Performance of Low-Complexity Greedy Scheduling Policies in Multi-Channel Wireless Networks: Optimal Throughput and Near-Optimal Delay

Bo Ji, Gagan R. Gupta, Xiaojun Lin, and Ness B. Shroff

Abstract*—In this paper, we focus on the scheduling problem in multi-channel wireless networks, e.g., the downlink of a single cell in fourth generation (4G) OFDM-based cellular networks. Our goal is to design efficient scheduling policies that can achieve *provably good performance* in terms of both *throughput* and *delay*, at a *low complexity*. While a recently developed scheduling policy, called *Delay Weighted Matching (DWM)*, has been shown to be both rate-function delay-optimal (in the many-channel many-user asymptotic regime) and throughput-optimal (in general non-asymptotic setting), it has a high complexity $O(n^5)$, which makes it impractical for modern OFDM systems. To address this issue, we first develop a simple greedy policy called *Delay-based Queue-Side-Greedy (D-QSG)* with a *lower complexity* $O(n^3)$, and rigorously prove that D-QSG not only achieves *throughput optimality*, but also guarantees *near-optimal rate-function-based delay performance*. Specifically, the rate-function attained by D-QSG for any fixed integer threshold $b > 0$, is no smaller than the maximum achievable rate-function by any scheduling policy for threshold $b - 1$. Further, we develop another simple greedy policy called *Delay-based Server-Side-Greedy (D-SSG)* with an even lower complexity $O(n^2)$, and show that D-SSG achieves the same performance as D-QSG. Thus, we are able to achieve a dramatic reduction in complexity (from $O(n^5)$ of DWM to $O(n^2)$) with a minimal drop in the delay performance. Finally, we conduct numerical simulations to validate our theoretical results in various scenarios. The simulation results show that our proposed greedy policies not only guarantee a near-optimal rate-function, but also empirically are virtually indistinguishable from the delay-optimal policy DWM.

I. Introduction

In this paper, we consider the scheduling problem in a multi-channel wireless network, where the system has a large bandwidth that can be divided into multiple orthogonal sub-bands (or channels). A practically important example of such a multi-channel network is the downlink of a single cell of a fourth generation (4G) OFDM-based wireless cellular system (e.g., LTE and WiMax). In such a multi-channel system, a key challenge is *how to design efficient scheduling policies that can simultaneously achieve high throughput and low delay?* This problem becomes extremely critical in OFDM systems that are expected to meet the dramatically increasing demands from multimedia applications with more stringent Quality-of-Service (QoS) requirements (e.g., voice and video applications), and thus look for new ways to achieve higher data rates, lower latencies, and a much better user experience. Yet, an even bigger challenge is *how to design such high-performance scheduling policies at a low complexity?* For example, in OFDM systems, the *Transmission Time Interval (TTI)*, within which the scheduling decisions need to be made, is typically on the order of a few milliseconds. On the other hand, there are hundreds of orthogonal channels that need to be allocated to hundreds of users. Hence, the scheduling decision has to be made within a very short scheduling cycle.

B. Ji is with Department of ECE at the Ohio State University. X. Lin is with School of ECE at Purdue University. N. B. Shroff is with Departments of ECE and CSE at the Ohio State University. Emails: ji@ece.osu.edu, gagan.gupta@iitdalumni.com, linx@ecn.purdue.edu, shroff@ece.osu.edu.

This work has been supported in part by the Army Research Office MURI Award W911NF-08-1-0238; the NSF grants CNS-1065136, CNS-1012700, and CNS-0643145; and Intel Corporation.

We consider a single-cell multi-channel system consisting of n channels and a proportionally large number of users, with intermittent connectivity between each user and each channel. We assume that the Base Station (BS) maintains a separate First-in First-out (FIFO) queue associated with each user, which buffers the packets for the user to download. The delay performance that we focus on in this paper is the probability that the largest packet waiting time (or delay) in the system exceeds a certain fixed threshold. Such a probability can be estimated by its asymptotic *decay-rate* (or called *rate-function* in large-deviations theory) when n becomes large. We refer to this setting as the *many-channel many-user asymptotic regime*.

