DSS: Distributed SINR based Scheduling Algorithm for Multi-hop Wireless Networks

Jiho Ryu, *Student Member, IEEE*, Changhee Joo, *Member, IEEE*, Ted "Taekyoung" Kwon, *Member, IEEE*, Ness B. Shroff, *Fellow, IEEE*, and Yanghee Choi, *Senior Member, IEEE*

Abstract—The problem of developing distributed scheduling algorithms for high throughput in multi-hop wireless networks has been extensively studied in recent years. The design of a distributed low-complexity scheduling algorithm becomes even more challenging when taking into account a physical interference model, which requires the SINR at a receiver to be checked when making scheduling decisions. To do so, we need to check whether a transmission failure is caused by interference due to simultaneous transmissions from distant nodes. In this paper, we propose a scheduling algorithm under a physical interference model, which is amenable to distributed implementation with 802.11 CSMA technologies. The proposed scheduling algorithm is shown to achieve throughput optimality. We present two variations of the algorithm to enhance the delay performance and to reduce the control overhead, respectively, while retaining throughput optimality.

Index Terms—Wireless scheduling, SINR, CSMA, Discrete Time Markov Chain

1 Introduction

It is widely acknowledged that link scheduling is a major performance bottleneck in wireless multi-hop networks. Link scheduling determines which transmitter-receiver pairs (links) are to be simultaneous activated at a given moment in order to achieve high throughput, low delay, fairness, etc. Over the past couple of decades, a variety of link scheduling algorithms under different interference models have been studied in order to achieve high performance and low complexity. In wireless multi-hop communication environments, simultaneous link activation could cause siginificant mutual interference that results in transmission failure if, e.g., the interference exceeds a certain threshold. Also, for a practical implementation, it is desirable to design distributed algorithms for link scheduling. Therefore, developing high-performance distributed scheduling solutions that are amenable to pratical implementation is one of the most significant challenges in multi-hop wireless networks. So far, many scheduling schemes, centralized or distributed, have been studied in the literature under different interference models with different time granularities (continuous or time-slotted). They often seek to optimize performance objectives such as achievable throughput, or explore tradeoffs among complexity, message-passing overhead, and throughput.

The problem of achieving throughput optimality in wireless multi-hop networks has been extensively studied. The well-known Max-Weight Scheduling (MWS) [2] algorithm has been shown to achieve throughput optimality at the cost of the very high time-complexity. In a time slotted system, the MWS algorithm picks, at each time slot, the set of non-conflicting (e.g., not causing transmission failures) links whose queue-weighted sum is the largest. In general, finding such a set of max-weighted non-conflicting links is NP-hard and requires centralized implementation. Many sub-optimal (and hence more practical) solutions have been proposed over the past several years aimed to reduce this algorithmic complexity. For example, Greedy Maximal Scheduling (GMS) [3], [4], [5], [6] is a well known sub-optimal solution that approximate MWS. It picks links in decreasing order of their queue lengths without violating their underlying interference constraints. GMS can be implemented in a distributed fashion [7] but only at the expense of increased complexity due to the requirement that links be globally ordered. In [8], the authors have addressed the difficulty in global ordering and developed Local Greedy Scheduling (LGS), which suggests that local ordering may be sufficient to achieve high performance in practice. Empirical results show that LGS provides good throughput and delay performance at the lower complexity. However, although LGS requires only local message exchanges of queue length information among neighboring links, it may suffer from high message-passing overhead, if the number of local neighbors is large.

Recently, a class of scheduling algorithms that exploit Carrier Sensing Multiple Access/Collision Avoidance

- *J. Ryu, T. Kwon, and Y. Choi are with the School of Computer Science and Engineering, Seoul National University, Seoul 151-742, Korea. E-mail: {jihoryu, tkkwon, yhchoi}@snu.ac.kr.*
- *C. Joo is with the School of Electrical and Computer Engineering, UNIST, Ulsan 689-798, Korea. E-mail: cjoo@unist.ac.kr.*
- *N. B. Shroff is with the Departments of ECE and CSE, The Ohio State University, Columbus, OH 43210, USA. E-mail: shroff@ece.osu.edu.*
- *A preliminary version of this work was presented at ACM MSWiM, 2010 [1].*

(CSMA/CA) have been developed and shown to achieve throughput optimality without global information [9], [10], [11], [12]. In particular, a discrete time system, named Q-CSMA, has been developed in [11] modelling a multi-hop network as a Discrete Time Markov Chain (DTMC) with a product form stationary distribution. Q-CSMA allows each link to choose itself with a certain probability that depends on its own queue length. As the selection procedure continues, it has been shown to yield a stationary distribution of schedules with optimal throughput performance. Although Q-CSMA is throughput-optimal, it often performs poorly in terms of delay, especially under heavy traffic load [13]. **Qiao et al. recently developed a new CSMA based scheduling scheme that improves delay performance [14]. They employ simplex algorithm to solve linear programming (LP) problems, and promote quick transitions between optimal schedules to achieve better delay performance.**

We note that the above scheduling schemes have been developed assuming "theoretical" graph-based interference models, under which only a binary interference relation exists between each pair of links. This, however, cannot capture accumulative nature of wireless interference from multiple transmitters. In practice, Signal-to-Interference-plus-Noise-Ratio (SINR) among all activated links should be taken into account. **Although several distributed link scheduling algorithms under the so-called SINR-based model have been proposed [15], [16], [17], they attempt to maximize the number of (simultaneously) activated links and achieves only sub-optimal throughput performance.**

In this paper, we develop a distributed SINR-based scheduling scheme (DSS) that is throughput-optimal in wireless multi-hop networks. Unlike the previous works based on graph-based interference models, we considers more realistic SINR-based interference model. We design a distributed algorithm that operates under the SINR-based model, and show that it achieves a product-form stationary distribution of the system state (i.e., the set of activated links), which implies throughput optimality of the proposed scheduling scheme. To the best knowledge of the authors, this is the first distributed scheduling solution that achieves optimal throughput under the SINR-based interference model. We extend DSS to improve delay performance. By developing a novel dual-state approach, we can separate actual transmission schedule from the system state, and include a larger number of active links in a schedule. Finally, we address control overhead problem for carrier-sensing and contention, and reduce the overhead by adopting p-persistent CSMA contention mechanism. We show that the new solution still retains throughput optimality and achieves better empirical performance when the contention period is short.

Note that unlike the standard IEEE 802.11 DCF, our solutions are a synchronous time-slotted system and adaptively control attempts to channel. However, both our solutions and the IEEE 802.11 DCF have very similar time-slot structure for contention and data transmission. This similarity will facilitate practical implementation of our proposed solutions by modifying the existing IEEE 802.11 DCF implementation as in [18].

The rest of this paper is organized as follows. We first provide the system model in Section 2. We describe our DSS algorithm and show that it achieves optimal throughput in Section 3. We extend it and enhance delay performance using a novel dual-state approach in Section 4. Then we address the control overhead problem and develop an algoirthm to reduce the overhead in Section 5. Numerical evaluation of our solutions in comparison with other CSMA-based scheduling algorithms have been provided in Section 6. We conclude our paper in Section 7.

2 Network model

A wireless network is modeled by a graph $G(V, E)$, where V denotes the set of nodes and E denotes the set of links. We represent a directed link by an ordered pair of nodes. Two nodes are directly connected by a link if they can successfully exchange packets when there is no other transmission. We assume that the transmission power P is fixed, and that a single frequency channel is available for the whole system; that is, two or more links that transmit at the same time may interfere with each other. The transmission of a packet over a link will be successful if the SINR at the receiver is above a certain SINR threshold denoted by θ_{th}. We assume that a single SINR threshold is used for all the links. The transmission rate and its corresponding SINR threshold can be carefully chosen by considering the network density [19].

All links are assumed to have unit capacity (i.e., a packet can be transmitted at a unit time) and we consider a time-slotted system, where each time slot consists of a control slot and a data slot. A transmission failure occurring the SINR at the receiving node of a link is below a certain threshold θ_{th}. The control slot is intended to determine a "feasible" transmission schedule for the data slot. A feasible schedule means that the links in the schedule can be activated without causing transmission failures. The data slot is used for the scheduled links to transmit a data packet. A feasible schedule is defined as a set of links that can be active simultaneously, and each link has a sufficient SINR value at the receiver. We represent a schedule $\vec{x}(t)$ in time slot t by a vector $\{0,1\}^{|E|}$, where $x_i(t) = 1$ if link i is achieved at time slot t and 0 otherwise. Slightly abusing the notation, we also denote the set of active links at slot t by $x(t)$, i.e., $i \in x(t)$ implies that $x_i(t) = 1$.

We assume that data packets arrive at a link and leave the system immediately once it is transmitted over the link. Although we consider only single-hop traffic, scheduling difficulty lies in multi-hop nature of wireless interference. Let λ_i be the arrival rate of data

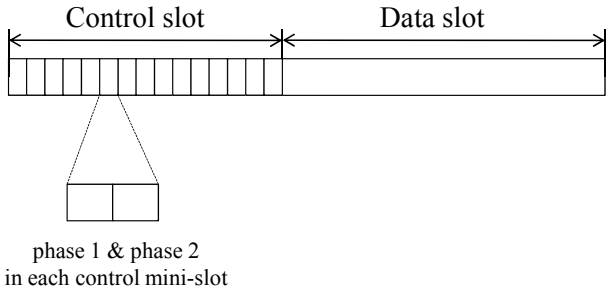


Fig. 1. A time slot is illustrated, which consists of a control slot and a data slot. The control slot is divided into multiple mini-slots, each of which consists of two phases.

packets at link $i \in E$, and let $\lambda = \{\lambda_1, \lambda_2, ..., \lambda_{|E|}\}$ be the corresponding vector. The *capacity region* of a scheduling policy is the set of arrival rates λ, for which there exists a scheduling algorithm that can stabilize the network. The network is said to be stable if the expected queue lengths of all links remain finite. Let $\mathcal{M}$ be the set of all feasible schedules in our network model. The capacity region can be written as

$$\Lambda = \{\lambda \mid \exists\, \mu \in Co(\mathcal{M}),\ \lambda < \mu\}, \qquad (1)$$

where $Co(\mathcal{M})$ denotes the convex hull of $\mathcal{M}$ [2] and the inequality is component-wise. We say that a scheduling algorithm is *throughput optimal*, if it can stabilize the network for any arrival rate in Λ.

