisions, i.e., around 0.05%, i.e., no dependency of holding time on the number of collisions. We conjecture that (i) there exists a nice arbitration by backoff in 802.11 DCF in spite of small CW values (for small holding times), and (ii) asynchronous operation of 802.11 induced by delay such as turnover time between transmission and reception and transition time for carrier sensing. More measurements are necessary to verify our conjectures. Therefore, we can conclude that in contrast to theory, there is no benefit of using large holding time when we implement UO-CSMA on top of the 802.11 hardware.

Step size: Transient behavior

We observe the impact on the queue behavior and convergence. of different step size policy. As shown in Figures 6, the trajectories with the step size $b = 0.01$ are oscillating within some neighborhood of the converged point, here 0.4 for flow 2 and 0.2 for flow 3. The trajectories with decreasing step size behaves in a way that it is reduced by 0.9 every 10 seconds starting from 0.1 converges within a few hundred seconds. We recall that this convergence is just related to queue length, while the convergence in terms of throughput is already achieved in both cases through the similar converged points, i.e., 0.4 for flow 2 and 0.2 for flow 3. (We omit the queue traces for flow 3 due to space limitation.)

Parameter V: Efficiency and average delay

We now measure changes in efficiency and the length of virtual queues as V changes. Intuitively, the parameter V controls sensitivity of response to network congestion that is reflected in the virtual queue lengths $\boldsymbol{q}[t]$. Thus, larger V results in higher throughput, yet larger virtual queue lengths (thus larger delay). Figures 7 and 8 show throughput changes and virtual queue behaviors for $V = 20, 100, and 500$, respectively. Figure 8 shows that for the small $V = 20$, we observe that the virtual queue length constantly stays at the pre-specified minimum value (i.e., 0.1). Conversely, for the large $V = 500$, the virtual queue length reach the pre-specified maximum (i.e., 2.3), and stays there from 22 secs on. Note that the virtual queue is nothing but a real queue length multiplied by step size when $\log$ utility is used, i.e., $q[t] = rq[t] \times b[t]$,

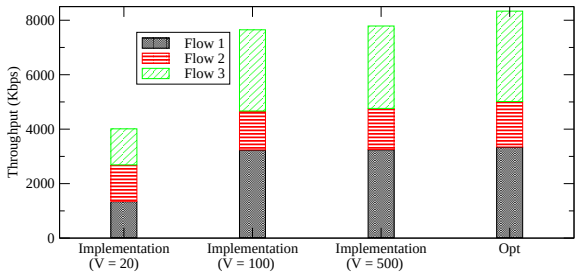
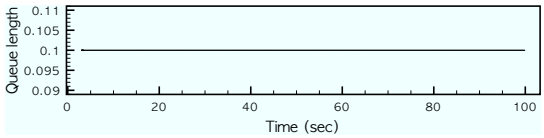


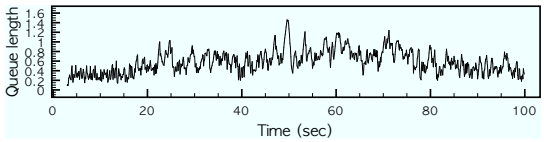
Fig. 7. Throughput for V=20, 100, 500 and optimality

where $rq[t]$ means the actual queue length. Typical lengths of our actual queue range from 100 to 300 pkts.

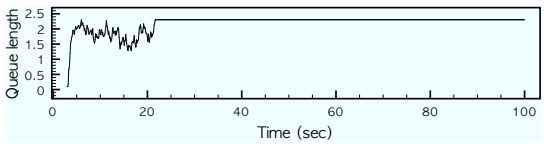
As a remark, our system is saturated and consists of single-hop sessions, where the virtual queue is used as a control variable to derive the system towards different directions. However, in implementation of UO-CSMA, we install per-link intermediate buffers to whom packets are injected (from an infinite backlogged reservoir) at a congestion controlled rate. Thus, the average lengths of such intermediate buffers imply the average queueing delay that a packet experiences. The average delay becomes much more clear when we extend our system to that with multi-hop sessions.