A number of recent works have considered a multi-channel system similar to ours, but looked at delay from different perspectives. A line of works focused on queue-length-based metrics: average queue length [1] or queue-length rate-function in the many-channel many-user asymptotic regime [2]–[5]. In [1], the authors focused on minimizing cost functions over a finite horizon, which includes minimizing the expected total queue length as a special case. The authors showed that their goal can be achieved in two special scenarios: 1) a simple two-user system, and 2) systems where fractional server allocation is allowed. In [2]–[5], delay performance is evaluated by the queue-overflow probability, and its associated rate-function, i.e., the asymptotic decay-rate of the probability that the largest queue length in the system exceeds a fixed threshold. Although [2] and [5] proposed scheduling policies that can guarantee both throughput optimality and rate-function optimality, they suffer from the following shortcomings. First, although the decay-rate of the queue-overflow probability may be mapped to that of the delay-violation probability when the arrival process is deterministic with a constant rate [6], this is not true in general, especially when the arrivals are correlated over time. Further, [7] and [8] have shown through simulations that good queue-length performance does not necessarily imply good delay performance. Second, their results on rate-function optimality strongly rely on the assumptions that the arrival process is *i.i.d. not only across users, but also in time*, and that per-user arrival at any time is no greater than the largest channel rate. Third, even under this more restricted model, their proposed algorithms with rate-function optimality are of complexity at least $O(n^3)$. For more general models, no algorithms with provable rate-function optimality are provided.

Similar to this paper, another line of work [7] directly focused on the delay performance rather than the queue-length performance. The performance of delay is often harder to characterize, because the delay in a queueing system often does not admit a Markovian representation, even for simple M/M/1 queues. The problem becomes even harder in a multi-user system with fading channels and interference constraints, since the service rate for individual queues becomes more unpredictable. In [7], the authors developed a scheduling policy called Delay Weighted Matching (DWM), which maximizes the sum of the delay of the packets scheduled in each time-slot. It has been shown that DWM is not only throughput-optimal, but also rate-function delay-optimal in many cases (i.e., maximizing *delay rate-function*, rather than *queue-length rate-function* as considered in [2]–[5].) However, DWM incurs a high complexity $O(n^5)$, which renders it impractical for modern OFDM systems with many channels and users (e.g., on the order of hundreds). Hence, scheduling policies with a lower complexity are preferred in such multi-channel systems.

This leads to the following natural but important questions: *Can we find scheduling policies that have a significantly lower complexity, with comparable or only slightly worse performance? How much complexity can we reduce, and how much performance do we need to sacrifice?* In this paper, we answer these questions positively. Specifically, we develop *low-complexity* greedy policies that achieve both *throughput optimality* and *rate-function near-optimality*.

We summarize our main contributions as follows.

First, we propose a greedy scheduling policy, called *Delay-based Queue-Side-Greedy (D-QSG)*, which has a *lower complexity* $O(n^3)$ compared to $O(n^5)$ of DWM. D-QSG, in an iterative manner, schedules the oldest packets remaining in the system one-by-one whenever possible. We rigorously prove that D-QSG not only achieves throughput optimality, but also guarantees a near-optimal rate-function. Specifically, the rate-function attained by D-QSG for any fixed integer threshold $b > 0$, is *not only positive but also no smaller than the maximum achievable rate-function by any scheduling policy for threshold* $b-1$. We obtain this result by comparing D-QSG with a new *Greedy Frame-Based Scheduling (G-FBS)* policy that can exploit a key property of D-QSG. We show that G-FBS policy guarantees a near-optimal rate-function, and that D-QSG dominates G-FBS in every sample-path.

Second, we propose another greedy scheduling policy, called *Delay-based Server-Side-Greedy (D-SSG)*, which has an even *lower complexity* $O(n^2)$. D-SSG, also in an iterative manner, allocates servers one-by-one to serve a connected queue that has the largest head-of-line (HOL) delay. Note that the queue-length-based counterpart of D-SSG, called Q-SSG, has been studied in [3], [4]. There, however, the authors were only able to prove a positive (queue-length) rate-function for restricted arrival processes that are *i.i.d.* not only across users, but also in time. On the contrary, we show that D-SSG achieves the same performance as D-QSG, by proving that D-SSG and D-QSG are sample-path equivalent under certain tie-breaking rules. Thus, we are able to achieve a dramatic reduction in complexity (from $O(n^5)$ of DWM to $O(n^2)$) with a minimal drop in the delay performance.