3 Distributed SINR-based Scheduling Algorithm (DSS)

In this section, we describe a basic throughput-optimal scheduling scheme that operates in a distributed manner under the SINR interference model specified in the previous section. We describe this scheduling algorithm called DSS and analyze its throughput performance.

3.1 Algorithm Description

At each time slot t, we decide a transmission schedule $x(t)$ by reusing some of the activated links from the previous transmission schedule $x(t-1)$ at slot $t-1$, and adding new links. To add new links, let $m(t)$ denote a *random candidate vector* at time slot t, which refers to a set of randomly selected links. From $m(t)$, we derive a new "feasible" *addition vector* $d(t)$ using the procedure described in the next paragraph. Then for each link $l \in d(t)$, the proposed scheduling algorithm decides probabilistically whether l will be activated or not. To summarize, we determine the final $x(t)$ by combining $x(t-1) \setminus m(t)$ and probabilistically activated links from $d(t)$.

The difficult part of the procedure is to find a feasible addition vector $d(t)$ in a distributed manner. To this end, we assume that a receiver can differentiate (i) the received signal strength (RSS) if the signal is transmitted from its sender, and (ii) the interference power if the signal is transmitted from an interferer. This can be calculated from a pre-measured radio frequency (RF) profile as in [21]. In the following, we describe the proposed scheduling algorithm with a focus on how to find $d(t)$. The pseudo code for DSS is provided in Algorithm 1.

Algorithm 1 Distributed SINR-based Scheduling Algorithm (DSS)

Initialization:
Each sender of link l includes l in $m(t)$ with the attempt probability p_a
$d_l \leftarrow 0$.
IF $l \in x(t-1) \backslash m(t)$ **THEN**
  backoff(l) = 0
ELSE IF $l \in m(t)$ **THEN**
  backoff(l) is chosen at random in $[1,\ M-1]$
ELSE
  backoff(l) = -1
END IF
FOR $c = 0$ to $M-1$ **do**
/* Phase 1 */
  IF backoff(l) $= c$ **THEN**
    Sender of link l broadcasts a control packet
  END IF
  IF Receiver of a control packet **THEN**
    Calculate $SINR$ and X_i
  ELSE
    X_i = measured signal strength
  END IF
/* Phase 2 */
  IF $d_l = 1$ or backoff(l) $= 0$ **THEN**
    IF $SINR(or RSS/X_i) < \theta_{th}$ **THEN**
      Receiver of link l generates a busy-tone.
    END IF
  END IF
  Check busy-tone.
  IF No busy-tone **THEN**
    $d_l \leftarrow 1$
    $I(l, c+1) \leftarrow I(l, c) + X_i$
  ELSE
    $I(l, c+1) \leftarrow I(l, c)$
  END IF
END FOR
/* Data slot: */
IF $d_l = 1$ **THEN**
  With prob. p_l, schedule link l, i.e., $l \in x(t)$, and send a packet during the data slot
ELSE IF backoff(l) $= 0$ **THEN**
  With prob. 1, schedule link l, i.e., $l \in x(t)$, and send a packet during the data slot
END IF

(1) At the beginning of time slot t, each link determines with attempt probability $p_a(t)$ whether the link is included in the random candidate vector $m(t)$, where

$$p_a(t) = \begin{cases} 0 & \text{if link } l \text{ has an empty queue} \\ p_a & \text{if link } l \text{ has a non-empty queue.} \end{cases}$$

(2) As shown in Fig.1, we divide the control slot into M mini-slots, where each mini-slot consists of two phases. The first mini-slot is reserved for links that continue transmissions from $x(t-1)$ to inform their transmissions (at slot t) to the other links. That is, these links, which were activated at time slot $t-1$ and are not included in $m(t)$, will be included in $x(t)$ (i.e., they are scheduled first). To elaborate, we let links $l \in x(t-1) \backslash m(t)$ be included in $x(t)$. Each link (i.e., its sender) in $x(t-1) \backslash m(t)$ transmits a small control packet to its receiver during phase 1 of the first mini-slot. The receivers of the control packets calculate the RSS and the interference power, while the other nodes calculate the interference power only. For the first mini-slot, the second phase is not used.

(3) Let $d(t, m)$ denote the *addition vector* after m-th mini-slot of the control slot in time slot t. After the first mini-slot, we have $d(t, 1) = x(t-1) \backslash m(t)$, and will have the final $d(t) = d(t, M)$ after M mini-slots. Let $RSS(l)$

denote the RSS at the receiver of link l. Also let X_i be the measured interference power at node i, and let $I(l,m)$ denote the accumulated interference power level up to mini-slot m at the receiver of link l. At each m-th mini-slot ($m > 1$), we perform the following procedure. A link (in $m(t)$) whose backoff timer expires transmits a control packet during phase 1. Note that there can be multiple links whose timers expire at a given mini-slot. The receiver i (of the link l) measures the RSS ($RSS(l)$) and the interference power X_i. A node j that is neither a sender nor a receiver will measure X_j only. Now each receiver of the link in $d(t,m)$ checks whether its SINR is violated or not. That is, receiver i of link l for which $RSS(l)/(I(l,m)+X_i) < \theta_{th}$ generates a busy-tone during phase 2 of m-th mini-slot to announce that the links that sent control packets should not be scheduled. If the SINR threshold is satisfied by every node, there is no busy-tone, and the links can be scheduled to $d(t,m)$.

(4) If no busy-tone is detected during phase 2 of m-th mini-slot, each link that transmits a control packet is added to $d(t,m)$ and the accumulated interference power of at node i (of link l) is updated by $I(l,m) = I(l,m-1)+X_i$. If a busy-tone is overheard, then the link that transmitted a control packet is not added by setting $d(t,m) = d(t,m-1)$ and the accumulated interference power is not changed $I(l,m) = I(l,m-1)$. We iterate steps (3) and (4) from 2nd to M-th mini-slots, and obtain the final addition vector by setting $d(t) = d(t,M)$.

We need to address a special case in which a node generates a busy-tone in step (3). If two nearby links attempt to transmit control packets at the same time, and their corresponding receivers cannot each decode the control packet due to their low SINR values, then the receivers cannot figure out that they are the intended receivers and will not generate a busy-tone. This can lead to an infeasible schedule since the senders will be included in the decision vector. To avoid such scenarios, we let a node generate a busy-tone if it receives a high signal strength but cannot decode the packet.

Let us illustrate the above procedure as follows. Suppose that the backoff timer of a particular sender (of a link in $m(t)$), say s_1, expires at mini-slot 2 (in slot t), while the senders of the other links are still decreasing their timers. Then s_1 broadcasts a small control packet which contains the corresponding receiver (say r_1) information during the phase 1 of mini-slot 2**, which is shown in Fig.2(a).** If r_1 successfully receives the control packet, no action is needed in the phase 2. Assume that the other receivers (whose links are already scheduled before $2nd$ mini-slot) still have their SINR values above the threshold despite s_1's transmission (i.e., interference), they will not send a busy-tone as well. Then s_1 concludes that its link is included in $d(t)$ (more precisely, $d(t,2)$). The other nodes that receive the control packet merely add the received interference power into its accumulated interference power. Note that even if a node cannot decode a packet (e.g., its SINR is below a decodable threshold), it can still measure its received interference power.

Now suppose that at mini-slot 5 the backoff counter of the second winning sender, say s_2, expires and s_2 broadcasts a control packet during the phase 1 of mini-slot 5 **as shown in Fig.2(b)**. Assume that its corresponding receiver r_2 receives the packet successfully, but its calculated SINR is less than the SINR threshold, θ_{th}. The receiver then broadcasts a busy tone in the phase 2 of mini-slot 5 **as shown in Fig.2(c)**. When s_2 receives this busy tone, it concludes that the link cannot be activated at time slot t. Also the other nodes will not add the received interference power to their accumulated interference powers since the link is not scheduled.

In this way, a link satisfying its SINR requirement is added to set $d(t)$ during the mini-slots. Note that not only does the intended receiver of a control packet check the SINR requirement, but so do all the receivers in $d(t)$ and $x(t-1)\backslash m(t)$ check whether their receptions can still meet the SINR threshold on receipt of the control packet. If any of the above receivers cannot meet the SINR threshold, it will generate a busy tone signal, announcing that the link cannot be added to $d(t)$.

By the above processes, new links will be added to $d(t)$ as long as they do not violate the SINR threshold of the links in $d(t)$ and $x(t-1)\backslash m(t)$ as well. When the control slot is over, we obtain a feasible addition vector $d(t)$ for slot t. We then determine the final transmission schedule $\vec{x}(t)$ as

$$x_i(t) = \begin{cases} 1 & \text{with probability } 1 \quad \text{if } i \in x(t-1)\backslash m(t) \\ d_i(t) & \text{with probability } p_i \quad \text{if } i \in d(t) \\ 0 & \text{with probability } 1-p_i \quad \text{if } i \in d(t), \end{cases}$$

where $d_i(t)$ is 1 if link i is included in $d(t)$. Each link i in $d(t)$ is activated with a link activation probability p_i, which will be explained in the following subsection. During the data slot of slot t, the links in $x(t)$ transmit a data packet. To help redears' understanding, we illustrate the process of DSS algorithm as a control block diagram in Fig.3.