(a) Queue trace of flow 2 for V=20



(b) Queue trace of flow 2 for V=100



(c) Queue trace of flow 2 for V=500

Fig. 8. Backlogs for different V parameters

Weight function: Queue length and convergence time

We also investigate the transient behaviors of queue length and throughput for different weight functions. We have tested two functions: $W(x) = x$ and $W(x) = \log\log(x + e)$. As depicted in Figures 9 and 10, we can observe that $W(x) = x$ has smaller virtual queue length and longer convergence time than $W(x) = \log\log(x + e)$ with the same throughput being achieved for both functions. The virtual queue lengths for $W(x) = x$ oscillate between 0.2 and 0.4 for $W(x) = x$, whereas between 0.5 and 1.5 for $W(x) = \log\log(x + e)$. This is due to the fact that the long-term throughput is determined by ρ (thus ρ should be the same for both functions), but since $\rho = \exp(W(q))$, we have smaller virtual queue

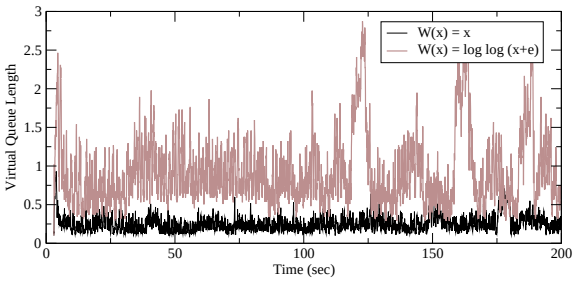


Fig. 9. Virtual queue traces for $W(x) = x$ and $W(x) = \log\log(x + e)$

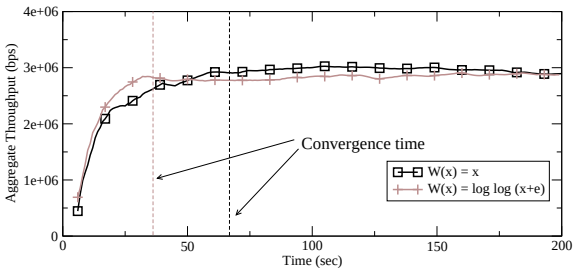


Fig. 10. Aggregate throughput traces and convergence times for $W(x) = x$ and $W(x) = \log\log(x + e)$

lengths for $W(x) = x$. Theoretical verification for faster convergence time for "less aggressive" weight function, i.e., $W(x) = \log\log(x+e)$ remains for future study. We conjecture that less aggressive weight functions result in larger virtual queue length fluctuations, which, in turn, let the protocol respond to congestion more sensitively.

IV. Theory-Practice Gaps

A. Overview of gaps

This section briefly discusses the origins of many gaps between theory and practice based on our experimental measurements. Such an understanding is important to improve theory-driven implementation of UO-CSMA. In addition, it motivates us to develop better theories capturing and bridging this gap. The key origins of the gaps include imperfect modeling of sensing, holding time, and interference (e.g., capture effect) in practice.

Other gap origins are caused by implementation methodologies. First, we adopt the Common Code Architecture to facilitate simulation and experiment, which, however, may lead to unexpected overhead as well as may have impact on the behaviors of networking, e.g., inappropriate event scheduling in CCA can starve transmission opportunity in the wireless interface. Second, we also implement a new MAC protocol on top of the conventional MAC hardware that are just partially controlled by and also even hidden to us (e.g., functions implemented at firmware). For example, when a contention window CW is set by us, the system randomly chooses a backoff counter in the interval $[0, CW - 1]$. We are not able to know the real backoff counter, which is sometimes needed to investigate the MAC behavior more rigorously.

B. Gap between Theory and Simulation

In simulation, we implemented UO-CSMA under perfect synchronization over a discrete slotted system by modifying the conventional CSMA with new CW-based backoff counter control mechanism. We also used a graph-based interference model based on which sensing mechanism is implemented. Carrier sensing is deterministic, and its range is set to be equal to transmission range. Thus, as seen in Figure 11, we observe a very small gap between theory and simulation, which is just due to collisions in slotted-time model.