Finally, we conduct numerical simulations to validate our theoretical results in various scenarios. Our simulation results show that our proposed greedy policies *not only guarantee a near-optimal rate-function, but also empirically are virtually indistinguishable from the delay-optimal policy DWM*. Further, the simulation results also show that *D-SSG consistently outperforms its queue-length-based counterpart Q-SSG in all scenarios that we consider.*

The remainder of the paper is organized as follows. In Section II, we describe the details of our system model and performance metrics. In Section III, we derive an upper bound on the rate-function that can be achieved by any scheduling policy. Then, in Sections IV and V, we present our main results on throughput optimality and near-optimal rate-function for our proposed low-complexity greedy policies. Further, we conduct numerical simulations in Section VI. Finally, we make concluding remarks in Section VII.

Due to space limitations, the detailed proofs are omitted and provided in our online technical report [9].

II. System Model

We consider a discrete-time model for the downlink of a single-cell multi-channel wireless network with n orthogonal channels and n users. In each time-slot, a channel can be allocated only to one user, but a user can be allocated with multiple channels simultaneously. *As in [2]–[5], [7], for ease of presentation, we assume that the number of users is equal to the number of channels. Our rate-function delay analysis follows similarly if the number of users scales linearly with the number of channels.* We let Q_i denote the FIFO queue associated with the i-th user, and let S_j denote the j-th server[1]. We consider the following *i.i.d.* ON-OFF channel model that has also been used in the previous works (e.g., [1]–[5], [7]). In such a model, the connectivity between each queue and each server change between ON and OFF from time to time. We assume that the perfect channel state information (i.e., whether each channel is ON or OFF for each user in each time-slot) is known at the BS. This is a reasonable assumption in the downlink scenario of a single cell in a multi-channel cellular system with dedicated feedback channels. We also assume unit channel capacity, i.e., at most one packet from Q_i can be served by S_j when the connectivity between Q_i and S_j is ON. *This assumption of unit channel capacity is made for ease of exposition, and our analysis can be readily extended to a 0-K channel model (where the channel capacity is K packets*

[1]Throughout this paper, we use the terms "user" and "queue" interchangeably, and use the terms "channel" and "server" interchangeably.

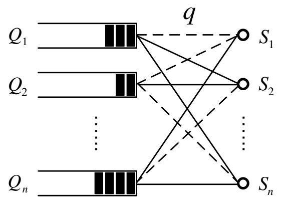

Fig. 1. System model. The connectivity between each pair of queue Q_i and server S_j is "ON" (denoted by a solid line) with probability q, and "OFF" (denoted by a dashed line) otherwise.

per time-slot when a channel is ON). Let $C_{i,j}(t)$ denote the connectivity between queue Q_i and server S_j in time-slot t. Then, $C_{i,j}(t)$ can be modeled as a Bernoulli random variable with a parameter $q \in (0,1)$, i.e.,

$$C_{i,j}(t) = \begin{cases} 1, & \text{with probability } q, \\ 0, & \text{with probability } 1-q. \end{cases}$$

We assume that all the random variables $C_{i,j}(t)$ are *i.i.d.* across all the variables i, j and t. Such a network can be modeled as a multi-queue multi-server system with stochastic connectivity, as shown in Fig. 1.

As in the previous works [1]–[3], [7], the above *i.i.d.* ON-OFF channel model is a simplification, and is assumed only for the analytical results. The ON-OFF model is a good approximation when the BS transmits at a fixed achievable rate if the SINR level is above a certain threshold at the receiver, and does not transmit otherwise. The sub-bands being *i.i.d.* is a reasonable assumption when the channel width is larger than the coherence bandwidth of the environment. Moreover, we believe that our results obtained for this simple channel model can provide useful insights for more general models. Indeed, we will show through simulations that our proposed greedy policies also perform well in more general models, e.g., accounting for heterogeneous (near- and far-)users and time-correlated channels. Further, we will briefly discuss how to design efficient scheduling policies in general scenarios towards the end of this paper.