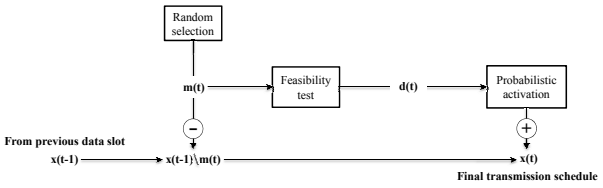
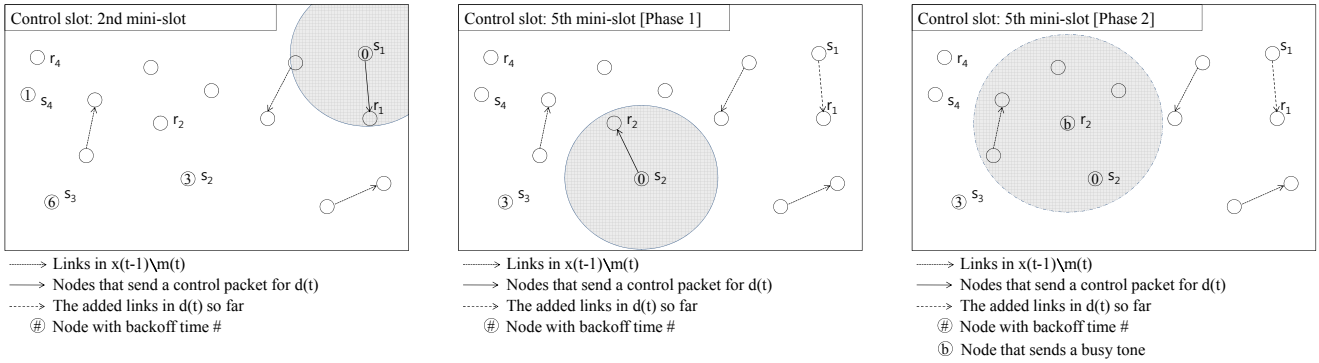


Fig. 3. Control block diagram of DSS algorithm.

3.2 Throughput optimality

In this section, we show that DSS can achieve throughput optimality. To this end by following the basic idea in [9], [11], we model the system state $x(t)$ as a DTMC. We show that the DTMC is irreducible and reversible. By carefully designing the transition probabilities, we obtain a product-form stationary distribution of the system showing throughput optimality.



(a) In the 2nd control mini-slot. (b) *Phase 1* of the 5th control mini-slot. (c) *Phase 2* of the 5th control mini-slot.

Fig. 2. **Snapshots of scheduling under DSS. (a) The 2nd mini-slot is shown a new link added successfully. (b) In the phase 1 of the 5th mini-slot, the sender s_2 broadcasts a control packet. (c) In the phase 2 of the 5th mini-slot, the corresponding receiver r_2 generates a busy tone signal since its calculated SINR becomes less than the SINR threshold, θ_{th}.**

Let $\mathcal{M}_0(m(t))$ (a subset of $\mathcal{M}$) denote the set of all the feasible schedules derived from a given random candidate vector $m(t)$. Since the transmission schedule $x(t)$ can be obtained from the previous transmission schedule $x(t-1)$ and a randomly chosen random candidate vector $m(t)$, we can model $x(t)$ as the system state of a DTMC. We derive the state transition probability between two transmission schedules $x(t-1)$ and $x(t)$ (or two system states) under DSS.

Once $d(t)$ is chosen, we can calculate the state transition probability from $x(t-1)$ to $x(t)$ as follows. We classify the action of each link in $d(t)$ into four cases:

(1) For link $i \in x(t-1)\backslash x(t)$: link i changes its state from 1 to 0 with probability $\bar{p_i}$.
(2) For link $j \in x(t)\backslash x(t-1)$: link j changes its state from 0 to 1 with probability p_j.
(3) For link $k \in d(t) \cap (x(t) \cap x(t-1))$: link k keeps its state 1, with probability p_k.
(4) For link $l \in d(t)\backslash(x(t-1) \cup x(t))$: link l keeps its state 0, with probability $\bar{p_l}$

Note that each link in $d(t)$ makes its activation decision independently of each other. Following the line of analysis in [11], we can represent the transition probability as a simple product of their activation probabilities, and thus can be written as

$$p(x(t-1), x(t)) = \mathcal{A} \cdot \mathcal{B} \cdot \mathcal{C} \cdot \mathcal{D} \cdot \mathcal{E} \cdot \mathcal{F}, \tag{2}$$

$$\mathcal{A} = \sum_{x(t-1)\cup x(t) \subseteq m(t)} \alpha(m(t)),$$

$$\mathcal{B} = \sum_{\substack{x(t-1)\cup x(t) \subseteq d(t) \subseteq m(t), \\ d(t) \subseteq \mathcal{M}_0(m(t))}} \beta(d(t)),$$

$$\mathcal{C} = (\prod_{i \in x(t-1)\backslash x(t)} \bar{p_i}), \mathcal{D} = (\prod_{j \in x(t)\backslash x(t-1)} p_j),$$

$$\mathcal{E} = (\prod_{k \in d(t) \cap (x(t) \cap x(t-1))} p_k), \mathcal{F} = (\prod_{l \in d(t)\backslash(x(t-1)\cup x(t))} \bar{p_l}), \tag{3}$$

where $\alpha(m(t))$ is the probability that a particular random candidate vector $m(t)$ is randomly selected, $\beta(d(t))$ is the probability that a particular addition vector $d(t)$ is selected from the $m(t)$ through the random backoff process, and $\bar{p_i} = 1 - p_i$. Obviously, $\sum_{m(t) \subseteq E} \alpha(m(t)) = 1$ and $\sum_{d(t) \subseteq \mathcal{M}_0(m(t))} \beta(d(t)) = 1$.

In our algorithm, links that do not interfere with any links in $x(t-1) \setminus m(t)$ can join $d(t)$. Since $d(t) \subseteq m(t)$, and both $d(t)$ and $m(t)$ are randomly determined, $\sum_{d(t) \in \mathcal{M}_0(m(t))} d(t) = E$. Then, it can be easily shown that the state of $x(t)$ has a product-form stationary distribution $\pi(x(t))$ as

$$\pi(x(t)) = \frac{1}{Z} \prod_{i \in x(t)} \frac{p_i}{\bar{p_i}}, \quad \text{where } Z = \sum_{x(t) \in \mathcal{M}} \prod_{i \in x(t)} \frac{p_i}{\bar{p_i}}. \tag{4}$$

We verify this by checking the detailed balance equation. We drop t for sake of exposition. Also, x and y indicate $x(t)$ and $x(t-1)$, respectively.

$$\begin{aligned} &\pi(x)p(x,y) \\ &= \frac{1}{Z} \prod_{i \in x} \frac{p_i}{\bar{p_i}} \sum \alpha(m) \sum \beta(d) (\prod_{i \in x \setminus y} \bar{p_i})(\prod_{i \in y \setminus x} p_i) \\ &\quad (\prod_{i \in d \cap (x \cap y)} p_i)(\prod_{i \in d \setminus (x \cup y)} \bar{p_i}). \end{aligned} \tag{5}$$

Note that from $(x \cap y) \subset d$ [11], we have $x \cap y = d \cap (x \cap y)$. Hence, we have

$$\begin{aligned} &\prod_{i \in x} \frac{p_i}{\bar{p_i}} \prod_{i \in x \setminus y} \bar{p_i} \prod_{i \in y \setminus x} p_i \\ &= (\prod_{i \in x \setminus y} \frac{p_i}{\bar{p_i}} \prod_{i \in x \cap y} \frac{p_i}{\bar{p_i}}) \prod_{i \in x \setminus y} \bar{p_i} (\prod_{i \in y} p_i \prod_{i \in y \cap x} \frac{1}{p_i}) \\ &= \prod_{i \in x \setminus y} p_i \prod_{i \in y \cap x} \frac{1}{\bar{p_i}} \prod_{i \in y} p_i \\ &= \prod_{i \in y} \frac{p_i}{\bar{p_i}} \prod_{i \in y} \bar{p_i} \prod_{i \in y \cap x} \frac{1}{\bar{p_i}} \prod_{i \in x \setminus y} p_i \\ &= \prod_{i \in y} \frac{p_i}{\bar{p_i}} \prod_{i \in y \setminus x} \bar{p_i} \prod_{i \in x \setminus y} p_i. \end{aligned} \tag{6}$$

Then, it easily follows that

$$\pi(x)p(x,y) = \pi(y)p(y,x).$$

Since the detailed balance equation holds, the DTMC is reversible with stationary distribution (4). It has been shown that CSMA-based scheduling schemes with the stationary distribution of the product form like (4) achieves optimal throughput when we set the activation probability of link l by $p_l = \frac{e^{w_l(t)}}{e^{w_l(t)}+1}$, where $w_l(t)$ is a weight function of link l at time slot t. In this paper, we use the logarithmic queue length as the weight, i.e., $w_l(t) = log(q_l(t))$, where $q_l(t)$ is a queue length of link l at time slot t. In general, a nondecreasing, continuous function of the queue length can be used [20]. Since follow the same line of analysis as in [20]; so the interested readers may refer to [11], [20] for the details of the optimality proof.

Therefore, DSS can stabilize any arrival rate in Λ, and thus the service rate will be no smaller than the arrival rate, i.e.,

$$\sum_{x\in\mathcal{M}} (\pi^*(x)x_i) \geq \lambda_i, \quad \forall i \in E \tag{7}$$

where $\pi^*(x)$ denotes the steady state probability of state x, and x_i indicates whether link i is activated..

4 Dual-state approach

Like Q-CSMA, DSS is also a CSMA-based scheduling algorithm. In general, CSMA-based throughput-optimal scheduling algorithms have somewhat poor delay performance [22]. In this section, we seek to improve the delay performance by modifying DSS in a novel way[1].

In the operation of DSS, the transmission schedule $x(t)$ is unlikely to be a maximal schedule[2], since only a subset of $d(t)$ are active in a probabilistic manner. Hence, we can add more links to $x(t)$ without violating the interference constraints; that is, we can activate the entire $d(t)$ without probabilistic selection. Clearly, additional link activation will enhance the delay performance. To this end, we extend DSS by taking a dual-state approach, called *DSS-D*. DSS-D basically maintains two kinds of states: (i) virtual states for the DTMC transition and (ii) actual states for the additional link activation.