TABLE II
THEORY-SIMULATION GAP

Gap	Theory	Simulation
Backoff	Data-slot based	Mini-slot based
Collision	No	Yes, and last for holding time

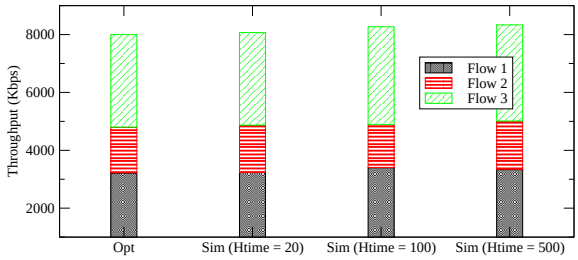


Fig. 11. Throughput between theory and simulation for holding time = 20, 100, and 500

C. Gap between Simulation and Implementation

(a) *Hidden-terminal: Link-oriented vs. node-oriented.* Theory does not cover the hidden-terminal problem that even if two nodes do not sense signal, their intended transmissions can collide. This is due to the gap between theory that interferences are characterized by *links* and practice that carrier-sensing is performed by *nodes*. The hidden-terminal problem can be a cause of throughput decrease in practice, when RTS/CTS mechanism can be a candidate solution. Note that RTS/CTS-like signaling may also help with driving the system with slotted model from theory into that with short collisions (see Section II-F).

(b) *Difference in sensing and decoding ranges.* For example, in 802.11b, systems are designed to be conservative, so that sensing range is larger than decoding range. Thus, transmissions may defer even if they can be successful in decoding. Interestingly, this difference in decoding and sensing ranges is heterogeneous depending on wireless systems. for example, sensing mechanism of 802.11a in our testbed differs from 802.11b, in that nodes first try to decode a preamble of packets from neighbors (thus sensing range is equal to decoding range). When the preamble is not decodable mainly due to some unpredictable behavior such as instantaneous hardware-malfunction, sensing range that is smaller than decoding range is applied for carrier-sensing, whereas in 802.11b sensing range is larger than decoding range.

TABLE III
SIMULATION-IMPLEMENTATION GAP

Gap	Simulation	Implementation
Hidden-terminal	No, due to link-based interference model	Yes, due to carrier sensing by senders
CS and TX ranges	Always same	Different and governed by PHY layer
Time-varying channel	No, fixed channel	Yes, no guarantee of holding time
Asymmetry	No, symmetric	Yes, both at transmission and carrier sensing

(c) *Time-varying channel.* Time-varying channels precludes us from ensuring holding times. The holding time set by a node i cannot always be guaranteed by an interfering neighbor of i that decrements its backoff counter whenever signal is not sensed (due to time-varying channel), and preempts the transmission of node i. Note that ensuring holding time correctly is the key to efficiency and fairness. Time-varying channels also generate packet loss induced by channel degradation that are not considered by theory.

(d) *Asymmetry.* Wireless links are often asymmetric because signals propagate differently between two nodes. Furthermore, link asymmetry is also time-varying. Theory adopts a model that both link channel conditions and interferences are symmetric and fixed.

D. Gap between Clean-slate and 802.11

There has been exciting discussion on the vision of clean-slate design for the overall future of the Internet. As an alternative option for a *local surgery* such as UO-CSMA, implementation over legacy hardware like conventional 802.11 also presents a least resistance path from theoretical advance to practical impact. However, there exist challenges and gaps generated by legacy hardwares, summarized below, using three examples.

(1) Holding time

The first example of this gap is holding time that relies on perfect carrier-sensing capability. Perfect carrier-sensing may be achieved by clean-slate design. However, in 802.11, we found out that it is very hard to do so due to (i) hidden state information (many parts are implemented in its firmware), (ii) imperfect synchronization, and (iii) asynchronous features.

We adopted several work-arounds to ensure that holding time is executed correctly.

- *Imperfect* $CW = 0$. We use AIFS (Arbitration Inter-Frame Space) from 802.11. The AIFS specifies an interval between packet transmissions. We set AIFS to be a large value only when a node first access the media right after its backoff counter reaches 0, but to be a small value for back-to-back transmissions in the middle of holding time. This heterogeneous setting of AIFS precludes a node from intercepting transmissions from other nodes being in the middle of holding time.