We present more notations used in this paper as follows. Let $A_i(t)$ denote the number of packet arrivals to queue Q_i in time-slot t. Let $A(t) = \sum_{i=1}^n A_i(t)$ denote the cumulative arrivals to the entire system in time-slot t, and let $A(t_1, t_2) = \sum_{\tau=t_1}^{t_2} A(\tau)$ denote the cumulative arrivals to the system from time t_1 to t_2. We let λ_i denote the mean arrival rate to queue Q_i, and let $\lambda \triangleq [\lambda_1, \lambda_2, \dots, \lambda_n]$ denote the arrival rate vector. We assume that packets arrive at the beginning of a time-slot, and depart at the end of a time-slot. We use $Q_i(t)$ to denote the length of queue Q_i at the beginning of time-slot t immediately after packet arrivals. Queues are assumed to have an infinite buffer capacity. Let $Z_{i,l}(t)$ denote the delay of the l-th packet at queue Q_i at the beginning of time-slot t, *which is measured since the time when the packet arrived to queue Q_i until the beginning of time-slot t*. Note that at the end of each time-slot, the packets that are still present in the system will have their delays increased by one due to the elapsed time. Further, let $W_i(t) = Z_{i,1}(t)$ denote the HOL delay of queue Q_i at the beginning of time-slot t. Finally, we define $(x)^+ \triangleq \max(x, 0)$, and use $\mathbb{1}_{\{\cdot\}}$ to denote the indicator function.

We now state the assumptions on the arrival processes. The throughput analysis is carried out under the following mild assumption, which has also been used in [10].

Assumption 1: For each user $i \in \{1, 2, \dots, n\}$, the arrival process $A_i(t)$ is an irreducible and positive recurrent Markov chain with countable state space, and satisfies the Strong Law of Large Numbers: That is, with probability one,

$$\lim_{t\to\infty} \frac{\sum_{\tau=0}^{t-1} A_i(\tau)}{t} = \lambda_i. \quad (1)$$

We also assume that the arrival processes are mutually independent across users (which can be relaxed for throughput analysis as discussed in [10].)

The following two assumptions are also used in the previous work [7] on the rate-function delay analysis.

Assumption 2: There exists a finite L such that $A_i(t) \le L$ for any i and t, i.e., instantaneous arrivals are bounded.

Assumption 3: The arrival processes are *i.i.d.* across users, and $\lambda_i = p$ for any user i. Given any $\epsilon > 0$ and $\delta > 0$, there exists $T > 0$, $N > 0$, and a positive function $I_B(\epsilon, \delta)$ independent of n and t such that

$$\mathbb{P}\left(\frac{\sum_{\tau=1}^t \mathbb{1}_{\{|\sum_{i=1}^n A_i(\tau) - pn| > \epsilon n\}}}{t} > \delta\right) < \exp(-ntI_B(\epsilon, \delta)),$$

for all $t > T$ and $n > N$.

Assumption 2 requires that the arrivals in each time-slot have bounded support, which is indeed true for real systems. Assumption 3 is also very general, and can be viewed as a result of the statistical multiplexing effect of a large number of sources. Assumption 3 holds for *i.i.d.* arrivals and arrivals driven by two-state Markov chains (that can be correlated over time) as two special cases.

A. Performance Objectives

In this paper, we consider two performance metrics: 1) the *throughput* and 2) the *rate-function* of the probability that the largest packet delay in the system exceeds a certain fixed threshold in the many-channel many-user asymptotic regime.