Let $D(t)$ denote the actual state for time slot t, which is given by $D(t) = d(t) \cup (x(t-1) \setminus m(t))$. Again, $x(t)$ is the virtual state for the DTMC with the Markovian property. The main idea of the dual-state approach is that while the system is viewed as a virtual state $x(t)$, the actually activated links are $D(t)$. Note that $D(t)$ is always the superset of $x(t)$ since $x(t)$ consists of $(x(t-1) \setminus m(t))$ and a subset of $d(t)$. Also recall that all the links in $D(t)$ satisfy the interference constraints. By maintaining two kinds of states separately, we can let $x(t)$ make transitions as before (for the optimal throughput), while more links are actually activated for every data slot.

1. This approach can also be applied to Q-CSMA [11] that considers the graph-based interference model.

2. If no more link can be added to a given schedule without making any of the existing links fail, then it is called a maximal schedule.

Algorithm 2 Distributed scheduling algorithm under SINR model: A Dual-state approach (DSS-D)

1: **/* Data slot: */**
2: **IF** $d_l = 1$ **THEN**
3: With prob. 1, schedule link l, i.e., $l \in D(t)$, and send a packet during the data slot
4: **ELSE IF** backoff(l) $= 0$ **THEN**
5: With prob. 1, schedule link l, i.e., $l \in D(t)$, and send a packet during the data slot
6: **END IF**

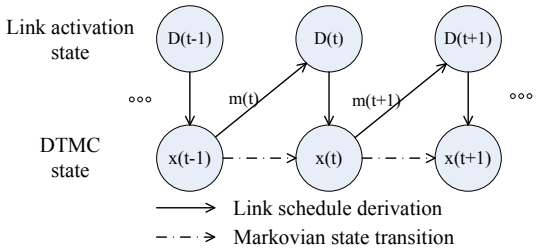


Fig. 4. Scheduling process under the dual-state approach (DSS-D). The *virtual schedule* $x(t)$ is used to determine the next transmission schedule while $D(t)$ is used for actual link activation.

Fig.4 illustrates the state transition under the dual-state approach. Like DSS, the previous virtual state is $x(t-1)$, and the random candidate vector is $m(t)$. Then, $D(t)$ is $d(t) \cup (x(t-1) \setminus m(t))$ and we need to replace lines (35-40) in Algorithm 1 with Algorithm 2. The DTMC state $x(t)$ is determined as a subset of $D(t)$ in a probabilistic manner as in Algorithm 1. The changed part of DSS-D in comparison to DSS is described in Algorithm 2.

4.1 Throughput Optimality of Dual-state approach (DSS-D)

In this section, we briefly show throughput optimality of DSS-D.

We can construct a DTMC with state $(x(t), D(t))$, since $m(t)$ and $d(t)$ (and thus $D(t)$) are chosen at random. Note that, under DSS-D, a DTMC state $x(t)$ makes transitions as exactly the same as that under DSS, which implies that (7) also holds, and thus clearly, the DTMC of DSS-D has a stationary state distribution. Let q^* denote the state distribution such that $q^*(x, D)$ is the probability of being in stationary state at (x, D), where x and D are the virtual state $x(t)$ and the actual decision schedule $D(t)$ of DSS-D, respectively. Since $\pi^*(x) = \sum_{D\in\mathcal{M}} q^*(x, D)$ and $x_i \leq D_i, \forall i \in E$, we obtain that, for each link i,

$$\begin{aligned}\sum_{x\in\mathcal{M}, D\in\mathcal{M}} q^*(x,D)D_i &= \sum_{x\in\mathcal{M}} \Big(\sum_{D\in\mathcal{M}} q^*(x,D)D_i \Big) \\ &\geq \sum_{x\in\mathcal{M}} \Big(\sum_{D\in\mathcal{M}} q^*(x,D)x_i \Big) \\ &= \sum_{x\in\mathcal{M}} \pi^*(x)x_i \geq \lambda_i,\end{aligned}$$

where the last inequality comes from (7).

4.2 An Illustration of DSS-D Operations

We now illustrate how DSS-D operates in time slot t. Thus, we have $x(t-1)$ at the beginning of the time slot t. First, each link determines whether it will be included in $m(t)$ or not. During phase 1 of of the 1st mini-slot, link $l \in x(t-1) \backslash m(t)$ broadcasts a control packet as shown in Fig.5(a). The senders of links in $m(t)$ each choose a backoff value between $[1, M-1]$, which are shown inside the circles. Suppose that three links (D, E), (F, C), (L, K) are included in $x(t-1) \setminus m(t)$. The senders of these links each broadcast a control packet. The receivers (K, E, C) measure the RSS (of their respective sender) and the interference power (of the other two transmissions), while the other nodes measure the interference power level of the three senders.

At the 2nd mini-slot, suppose A is the only sender whose backoff timer expires. During phase 1, node A sends a control packet to node B. Based on the RSS, node B calculates the SINR by taking into account the interference power measured during the 1st mini-slot. Assume that the SINR satisfies the threshold θ_{th}; node B takes no action during phase 2. Then node A schedules itself and link (A, B) is added to $d(t)$. Note that node B also has a packet to send. In this case, however, node B will not transmit a control packet when the backoff timer expires at the 4th mini-slot and at that moment it is scheduled to be a receiver already.

Suppose, at the 3rd mini-slot, node N's backoff timer expires and it sends a control packet to node M during phase 1. If the calculated SINR at node M is acceptable, the link (N, M) will be also included in $d(t)$.

An interesting event occurs at the 5th mini-slot, which is shown in Fig.5(b). Nodes G and J each have their timers expired, and they send control packets to nodes H and I, respectively. However, the control packets collide at node I, since the two received signal powers are too strong at node I. Thus, neither of the packets is decodable at node I. Then node I broadcasts a busy tone. Also, node E finds that the transmitted signal from node G causes a significant amount of interference such that its SINR is lower than θ_{th}. Then, node E also broadcasts a busy tone to reject the newly attempted links.

Finally when all the control mini-slots are over, we obtain a feasible candidate schedule $d(t)$ containing (A, B) and (N, M) as shown in Fig.5(c). From $D(t) = d(t) \cup (x(t-1) \setminus m(t))$, a total of 5 links will be activated during the data slot in time slot t. Note that a subset of $D(t)$ will be $x(t)$ depending on the link activation probabilities of links in $d(t)$ as described in DSS.

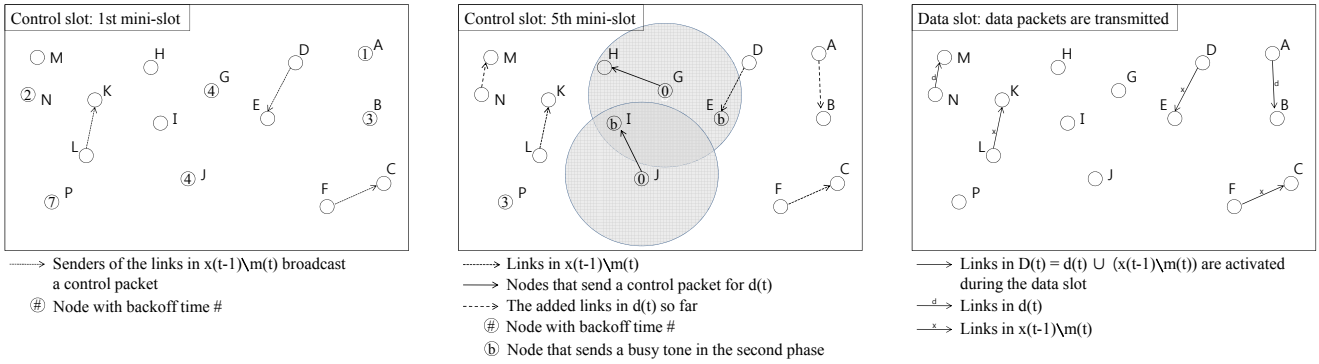
5 DSS-P

We have improved the delay performance of CSMA-based scheduling using the dual-state approach. Even though we have reduced the total queue length of the network, distributed CSMA-based scheduling schemes have another drawback: the overhead incurred due to the control mini-slots. As the traffic load increases, the performance of CSMA-based approaches becomes better as long as there are a sufficient number of control mini-slots. This overhead can be significant. Let us take the IEEE 802.11a OFDM PHY for example; the length of a single backoff slot is 9 μs. Then, the length of a single control mini-slot of DSS-family scheduling algorithms would be at least 18 μs since each control mini-slot has two phases. Since it takes 2 ms to transmit a 1500 byte long packet at 6 Mbps transmission rate (excluding the PHY/MAC header overhead for simplicity), the time overhead for the control slot exceeds the data transmission time if the number of mini-slots M is larger than 111. This control overhead becomes worse as the length of the data packet decreases and/or the transmission rate increases. Due to the random nature of the backoff process, the more links need to be scheduled, the more control mini-slots are required.

To reduce the control mini-slot overhead, we propose *DSS-P* which replaces the random backoff process by the p-persistent CSMA contention mechanism. In DSS-P, each sender in $m(t)$, attempts to transmit a control packet with the link activation probability p for each mini-slot. (The link activation probability p is the same as the link activation probability in Section 3.2.) Then we can check whether the attempting links constitute a feasible schedule by making the receivers calculate the SINR threshold as before. Whenever a feasible link set is obtained, there

Algorithm 3 DSS with p-persistent CSMA under SINR model: (DSS-P)

Initialization:
Each node includes its out-going link l in $m(t)$ with attempt probability p_a
$x(t) \leftarrow x(t-1) \backslash m(t)$
/* At the very first control mini-slot: */
  IF $l \in x(t)$ **THEN**
    Sender of link l broadcasts a control packet
  END IF
  IF Receiver of a control packet **THEN**
    Calculate $SINR$ and X_i
  ELSE
    X_i = measured signal strength
  END IF
FOR $c = 1$ to $M - 1$ **do**
/* Phase 1 */
  IF $l \in m(t) \backslash x(t)$ **THEN**
    Sender of link l broadcasts a control packet with probability p_l
  END IF
  IF Receiver of a control packet**THEN**
    Calculate $SINR$ and X_i
  ELSE
    X_i = measured signal strength
  END IF
/* Phase 2 */
  IF Link in $x(t)$ or receiver of a control packet **THEN**
    IF $SINR(or RSS/X_i) < \theta_{th}$ **THEN**
      Receiver of link generates a busy-tone.
    END IF
  END IF
  Check busy-tone.
  IF No busy-tone **THEN**
    $x(t) \leftarrow x(t) \cup \{l\}$
    $I(l, c+1) \leftarrow I(l, c) + X_i$
    Go to the end of the control mini-slot.
  ELSE
    $I(l, c+1) \leftarrow I(l, c)$
  END IF
END FOR
/* Data slot: */
IF $l \in x(t)$ **THEN**
  Send a packet with probability 1
END IF



(a) In the 1st control mini-slot. (b) In the 5th control mini-slot. (c) The links activated in the data slot.