TABLE IV
CLEAN SLATE-OVER 802.11 GAP

Gap	Clean slate	Over 802.11
Holding	Perfect	Not perfect
Contention control	By access prob.	By discrete back-off, but only 2^n-1 CW value available
Transmission type	User defined	Unicast with ACK
Synchronization	Synchronization from PHY	Asynchronous
Overhead	Hardware dependent	802.11 chipset dependent

- *Time-varying channel.* In time-varying channels, even while a node i is transmitting data, node i's interfering links sometimes decrement their backoff counters in case when interfering links may not sense node i's signal due to time-varying channels. To prevent it, we use the NAV (Network Allocation Vector) option recording the amount of time during which neighbors should be silent irrespective of carrier-sensing. The NAV option helps much when the packet with a NAV value is overheard and decoded by interfering neighbors.
- *More workaround solutions.* The workaround solutions mentioned earlier just partially solve the gap origin that we listed in the previous section. Obviously, we try to find more workaround solutions, for which we need to perform more rigorous measurement. We expect that new software and hardware help with it.

(2) CW granularity

The second example is the coarse CW granularity that prevents us from controlling access intensity perfectly. The 802.11 allows only $2^x - 1$ CW values, which is again implemented in firmware in the chip used in our experiment. In our implementation, we take a "ceiling function" of the computed CW value, say cw, i.e., the value that is larger than cw and closest to $2^x - 1$ for some positive integer x.

(3) 802.11-specific packets

The third example is specially treated packets in 802.11, where examples include beacon packets (with high priority) that are used to identify neighbors. These beacon packets may intercept chances for usual data transmission, having negative impact on guaranteeing holding times. To deal with it, we modified the device driver to minimize the impact of the packets for beacon signal by increasing beacon interval from 100 msec to 5 sec (Our measurement tell us that too large beacon intervals generate network malfunctions, e.g., slow connectivity update).

In addition to gaps due to the use of legacy hardware, there are additional challenges in making the overlay approach work, generating additional gaps. Mentioned earlier, we adopt the Common Code Architecture for compatibility of simulation with real experiment, where MAC is partially implemented in overlay. It it natural that Overlay MAC cannot fully utilize the hardware-level information such as channel status.

V. DEVELOPING IMPLEMENTATION-INSPIRED THEORY

Implementing theory-driven algorithms provides useful feedbacks, motivating us to look into and augment the existing theories with new theories. The new theory is required to revisit and even develop new theory by considering (i) what existing theory assumed away, e.g., overhead, asymmetry, control granularity, (ii) what existing theory modeled simplistically, e.g., imperfect holding and sensing, and finally (iii) what theory analyzed loosely, e.g., convergence speed, transient behavior like queue buildup, and parameter choice.

VI. NEXT STEPS IN IMPLEMENTATION

There are interesting next steps to take in the future, listed as follows:

- *Large-scale networks with multi-hop sessions.* Test over a simple topology with simple traffic model is insufficient to fully verify theory-driven algorithms and bridge the gaps. We plan to test UO-CSMA in large-scale networks consisting of more than 20 nodes with multi-hop sessions, where we need routing protocols as well as more practical transport protocols, e.g., TCP.
- *Software and hardware upgrade.* We plan to upgrade our testbed with new 802.11n cards, as well as newer Linux kernels, with which we believe that much more freedom to access the hardware is provided.

VII. CONCLUDING REMARKS

Theories depend on a mathematical crystallization of the engineering system under study. This process needs to ignore some parts of the physical characteristics of the system by making assumptions, simplify other parts by building tractable models, and focus on metrics that can lead to crisp quantification and tight analysis using the existing mathematical machinery. There perhaps is no other starting point towards a rigorous study. This paper aims at going one step further for the topic of distributed scheduling in wireless networks, through an implementation over conventional 802.11 hardware and a deployment in the WiMesh network at KAIST. This is simply an "interim report", where we report the first, small-scale experiment that confirms the ability of UO CSMA to get close to utility optimality despite many gaps between theory and practice. The discovery, quantification, and bridging of these gaps are more important than the numerical results. We identify the key gaps, group them in three types, and explain where they originated and how they might be bridged, either by work-around engineering solutions or by addressing the new modeling challenges through enriched theories of wireless scheduling.