We first define the *optimal throughput region* (or *stability region*) of the system for any fixed integer $n > 0$ under Assumption 1. As in [10], a stochastic queueing network is said to be *stable* if it can be described as a discrete-time countable Markov chain and the Markov chain is stable in the following sense: The set of positive recurrent states is nonempty, and it contains a finite subset such that with probability one, this subset is reached within finite time from any initial state. When all the states communicate, stability is equivalent to the Markov chain being positive recurrent [11]. The *throughput region* of a scheduling policy is defined as the set of arrival rate vectors for which the network remains stable under this policy. Then, the *optimal throughput region* is defined as the union of the throughput regions of all possible scheduling policies, which is denoted by Λ^*. A scheduling policy is *throughput-optimal*, if it can stabilize any arrival rate vector strictly inside Λ^*. For more discussions on the optimal throughput region Λ^* in our multi-channel systems, please refer to our online technical report [9].

Next, we consider the probability that the largest packet delay in the system exceeds a certain fixed threshold, and its *rate-function* in the many-channel many-user asymptotic regime. Let $W(t) \triangleq \max_{1\le i\le n} W_i(t)$ denote the largest HOL delay over all the queues (i.e., the largest packet delay in the system) at the beginning of time-slot t. Assuming that the system is stationary and ergodic, we define rate-function $I(b)$ as the asymptotic decay-rate of the probability that the largest packet delay exceeds any fixed integer threshold $b \ge 0$, as the system size n goes to infinity, i.e.,

$$I(b) \triangleq \lim_{n\to\infty} \frac{-1}{n} \log \mathbb{P}(W(0) > b). \tag{2}$$

Note that once we know this rate-function, we can then estimate the delay-violation probability using $\mathbb{P}(W(0) > b) \approx \exp(-nI(b))$. The estimate tends to be more accurate as n becomes larger. Clearly, for systems with a large n, a larger value of the rate-function implies a better delay performance, i.e., a smaller probability that the largest packet delay in the system exceeds a certain threshold. We define the *optimal rate-function* as the maximum achievable rate-function over all possible scheduling policies, which is denoted by $I^*(b)$. A scheduling policy is *rate-function delay-optimal* if it achieves the optimal rate-function $I^*(b)$ for any fixed integer threshold $b \ge 0$.

III. An Upper Bound on The Rate-Function

In this section, we derive an upper bound on the rate-function that can be achieved by any scheduling algorithm.

Let $I_{AG}(t,x)$ denote the asymptotic decay-rate of the probability that in any interval of t time-slots, the total number of packet arrivals is greater than $n(t+x)$, as n tends to infinity, i.e.,

$$I_{AG}(t,x) \triangleq \liminf_{n\to\infty} \frac{-1}{n} \log \mathbb{P}(A(-t+1,0) > n(t+x)).$$

Let $I_{AG}(x)$ be the infimum of $I_{AG}(t,x)$ over all $t>0$, i.e.,

$$I_{AG}(x) \triangleq \inf_{t>0} I_{AG}(t,x).$$

Also, we define $I_X \triangleq \log \frac{1}{1-q}$.

Theorem 1: Given the system model described in Section II, for any scheduling algorithm, we have

$$\limsup_{n\to\infty} \frac{-1}{n} \log \mathbb{P}(W(0) > b) \\ \le \min\{(b+1)I_X, \min_{0\le c\le b}\{I_{AG}(b-c) + cI_X\}\} \triangleq I_U(b).$$

Theorem 1 can be shown by considering two events that lead to $\{W(0) > b\}$, and computing their probabilities and decay-rates. We provide the proof in our online technical report [9].

Remark: Theorem 1 implies that $I_U(b)$ is an upper bound on the rate-function that can be achieved by any scheduling policy. Hence, even for the optimal rate-function $I^*(b)$, we must have $I^*(b) \le I_U(b)$ for any fixed integer threshold $b \ge 0$.

In [7], the authors proposed the **Delay Weighted Matching (DWM)** policy that is rate-function delay-optimal and achieves upper-bound $I_U(b)$ in many cases. However, it suffers from a high complexity $O(n^5)$. Specifically, DWM requires computing a maximum-weight matching over a bipartite graph $G[V,E]$ with $|V| = O(n^2)$ and $|E| = O(n^3)$, which has a complexity $O(|V||E| + |V|^2 \log |V|) = O(n^5)$ in general [12].