Fig. 5. Snapshots of scheduling under DSS-D. (a) The first mini-slot is reserved to measure the SINR level from the links in $x(t-1)$. (b) The fifth mini-slot is shown a collision due to interference violation. (c) The links in $x(t-1) \setminus m(t)$ and newly added links in $d(t)$ are activated in the data slot.

are no busy tones and a feasible schedule is found. This scheduling algorithm also achieves throughput optimality, which will be shown in the following section. If obtaining a feasible schedule fails despite using all the mini-slots, the schedule used in the previous time slot will be reused. The detailed scheduling algorithm of DSS-P is described in Algorithm 3.

5.1 Throughput Optimality of DSS-P

In this section, we briefly prove how the DSS-P algorithm can achieve throughput optimality. We first show that the final set of activated links at slot t, $x(t)$, can evolve as a DTMC just like the DSS algorithm. Next, we make a product-form stationary distribution for the transition probability between two states. Then we will show that throughput optimality can be satisfied as in [11], [20].

Let $\mathcal{M}_0(m(t))$ (a subset of $\mathcal{M}$) denote the set of all the feasible schedules that can be probabilistically derived from a given random candidate vector $m(t)$. Since the transmission schedule $x(t)$ only depends on the previous transmission schedule $x(t-1)$ and the random candidate vector $m(t)$, we can model $x(t)$ as the state of a DTMC. Then, we need to derive the transition probability between two transmission schedules (or states) $x(t-1)$ and $x(t)$ under DSS-P. According to the DSS-P algorithm, the feasible link set is probabilistically determined by the link activation probability p at a certain mini-slot. Then, the determined schedule is directly used as the final transmission schedule $x(t)$. Thus, the transition probability from $x(t-1)$ to $x(t)$ is written as

$$p'(x(t-1),x(t)) = \sum_{x(t-1)\cup x(t)\subseteq m(t)} \alpha(m(t)) \cdot \mathcal{B} \cdot \mathcal{C} \cdot \mathcal{D} \cdot \mathcal{E}, \quad (8)$$

$$\mathcal{B} = \Big(\prod_{i\in x(t-1)\setminus x(t)} \bar{p_i}\Big), \mathcal{C} = \Big(\prod_{j\in x(t)\setminus x(t-1)} p_j\Big),$$

$$\mathcal{D} = \Big(\prod_{k\in m(t)\cap(x(t)\cap x(t-1))} p_k\Big), \mathcal{E} = \Big(\prod_{l\in m(t)\setminus(x(t-1)\cup x(t))} \bar{p_l}\Big), \quad (9)$$

where $\alpha(m(t))$ is the probability that a particular random candidate vector $m(t)$ is randomly selected as in Section 3.2. Then we can verify that the state distribution in Eq. (4) satisfies the following detailed balance equation:

$$\pi(x(t-1))p'(x(t-1),x(t)) = \pi(x(t))p'(x(t),x(t-1)), \quad (10)$$

similar to Eqs. (5) and (6). Then the throughput optimality of the DSS-P algorithm can be proven by following the same procedure as in [11], [20].

5.2 Extension of DSS-P

Even though DSS-P achieves throughput optimality, we can enhance its performance by opportunistically finding a schedule with a greater number activated links. In the original DSS-P (as described in Algorithm 3), the process to find a feasible schedule is over when a feasible link set is obtained for the first time even if there are still remaining control mini-slots. If these remaining control mini-slots can be exploited to find a better schedule, then the delay performance can be further improved. To this end, we continue the probabilistic attempt process to find a better schedule incrementally until the end of the control mini-slots. That is, even if we have already obtained a feasible schedule, we seek to add some of the remaining links while satisfying the feasibility to the obtained schedule.

The extended DSS-P operates as follows. After finding the first feasible link set at a certain mini-slot, more links can be added in addition to the feasible schedule at a later mini-slot as long as the SINR constraints of the already scheduled links are not violated. Note that the addition of the links to the schedule can take place multiple times. As additional links will cause more interference to the already scheduled links, we should be conservative. That is, the link activation probability, p, may as well be reduced. To increase the chances of adding more links to the schedule, we geometrically reduce the

link activation probability (of the other remaining links in $m(t)$) whenever the feasible link set is augmented. In our extended DSS-P algorithm, whenever more links are added to the schedule, other remaining candidate links reduce the current activation probability by multiplying γ ($\gamma < 1$) by p. Since this extension is similar to the dual-state approach, the throughput optimality of the extended DSS-P algorithm can be proven similarly and hence skipped.

6 Performance Evaluation

We evaluate the performance of **DSS, DSS-D, and DSS-P** with other representative scheduling schemes in the literature under the SINR based interference model. We consider a network with nodes that are placed on an area of 100 x 100 square units. We construct topologies as follows. We first randomly select the position of a sender uniformly in the area, and locate its corresponding receiver at a random place uniformly within 10 units from the sender. Repeating the locating processes, we generate two topologies; one with 49 and the other with 196 links. (That is, they have total 98 and 392 nodes, respectively.) The signal transmitted by a sender attenuates as it propagates over space. For the radio propagation model, we adopt the simple two-ray ground model [23] and all the other channel effects (e.g., short and long term fading) are not considered. At the receiver, we assume that the signal can be decoded if the SINR is over a certain threshold, and that all the links have the same SINR threshold value θ_{th}, which is set to 10 dB.

For each link, we consider single-hop traffic according the a Poisson process with (packet) arrival rate being either 0.1 or 0.9 (evenly at random). Note that the traffic load (which is a simulation parameter) should be multiplied to the packet arrival rate for the effective traffic load of each link. Our performance metric corresponding to the measured total queue lengths of all the links **and the throughput** after 5000 time slots. For each plot, we obtain the average over 20 simulation runs with different arrival rate patterns.

We evaluate the performance of CSMA-based scheduling schemes including DSS, DSS-D, DSS-H, and a centralized scheme of GMS (which is an omniscient scheduling scheme). **As DSS can be considered as Q-CSMA [11] under the SINR-based interference model, DSS-H can be considered as a counterpart of HQ-CSMA [12] under the SINR interference model, which improves the delay performance of Q-CSMA by leveraging prioritization. The operation of HQ-CSMA can be summarized as follows. The control mini-slots are divided into two parts; one part has W_0 mini-slots to activate links whose weights are greater than or equal to a certain threshold w_0 following the Q-CSMA algorithm, and the other part has the remaining ($M - W_0$) mini-slots for the rest of the links whose weights are lower than w_0 following a greedy approach (i.e., it selects the link with a larger queue length first). In our simulation, we set $W_0 = M/2$ and $w_0 = 100$. For the detailed operations of HQ-CSMA, we refer to [12].** For CSMA-based scheduling schemes, the link weight of link i is set to $w_i(t) = \log(cq_i(t))$ with a constant $c = 0.1$ as in [12], and we vary the number of control mini-slots M and the traffic load of each link.

6.1 Delay performance of DSS and DSS-D

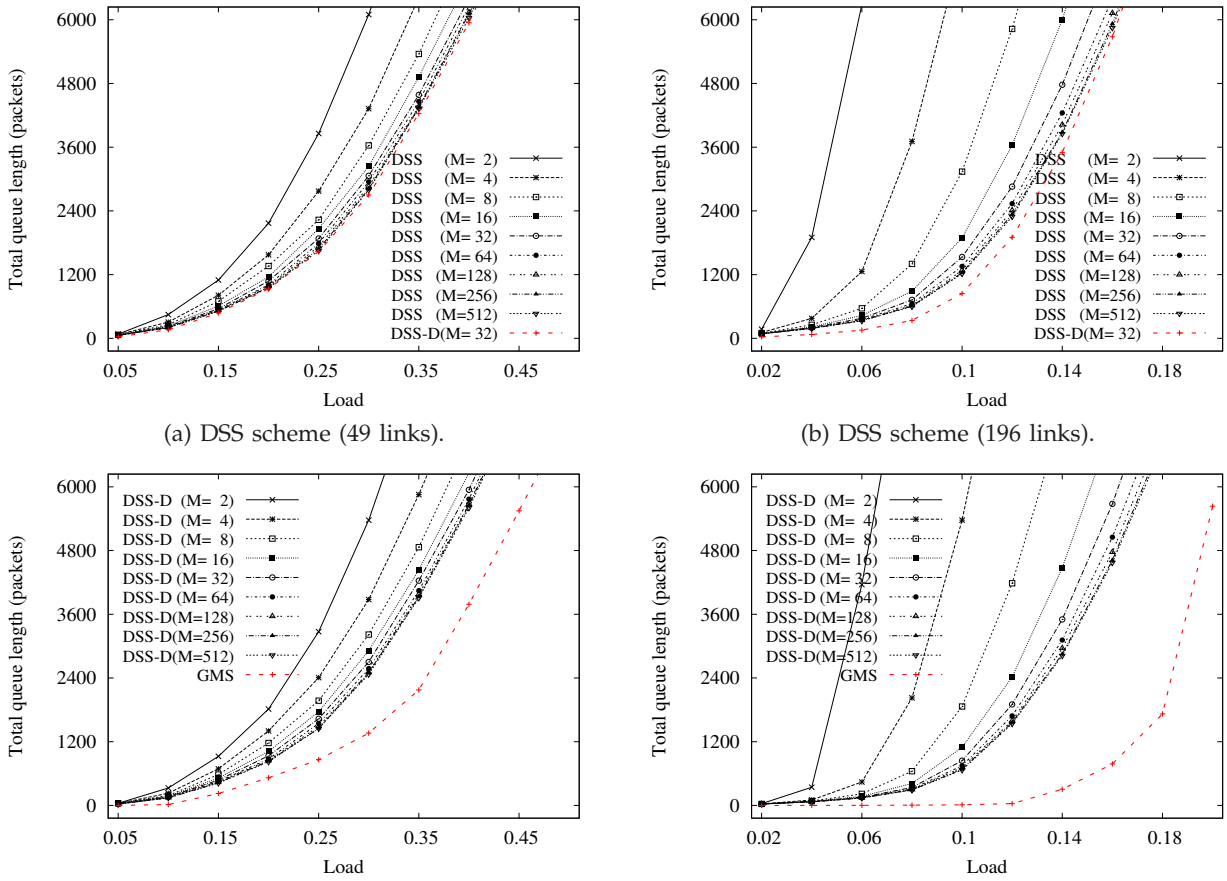
We first evaluate the **delay** performance of DSS and DSS-D for various values of M, as the traffic load increases. Figs. 6 (a) and (b) show the total queue lengths of DSS with 49 links and 196 links, respectively. We set the attempt probability p_a to 0.1, with which a link attempts to include itself in $m(t)$. As the number of control mini-slots increases, DSS shows lower delay performance in both topologies. However, for both topologies, beyond a certain value of M, the performance gains are of diminishing value. DSS-D also shows a similar trend as shown in Figs. 6 (c) and (d). However, DSS-D outperforms DSS since DSS-D with $M = 32$ has smaller queue lengths than DSS with large M.