ACKNOWLEDGMENTS

We thank helpful discussions with L. Jiang, J. K. Lee, J. Liu, H. V. Poor, R. Srikant, D. Shah, and J. Walrand. Part of this work has been supported in part by IT R&D program of MKE/IITA [2009-F-045-01], Korea Research Council of Fundamental Science and Technology, and the Princeton EDGE Lab that is in part sponsored by the US NSF Computing

Research Infrastructure program, the ONR Defense University Research Instrumentation Program, and Qualcomm.

REFERENCES

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

[2] L. Tassiulas, "Linear complexity algorithms for maximum throughput in radionetworks and input queued switches," in *Proceedings of IEEE Infocom*, San Francisco, CA, 1998.

[3] P. Chaporkar, K. Kar, and S. Sarkar, "Throughput guarantees through maximal scheduling in wireless networks," in *Proceedings of the 43rd Annual Allerton Conference on Communication, Control and Computing*, Monticello, IL, 2005.

[4] E. Modiano, D. Shah, and G. Zussman, "Maximizing throughput in wireless networks via gossiping," in *Proceedings of ACM Sigmetrics*, Saint Malo, France, 2006.

[5] A. Eryilmaz, A. Ozdaglar, and E. Modiano, "Polynomial complexity algorithms for full utilization of multi-hop wireless networks," in *Proceedings of Infocom*, Anchorage, AK, 2007.

[6] S. Sanghavi, L. Bui, and R. Srikant, "Distributed link scheduling with constant overhead," in *Proceedings of ACM Sigmetrics*, San Diego, CA, 2007.

[7] S. Ray and S. Sarkar, "Arbitrary throughput versus complexity tradeoffs in wireless networks using graph partitioning," in *Proceedings of Information Theory and Applications Second Workshop*, La Jolla, CA, 2007.

[8] C. Joo and N. B. Shroff, "Performance of random access scheduling schemes in multi-hop wireless networks," in *Proceedings of Infocom*, Anchorage, AK, 2007.

[9] Y. Yi and M. Chiang, "Wireless scheduling with O(1) complexity for m-hop interference model," in *Proceedings of IEEE International Conference on Communications*, Beijing, China, 2008.

[10] A. Gupta, X. Lin, and R. Srikant, "Low-complexity distributed scheduling algorithms for wireless networks," in *Proceedings of IEEE Infocom*, Anchorage, AK, 2007.

[11] X. Lin and S. Rasool, "Constant-time distributed scheduling policies for ad hoc wireless networks," in *Proceedings of IEEE Conference on Decision and Control*, San Diego, CA, 2006.

[12] K. Kar, S. Sarkar, and L. Tassiulas, "Achieving proportional fairness using local information in aloha networks," *IEEE Transactions on Automatic Control*, vol. 49, no. 10, pp. 1858–1862, 2004.

[13] J. W. Lee, M. Chiang, and R. A. Calderbank, "Utility-optimal medium access control: reverse and forward engineering," in *Proceedings of IEEE Infocom*, Barcelona, Spain, 2006.

[14] X. Wang and K. Kar, "Cross-layer rate optimization for proportional fairness in multihop wireless networks with random access," in *Proceedings of ACM Mobihoc*, Urbana-Champaign, IL, 2005.

[15] A. H. Mohsenian-Rad, J. Huang, M. Chiang, and V. W. S. Wong, "Utility-optimal random access: Optimal performance without frequent explicit message passing," *IEEE Transactions on Wireless Communications*, vol. 8, no. 2, pp. 898–911, 2009.

[16] ——, "Utility-optimal random access: Reduced complexity, fast convergence, and robust performance," *IEEE Transactions on Wireless Communications*, vol. 8, no. 2, pp. 898–911, 2009.

[17] J. Liu, A. Stolyar, M. Chiang, and H. V. Poor, "Queue backpressure random access in multihop wireless networks: Optimality and stability," *IEEE Transactions on Information Theory*, vol. 55, no. 9, 2008.