IV. Delay-based Queue-Side-Greedy (D-QSG)

In this section, we develop a simple greedy scheduling policy called *Delay-based Queue-Side-Greedy (D-QSG)*. D-QSG, in an iterative manner, schedules the oldest packets in the system one-by-one whenever possible. In this sense, D-QSG can be viewed as an approximation of *First-Come First-Serve (FCFS)* policy, which has been known to be delay-optimal in many systems (e.g., a single-server queue) [7]. We will show that D-QSG not only achieves throughput optimality, but also guarantees a near-optimal rate-function, at a complexity $O(n^3)$.

A. Algorithm Description

We start by presenting some additional notations. In the D-QSG policy, there are at most n rounds in each time-slot t. Let $Q_i^k(t)$, $Z_{i,l}^k(t)$ and $W_i^k(t) = Z_{i,1}^k(t)$ denote the length of queue Q_i, the delay of the l-th packet of Q_i, and the HOL delay of Q_i after the k-th round in time-slot t, respectively. In particular, we have $Q_i^0(t) = Q_i(t)$, $Z_{i,l}^0(t) = Z_{i,l}(t)$, and $W_i^0(t) = W_i(t)$. Let $\Upsilon_k(t)$ denote the set of indices of the available servers at the beginning of the k-th round, and let $\Psi_k(t)$ denote the set of queues that have the largest HOL delay among all the queues that are connected to at least one server in $\Upsilon_k(t)$ at the beginning of the k-th round, i.e., $\Psi_k(t) \triangleq \{1 \le i \le n \mid W_i^{k-1}(t) \cdot \mathbb{1}_{\{\sum_{j\in\Upsilon_k(t)} C_{i,j}(t)>0\}} = \max_{1\le l\le n} W_l^{k-1}(t) \cdot \mathbb{1}_{\{\sum_{j\in\Upsilon_k(t)} C_{l,j}(t)>0\}}\}$. Also, let $i(k,t)$ be the index of the queue that is served in the k-th round of time-slot t, and let $j(k,t)$ be the index of the server that serves $Q_{i(k,t)}$ in that round. We then specify the operations of D-QSG as follows.

Delay-based Queue-Side-Greedy (D-QSG) policy: In each time-slot t,

1) Initialize $k = 1$ and $\Upsilon_1 = \{1, 2, \ldots, n\}$.
2) In the k-th round, allocate server $S_{j(k,t)}$ to $Q_{i(k,t)}$, where

$$i(k,t) = \min\{i \mid i \in \Psi_k(t)\},$$
$$j(k,t) = \min\{j \in \Upsilon_k(t) \mid C_{i(k,t),j}(t) = 1\}.$$

That is, in the k-th round, we consider the queues that have the largest HOL delay among those that have at least one available server connected (i.e., the queues in set $\Psi_k(t)$), and break ties by picking the queue with the smallest index (i.e., $Q_{i(k,t)}$). We then choose an available server that are connected to queue $Q_{i(k,t)}$, and break ties by picking the server with the smallest index (i.e., server $S_{j(k,t)}$), to serve $Q_{i(k,t)}$. At the end of the k-th round, update the length of $Q_{i(k,t)}$ to account for service, i.e., set $Q_{i(k,t)}^k(t) =$

$\left(Q^{k-1}_{i(k,t)}(t) - C_{i(k,t),j(k,t)}(t)\right)^+$ and $Q^k_i(t) = Q^{k-1}_i(t)$ for all $i \neq i(k,t)$. Also, update the HOL delay of $Q_{i(k,t)}$, by setting $W^k_{i(k,t)}(t) = Z^k_{i(k,t),1}(t) = Z^{k-1}_{i(k,t),2}(t)$ if $Q^k_{i(k,t)}(t) > 0$, and $W^k_{i(k,t)}(t) = 0$ otherwise, and setting $W^k_i(t) = W^{k-1}_i(t)$ for all $i \neq i(k,t)$.

3) Stop if k equals n. Otherwise, increase k by 1, set $\Upsilon_k(t) = \Upsilon_{k-1}(t) \backslash \{j(k,t)\}$, and repeat step 2.