In the following experiments, we investigate the effect of p_a on the total queue length. Intuitively, a larger value of p_a implies that more links will attempt to participate in the schedule over the random backoff process in the control slot, which may result in more links in the schedule ($d(t)$ and hence $x(t)$). However, too many links attempting to be scheduled may overwhelm the contention, which may require additional control mini-slots to resolve the contention.

Figs. 7 (a), (b), and (c) shows the results of DSS with M = 8, 16, and 128, respectively. In this case, DSS achieves better delay performance with smaller p_a, which can be interpreted as the larger p_a results in only additional contention overhead under DSS[3]. For DSS-D, we observe similar results as DSS when the number of mini-slot is small ($M < 16$). However, an interesting result is that when M is large ($M \geq 16$), DSS-D achieves better performance for the larger p_a. In this case, DSS-D successfully resolves the contention when M is sufficiently large, and hence the performance improvement from additional link activations prevail over the performance degradation from the contention.

Fig.8 compares the performance of different scheduling schemes in 49-link and 196-link networks. For each scheduling scheme, we choose the best p_a given the number of mini-slots M. In the 49-link network, DSS-D outperforms DSS and DSS-H. In particular, when the number of control mini-slot is large ($M \geq 256$), the performance of DSS-D is close to that of the centralized GMS scheme. In the 196-link network, both DSS-D and DSS-H significantly outperform DSS and only a slight difference is observed between DSS-D and DSS-H. This is partly because that DSS-H uses the DSS algorithm for a fraction

3. We also conduct simulations with probabilities less than 0.1. The results are similar to those in Fig.7, except for extremely small probabilities like 0.01.



(a) DSS scheme (49 links).

(b) DSS scheme (196 links).

(c) DSS-D scheme (49 links).

(d) DSS-D scheme (196 links).

Fig. 6. Performance of DSS and DSS-D with 49 links and 196 links as the traffic load increases. As the number of control mini-slots, M, increases, both DSS and DSS-D show lower delay performance in both topologies. DSS-D with $M = 32$ outperforms DSS with any number of control mini-slots.

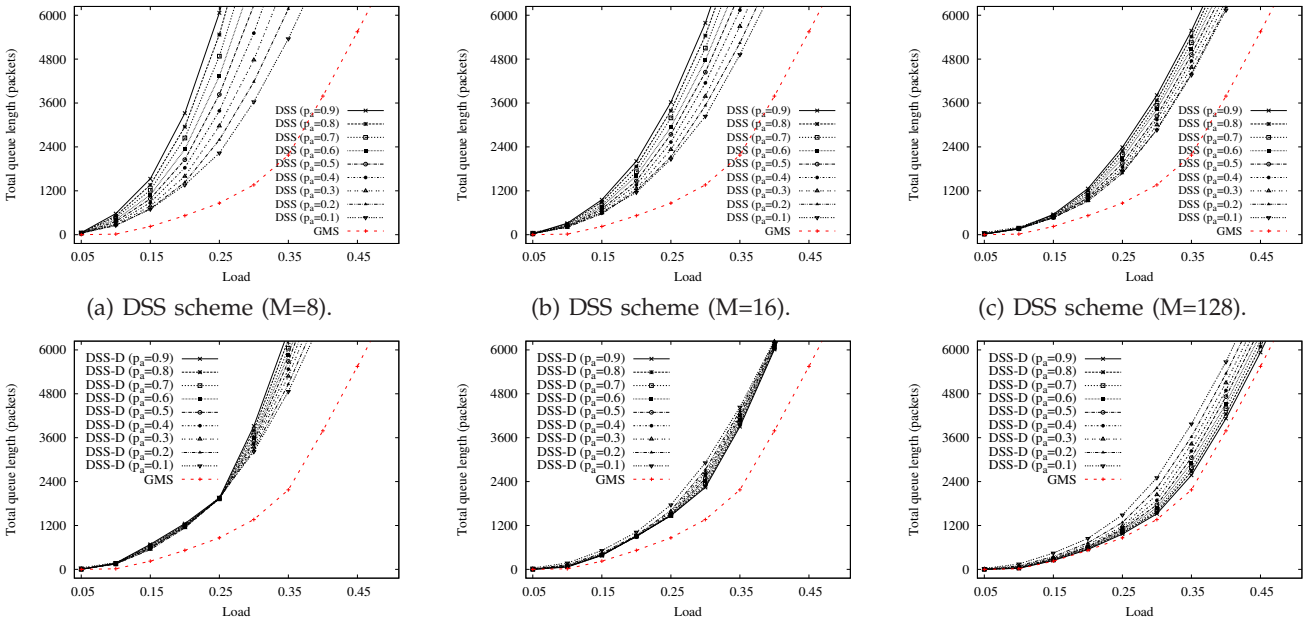
of control mini-slots (i.e., W_0). When there are only 49 links, most links have a small queue length less than w_0, and thus DSS-H cannot effectively exploit W_0 control mini-slots, which results in low performance with the 49-link network. In the 196-link network, however, 4 times more links are available, thus it will increase the number of the links whose queue lengths are larger than w_0. Therefore, DSS-H works more efficiently as the network size becomes larger and/or the traffic load increases.

6.2 Delay performance of DSS-P

In this section, we evaluate the **delay** performance of DSS-P. Since DSS-P shows the best performance with $p = 0.1$, we set the link activation probability p to 0.1 and simulate DSS-P by changing M, as the traffic load increases. For the other schemes, we also choose the best system parameters for the given the number of mini-slots M.

Fig.9 illustrates the total queue length of DSS-P under the networks of 49 links and 196 links with various number of control mini-slots. DSS-P exhibits similar performance to DSS-D or DSS in Fig.6. It also has a better delay performance as the number of control mini-slots increases. Note that DSS-P achieves good performance even with small number of control mini-slots. This implies that DSS-P utilizes control mini-slots more efficiently by leveraging the p-persistent CSMA mechanism. However, the performance of DSS-P is not notably higher than DSS-D or DSS when each scheme has a sufficient number of control mini-slots.

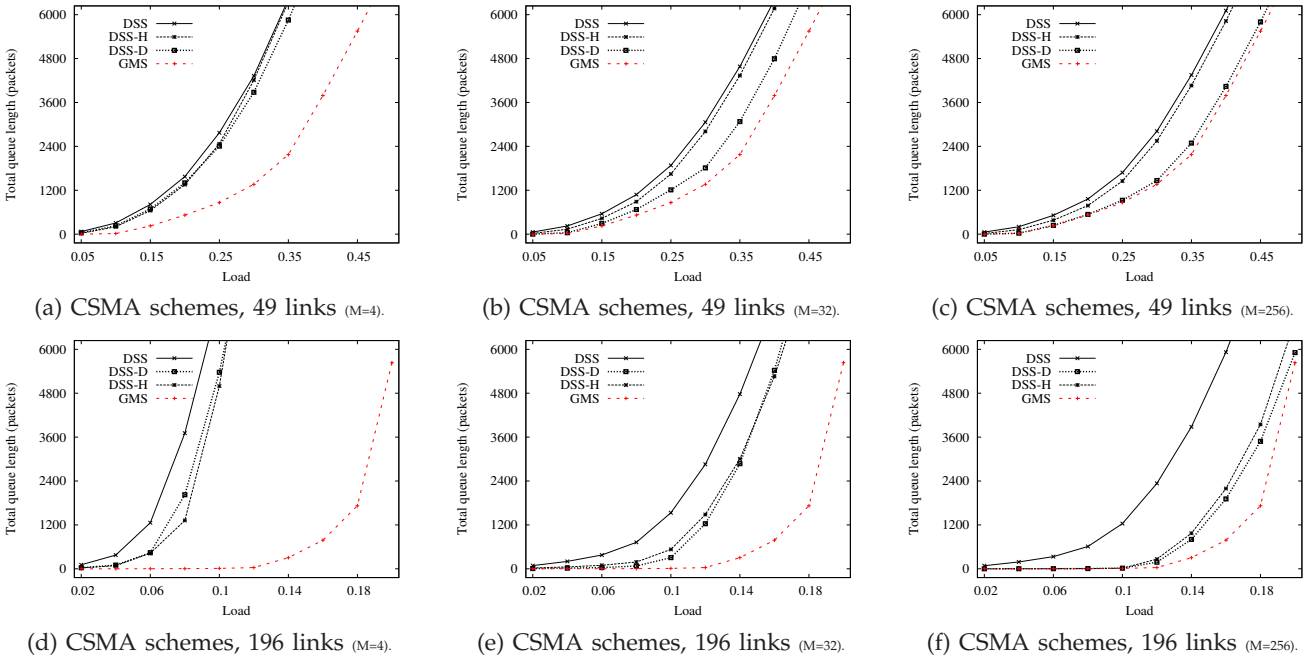
Fig.10 illustrates the total queue lengths of the CSMA scheduling schemes and GMS in the 49-link and 196-link network topologies. We only consider small numbers of control mini-slots (4, 8, and 16) to see how the CSMA-based scheduling schemes performs with small M. Since other CSMA-based scheduling schemes rely on random backoff operations, they do not achieve good performance with small number of control mini-slots in both 49-link and 196-link topologies. The performance is getting worse when the number of links is high due to the higher contention overhead. DSS-P achieves much better delay performance than other CSMA-based scheduling schemes since all of selected candidate links in $m(t)$ attempt to be schedule at each control mini-slot.