[18] X. Lin and N. B. Shroff, "The impact of imperfect scheduling on cross-layer rate control in wireless networks," in *Proceedings of IEEE Infocom*, Miami, FL, 2005.

[19] M. J. Neely, E. Modiano, and C. Li, "Fairness and optimal stochastic control for heterogeneous networks," in *Proceedings of IEEE Infocom*, Miami, FL, 2005.

[20] L. Chen, S. H. Low, M. Chiang, and J. C. Doyle, "Joint optimal congestion control, routing, and scheduling in wireless ad hoc networks," in *Proceeding of IEEE Infocom*, Barcelona, Spain, 2006.

[21] A. Eryilmaz and R. Srikant, "Joint congestion control, routing, and MAC for stability and fairness in wireless networks," *IEEE Journal on Selected Areas of Communication (JSAC), Special Issue on Nonlinear Optimization of Communication Systems*, vol. 24, no. 8, pp. 1514–1524, 2006.

[22] A. L. Stolyar, "Maximizing queueing network utility subject to statbility: greedy primal-dual algorithm," *Queueing Systems Theory and Applications*, vol. 50, no. 4, pp. 401–457, 2005.

[23] Y. Yi and M. Chiang, "Stochastic network utility maximization and wireless scheduling," 2009, to be published as a book chapter of *Next-Generation Internet Architectures and Protocols*, Cambridge University Press. Also, available at http://lanada.kaist.ac.kr/pubs/scheduling.pdf.

[24] Y. Yi, A. Proutiere, and M. Chiang, "Complexity in wireless scheduling: Impact and tradeoffs," in *Proceedings of ACM Mobihoc*, Hong Kong, China, 2008.

[25] Y. Yi, J. Zhang, and M. Chiang, "Delay and effective throughput of wireless scheduling in heavy traffic regimes: Vacation model for complexity," in *Proceedings of ACM Mobihoc*, New Orleans, LA, 2009.

[26] L. Jiang and J. Walrand, "A CSMA distributed algorithm for throughput and utility maximization in wireless networks," in *Proceedings of the 46th Annual Conference on Communication, Control and Computing*, Monticello, IL, 2008.

[27] S. Rajagopalan and D. Shah, "Distributed algorithm and reversible network," in *Proceedings of the Cconference on Information Science and Systems*, Princeton, NJ, 2008.

[28] J. Liu, Y. Yi, A. Proutiere, M. Chiang, and H. V. Poor, "Adaptive CSMA: Approaching optimality without message passing," *Wiley Journal of Wireless Communications and Mobile Computing, Special Issue on Recent Advances in Wireless Communications and Networking*, Dec. 2009, Preliminary version in Microsoft Research Technical Report, TR2008-128.

[29] F. Kelly, "Stochastic models of computer communication systems," *Journal of the Royal Statistical Society*, vol. 47, no. 3, pp. 379–395, 1985.

[30] M. Durvy and P. Thiran, "Packing approach to compare slotted and non-slotted medium access control," in *Proceedings of IEEE Infocom*, Barcelona, Spain, 2006.

[31] C. Bordenave, D. McDonald, and A. Proutiere, "Performance of random medium access control: An asymptotic approach," in *Proceedings of ACM Sigmetrics*, Annapolis, MD, 2008.

[32] F. Kelly, *Reversibility and Stochastic Networks*. Wiley, Chichester, 1979.

[33] V. Borkar, "Stochastic approximation with controlled markov noise," *Systems and control letters*, vol. 55, pp. 139–145, 2006.

[34] S. Shrestha, J. Lee, A. Lee, K. Lee, J. Lee, and S. Chong, "An open wireless mesh testbed architecture with data collection and software distribution platform," in *In Proceedings of TridentCom*, Orlando, FL, 2007.

[35] "Ns-2," http://www.isi.edu/nsnam/ns/.

[36] "Glomosim," http://pcl.cs.ucla.edu/projects/glomosim/.

[37] J. Lee, J. Lee, S. Shrestha, and S. Chong, "Common code architecture for future internet researches of wireless mesh networks," in *Proceedings of CFI*, Seoul, Korea, 2008.

[38] "Madwifi project." [Online]. Available: http://madwifi-project.org/