Remark: D-QSG has a complexity $O(n^3)$, since there are at most n rounds, and in each round, it takes $O(n^2 + n) = O(n^2)$ time to find a queue that has at least one connected and available server (which takes $O(n^2)$ time to check for all queues) and that has the largest HOL delay (which takes $O(n)$ time to compare). It should be noted that in each round, when there are multiple queues that have the largest HOL delay, D-QSG chooses the queue with the smallest index; when there are multiple available servers that are connected to the chosen queue, D-QSG allocates the server with the smallest index. We specify such a tie-breaking rule for ease of analysis. In practice, we can also break ties arbitrarily.

B. Near-optimal Delay Performance

In this section, we present the main result of this paper on near-optimal rate-function. We first define near-optimal rate-function, and then evaluate the delay performance of D-QSG.

A policy $\mathbf{P}$ is said to achieve *near-optimal rate-function* if the delay rate-function $I(b)$ attained by policy $\mathbf{P}$ for any fixed integer threshold $b > 0$, is no smaller than $I^*(b-1)$, the optimal rate-function for threshold $b-1$. That is,

$$I(b) = \liminf_{n\to\infty} \frac{-1}{n} \log \mathbb{P}\left(W(0) > b\right) \geq I^*(b-1). \quad (3)$$

We next present our main result in the following theorem, which states that D-QSG achieves a near-optimal rate-function.

Theorem 2: Under Assumptions 2 and 3, D-QSG achieves a near-optimal rate-function, as given in (3).

We prove Theorem 2 by the following strategy: 1) motivated by a key property of D-QSG (Lemma 1), we propose the *Greedy Frame-Based Scheduling (G-FBS)* policy, which is a variant of the FBS policy in [7] that has been shown to be rate-function delay-optimal in many cases; 2) show that G-FBS achieves a near-optimal rate-function (Theorem 3); 3) prove a dominance property of D-QSG over G-FBS. Specifically, in Lemma 2, we show that for any given sample path, by the end of each time-slot, D-QSG has served every packet that G-FBS has served. *Note that Theorem 2 holds for D-QSG with any tie-breaking rules, under which, when allocating a server to a queue, it does not account for the connectivity between this server and the other queues.* The performance of D-QSG may be further improved, if a better tie-breaking rule is applied.

We now present a crucial property of D-QSG in Lemma 1, which is the key to proving the rate-function near-optimality for G-FBS and D-QSG.

Lemma 1: Consider any n packets and any strictly increasing function $f(n) < \frac{n}{2}$. Suppose that D-QSG is applied to schedule these n packets. Then, there exists a finite integer $N_X > 0$ such that for all $n \geq N_X$, with probability no smaller than $1 - 2(1-q)^{n-2f(n)}$, D-QSG schedules at least $n - 2\sqrt{n}$ packets, including the oldest $f(n)$ packets.

We provide the proof of Lemma 1 in our online technical report [9], and explain the importance of Lemma 1 as follows. We first recall how DWM is shown to be rate-function delay-optimal in [7]. Specifically, the authors of [7] compare DWM with another policy FBS. In FBS, packets are filled into frames with size $n - H$ in a FCFS manner, where H is a suitably chosen constant independent of n. The FBS policy attempts to serve the entire HOL frame whenever possible. The authors of [7] first establish the rate-function optimality of the FBS policy. Then, by showing that DWM dominates FBS (i.e., DWM will serve the same packets in the entire HOL frame whenever possible), the delay optimality of DWM then follows.

However, this comparison approach will not work directly for D-QSG. In order to serve all packets in a frame whenever possible, one would need certain back-tracking (or rematching) operations as in a typical maximum-weight matching algorithm like DWM. For a simple greedy algorithm like D-QSG that does not do back-tracking, it is unlikely to attain the same probability of serving the entire frame. In fact, even if we reduce the maximum frame size to $n - 2\sqrt{n}$, we are still unable to show that D-QSG can serve the entire frame with a sufficiently high probability. Thus, we cannot compare D-QSG with FBS as in [7].