(a) DSS scheme (M=8). (b) DSS scheme (M=16). (c) DSS scheme (M=128).

(d) DSS-D scheme (M=8). (e) DSS-D scheme (M=16). (f) DSS-D scheme (M=128).

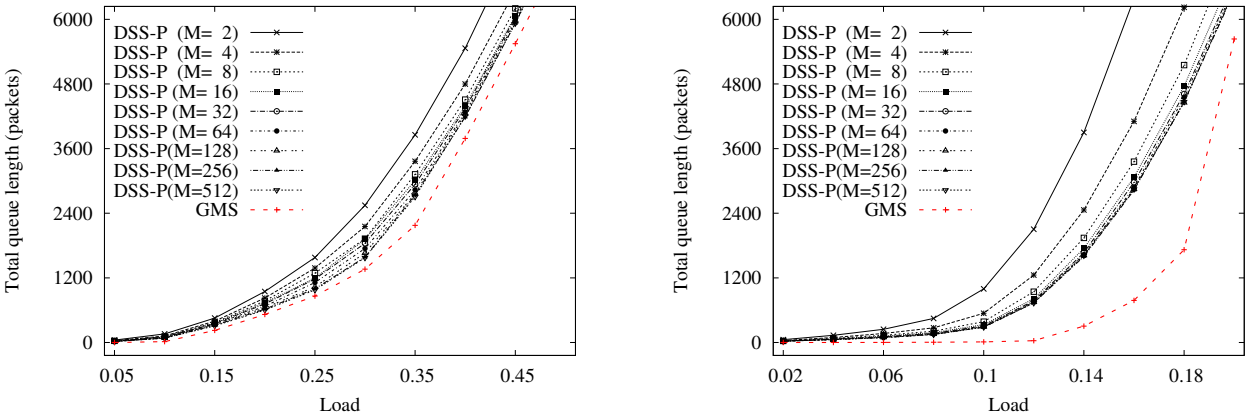
Fig. 7. Performance of the DSS and DSS-D schemes with 49 links is shown as p_a increases. DSS achieves better delay performance with smaller p_a. DSS-D also shows better delay performance as the attempt probability p_a decreases when the number of control mini-slots (M) is less than 16. However, when M is 16 or higher, DSS-D achieves the increasingly better performance as p_a increases.



(a) CSMA schemes, 49 links (M=4). (b) CSMA schemes, 49 links (M=32). (c) CSMA schemes, 49 links (M=256).

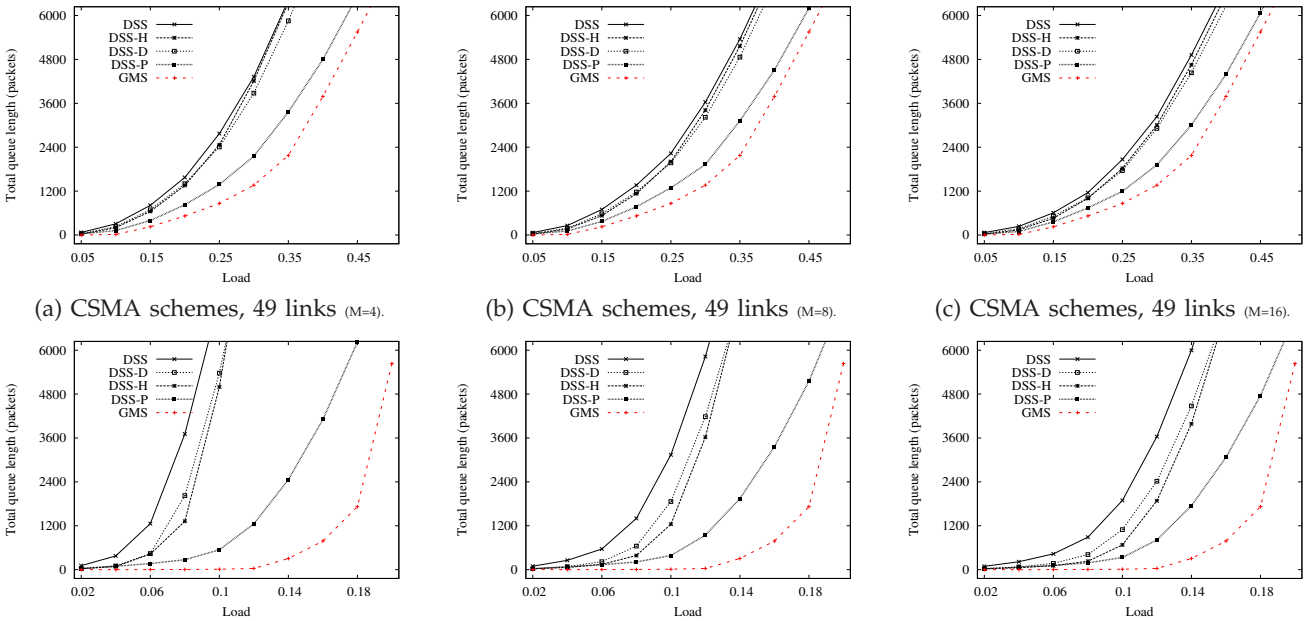
(d) CSMA schemes, 196 links (M=4). (e) CSMA schemes, 196 links (M=32). (f) CSMA schemes, 196 links (M=256).

Fig. 8. Comparison of the proposed DSS-D scheme with DSS, DSS-H, and GMS in case of 49-link and 196-link topologies. In the 49-link network, when the number of control mini-slots is small, there is little difference among the CSMA-based schemes. As the number of control mini-slots increases, however, DSS-D outperforms both DSS and DSS-H. In the 196-link network, as the number of control mini-slots increases, both DSS-D and DSS-H outperform DSS. Unlike the results of 49-link network, there is marginal difference between DSS-D and DSS-H.



(a) DSS-P scheme (49 links). (b) DSS-P scheme (196 links).

Fig. 9. Performance of DSS-P in the networks of 49 links and 196 links with different numbers of control mini-slots. Note that, as M increases, the performance of DSS-P converges rapidly compared to other schemes in the previous sections. It achieves comparable performance with smaller number of control mini-slots (less than 16), in contrast to other CSMA-based scheduling algorithms.



(a) CSMA schemes, 49 links (M=4). (b) CSMA schemes, 49 links (M=8). (c) CSMA schemes, 49 links (M=16).

(d) CSMA schemes, 196 links (M=4). (e) CSMA schemes, 196 links (M=8). (f) CSMA schemes, 196 links (M=16).

Fig. 10. Comparison of DSS-P with DSS, DSS-H, DSS-D, and GMS in case of 49-link and 196-link networks with small number of the control mini-slots. In the 49-link topology, DSS-P outperforms all other CSMA-based scheduling schemes and the performance gap from GMS is not substantial. In the 196-link topology, DSS-P still achieves the better performance than the other CSMA-based scheduling schemes.

When the number of contending links is 196 in Fig.10 (d), (e), and (f), DSS-P achieves substantially lower performance than GMS due to the high contention overhead. However, DSS-P still shows significantly better delay performance than other CSMA-based scheduling schemes.

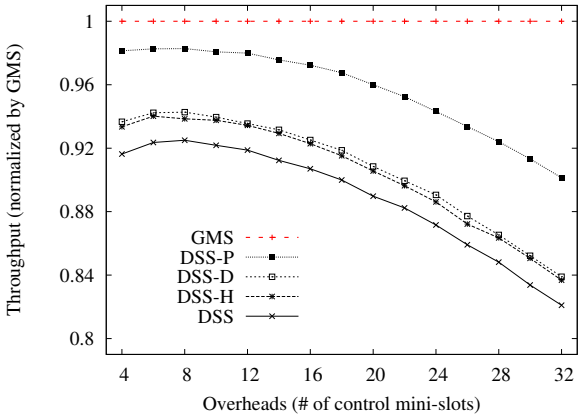
6.3 Throughput performance comparison

We evaluate throughput performance of each scheduling scheme with different control mini-slots overhead. In this particular experiment, we define the *throughput* metric as the ratio of served packets to generated packets, i.e.,

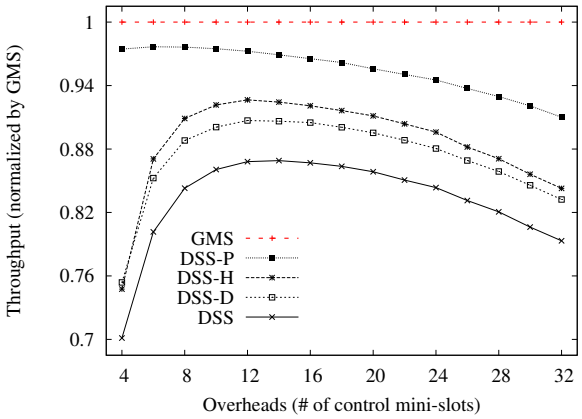
$$\frac{(\textit{Total number of served packets of all the links})}{(\textit{Total number of generated packets of all the links})}.$$

We set our simulation following the IEEE 802.11a OFDM PHY model. A single control mini-slot of DSS-family scheduling algorithms is set to 18μs, since each control mini-slot has two phases. We assume that the size of each packet is 1000 byte long and data rate is 6 Mbps. We exclude the PHY/MAC header overhead for simplicity. In Fig.11, we normalize the throughput of each scheduling algorithm with respect to GMS. We use the traffic load parameter 0.25 and 0.1 for 49-link topology and 196-link topology, respectively.

Due to the small number of control mini-slot and low traffic load, DSS-P outperforms the other CSMA schemes in both network topologies. As the number of control mini-slot increases, its throughput performance decreases, especially when $M > 8$ in both network topologies. Other CSMA schemes also show a similar pattern as DSS-P with 49-link topology. However, they are different from DSS-P with 196-link topology. When the number of control mini-slot is small (i.e., $M < 8$), the throughput gain is much larger than the control mini-slot overhead, but when the number of control mini-slot becomes larger (i.e., $M > 12$), the performance starts decreasing, since the gain becomes smaller than the control mini-slots overhead.



(a) CSMA schemes, 49 links with load 0.25.



(b) CSMA schemes, 196 links with load 0.1.

Fig. 11. **Throughput performance of each scheduling algorithm with different control mini-slot overheads (normalized by GMS). All CSMA schemes show decreasing throughput performance when the number of control mini-slot becomes larger in both 49-link and 196-link topologies.**

7 Conclusion

In this paper we investigate the scheduling problem in multi-hop wireless networks under realistic SINR-based interference model. We first develop a fully distributed throughput optimal base-line scheme, called DSS, that leverages carrier sensing and exploits recent result in throughput optimality. We then extend this scheme in various ways. We improve delay performance by developing DSS-D that separates activation states for data transmission from virtual states for state transition, thus reducing the delay while achieving throughput optimality. We also propose DSS-P to reduce the control overhead. DSS-P replaces a random backoff process of DSS (or DSS-D) with p-persistent CSMA. We show that DSS-P is throughput-optimal and uses far less overhead, outperforming the other CSMA-based schemes for a large class of topologies.

There are many open problems in scheduling under the SINR model. Since most recent communication

technologies allow rate adaptation and/or variable packet sizes based on the received SINR level, it is interesting to develop CSMA based schemes that achieve high performance under time-varying link rates. Understanding transitions of Markov Chain state and achieving convergence of state distribution are of particular interest. Also, getting timely SINR feedback from receivers will increase overhead and requires further research to reduce the complexity.

REFERENCES

[1] J. Ryu, C. Joo, T. T. Kwon, N. B. Shroff and Y. Choi, "Distributed SINR based scheduling algorithm for multi-hop wireless networks," in ACM MSWiM, 2010.

[2] L. Tassiulas and A. Ephremides, "Stability properties of constrained queueing systems and scheduling policies for maximum throughput in multihop radio networks," in IEEE Transactions on Automatic Control, vol. 36, pp. 1936-1948, 1992.

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

[4] E. Leonardi, M. Mellia, F. Neri, and M. A. Marsan, "On the stability of input-queued switches with speed-up," in IEEE/ACM Trans. Netw., vol. 9, no. 1, pp. 104-118, 2001.

[5] N. McKeown, "Scheduling algorithms for input-queued cell switches," Ph.D. dissertation, Univ. California, Berkeley, 1995.

[6] A. Dimakis and J. Walrand, "Sufficient conditions for stability of longest-queue-first scheduling: Second-order properties using fluid limits," in Adv. Appl. Probab., vol. 38, no. 2, pp. 505-521, 2006.

[7] J.-H. Hoepman, "Simple distributed weighted matchings," Oct. 2004 [Online]. http://arxiv.org/abs/cs/0410047v1

[8] C. Joo, "A Local greedy scheduling scheme with provable performance guarantee," in ACM MobiHoc, 2008.

[9] L. Jiang and J. Walrand, "A distributed CSMA algorithm for throughput and utility maximization in wireless networks," in Allerton Conference, 2008.

[10] P. Marbach and A. Eryilmaz, "A backlog-based csma mechanism to achieve fairness and throughput-optimality in wireless networks," in Allerton Conference, 2008.

[11] J. Ni and R. Srikant, "Distributed CSMA/CA algorithms for achieving maximum throughput in wireless networks," in ITA, 2009.

[12] J. Ni, B. Tan, and R. Srikant, "Q-CSMA: Queue-length based CSMA/CA algorithms for achieving maximum throughput and low delay in wireless networks," in IEEE INFOCOM 2010.

[13] J. Ghaderi and R. Srikant, "On the design of efficient CSMA algorithms for wireless networks," in IEEE CDC 2010.

[14] Q. Li and R. Negi, "Distributed throughput-optimal scheduling in ad hoc wireless networks," in IEEE ICC, 2011.

[15] M. dinitz, "Distributed algorithms for approximating wireless network capacity," in IEEE INFOCOM 2010.

[16] E. I. Ásgeirsso and P. Mitra, "On a game theoretic approach to capacity maximization in wireless networks," in IEEE INFOCOM 2011.

[17] G. Pei and V. S. A. Kumar, "Distributed link scheduling under the physical interference model," to appear in IEEE INFOCOM 2012.

[18] C. Joo and N. B. Shroff, "Performance of Random Access Scheduling Schemes in Multi-hop Wireless Networks," in IEEE/ACM Trans. Netw., vol. 17, no. 5, pp. 1481–1493, 2009.

[19] N. Ehsan and R. Cruz, "On the optimal SINR in random access networks with spatial reuse," in IEEE CISS, 2006.

[20] A. Eryilaz, R. Srikant, and J. R. Perkins, "Stable scheduling policies for fading wireless channels," in IEEE/ACM Trans. Netw., vol. 13, no. 2, pp. 411-424, 2005.

[21] L. Qiu, Y. Zhang, F. Wang, M. K. Han, and R. Mahajan, "A general model of wireless interference," in ACM MobiCom, 2007.

[22] D. Shah, D. Tse, and J. N. Tsitsiklis, "Hardness of low delay network scheduling," submitted to IEEE Trans. Information Theory, 2009.

[23] T. Rappaport, "Wireless communications: Principles and practice," Prentice Hall PTR, pp. 120-125, 2001.

Jiho Ryu (S07) received the B.S. degree in computer science (major) and mathematics (minor) from Korea Advanced Institute of Science and Technology (KAIST), Daejeon, in 2005. Currently, he is working towards the Ph.D. degree at the School of Computer Science and Engineering, Seoul National University, Seoul, Korea. His research interests include wireless scheduling, wireless LANs, sensor networks, wireless mesh networks, and real-time systems.

Changhee Joo (S98-M05) received his Ph.D degree from the school of Electrical Engineering and Computer Science, Seoul National University, Korea, 2005. He joined the Center of Wireless Systems and Applications, Purdue University, USA, as an associated researcher and moved to the Ohio State University, USA, in 2007. From September 2011, he is with Ulsan National Institute of Science and Technology (UNIST), Korea. His research interests include communication systems, resource allocation with focus on scheduling, cross-layer optimization, data aggregation in wireless sensor networks, congestion control, and active queue management. He is a member of IEEE, and a recipient of the IEEE INFOCOM 2008 best paper award.

Ted "Taekyoung" Kwon (M00) received the B.S., M.S., and Ph.D. degrees in computer engineering from Seoul National University, Seoul, Korea, in 1993, 1995, and 2000, respectively. He was a Visiting Student with the IBM T. J. Watson Research Center, Yorktown Heights, NY, in 1998 and a Visiting Scholar with the University of North Texas, Denton, in 1999. He is currently an Associate Professor with the Multimedia and Mobile Communications Laboratory, School of Computer Science and Engineering, Seoul National University. His recent research interests include radio resource management, wireless technology convergence, mobility management, and wireless sensor networks

Ness B. Shroff (S91 / M93 / SM01/ F07) is currently the Ohio Eminent Scholar of Networking and Communications, and Professor of ECE and CSE at The Ohio State University. Previously, he was a Professor of ECE at Purdue University and the director of the Center for Wireless Systems and Applications (CWSA), a university-wide center on wireless systems and applications. His research interests span the areas of wireless and wireline communication networks, where he investigates fundamental problems in the design, performance, pricing, and security of these networks. Dr. Shroff has received numerous awards for his networking research, including the NSF CAREER award, the best paper awards for IEEE INFOCOM 06 and IEEE INFOCOM08, the best paper award for IEEE IWQoS06, the best paper of the year award for the Computer Networks journal, and the best paper of the year award for the Journal of Communications and Networks (JCN) (his IEEE INFOCOM05 paper was one of two runner-up papers). He also currently serves as a guest chaired professor of Wireless Communications in the department of Electronic Engineering, at Tsighnua University, in Beijing, China.

Yanghee Choi (SM99) received the B.S. degree in electronics engineering from Seoul National University, Seoul, Korea, in 1975, the M.S. degree in electrical engineering from the Korea Advanced Institute of Science and Technology, Daejeon, Korea, in 1977, and the Ph.D. degree of engineering in computer science from the Ecole Nationale Superieure des Telecommunications, Paris, France, in 1984. From 1977 to 1991, he was with the Electronics and Telecommunications Research Institute, where he served as the Director of the Data Communication Section, and the Protocol Engineering Center. From 1981 to 1984, he was a Research Student with the Center National dEtude des Telecommunications, Issy-les-Moulineaux. From 1988 to 1989, he was a Visiting Scientist with the IBM T. J. Watson Research Center, Yorktown Heights, NY. Since 1991, he has been with the School of Computer Engineering, Seoul National University, where he is currently leading the Multimedia and Mobile Communications Laboratory. His research interest includes future Internet. From 2009 to 2011, he was the President of the Korean Institute of Information Scientists and Engineers. Dr. Choi is the Chair of the Future Internet Forum. He is a regular member of the National Academy of Engineering of Korea and the Korean Academy of Science and Technology, Seongnam.