Fortunately, Lemma 1 provides an alternate avenue. Specifically, for a frame of size n, even though D-QSG may not serve any *given* subset of $n - 2\sqrt{n}$ packets with a sufficiently high probability, it will serve *some* subset of $n - 2\sqrt{n}$ packets with a sufficiently high probability. Further, this subset must contain the oldest $2\sqrt{n}$ packets for a large n, if we choose $f(n)$ in Lemma 1 such that $f(n) \in \omega(\sqrt{n})$. Note that D-QSG still leaves (at most) $2\sqrt{n}$ packets to the next time-slot. In the next time-slot, if we can make sure that D-QSG serves all of these $2\sqrt{n}$ leftover packets, which also happen to be the oldest, we would then at worst suffer an additional one-time-slot delay. Intuitively, we would then be able to show that D-QSG attains a near-optimal delay rate-function.

To make this argument rigorous, we next compare D-QSG with a new policy called **Greedy Frame-Based Scheduling (G-FBS)**. Note that G-FBS is only for assisting our analysis, and will not be used as an actual scheduling algorithm. In the G-FBS policy, packets are grouped into frames. Each frame has a capacity of $n_0 = n - 2\sqrt{n}$ packets, i.e., at most n_0 packets can be filled into a frame. As packets arrive to the system in each time-slot, the frames are created by filling the packets sequentially. Specifically, packets that arrive earlier are filled into the frame with a higher priority, and packets from queues with a smaller index are filled with a higher priority when multiple packets arrive in the same time-slot. Once the current frame is fully filled, it will be closed and a new frame will be open. We also assume that there is a "leftover" frame, called *L-frame* for simplicity, with a capacity of $2\sqrt{n}$ packets. The L-frame is for storing the packets that are not served in the previous time-slot and are carried over to the current time-slot. At the beginning of each time-slot, we combine the HOL frame and the L-frame into a "super" frame, called *S-frame* for simplicity, with a capacity of n packets. If there are less than n packets in the S-frame, we can artificially add some dummy packets with a delay of zero at the end of the S-frame so that the S-frame is fully filled. In each time-slot, G-FBS runs the D-QSG policy, but restricted to only the n packets of the S-frame. We call it a *success*, if D-QSG can schedule at least n_0 packets, including the oldest $f(n)$ packets, from the S-frame, where $f(n) < \frac{n}{2}$ is any function that satisfies that $f(n) \in o(n)$ and $f(n) \in \omega(\sqrt{n})$. In each time-slot, if a success does not occur, then no packets will be served. When there is a success, the G-FBS policy serves all the packets that are scheduled by D-QSG restricted to the S-frame in that time-slot. *Lemma 1 implies that in each time-slot, a success occurs with probability at least* $1-2(1-q)^{n-2f(n)}$. When there is a success, all packets from the S-frame, except for at most $2\sqrt{n} = n - n_0$ packets, are successfully served, and these served packets include the oldest $f(n)$ packets. The packets that are not served will be stored in the L-frame, and carried over to the next time-slot (except for the dummy packets, which will be discarded.)

Remark: Although G-FBS is similar to FBS policy [7], it exhibits a key difference from FBS. In the FBS policy, in each time-slot, either an entire frame (i.e., all the packets in the frame) will be completely served or none of its packets will be served. Hence, it does not allow packets to be carried over to the next frame. In contrast, G-FBS allows leftover packets and is thus more flexible in serving frames. This property is the key reason that we can use a lower-complexity policy (like D-QSG). On the other hand, it leads to a small gap between the rate-functions achieved by G-FBS and delay-optimal policies (e.g., FBS and DWM). Nonetheless, this gap can be well characterized by using Lemma 1. Specifically, in the G-FBS policy, an L-frame contains at most $2\sqrt{n}$ packets, because at most $2\sqrt{n}$ packets are not served whenever there is a success. Further, these (at most) $2\sqrt{n}$ leftover packets will be among the oldest $f(n)$ packets (in the S-frame) in the next time-slot when n is large, due to our choice of $f(n) \in \omega(\sqrt{n})$. Hence, another success will serve all the leftover packets. This implies that at most $x+1$ successes are needed to completely serve x frames, for any finite integer $x > 0$. *In fact, this property is the key reason for a one-time-slot shift in the guaranteed rate-function by G-FBS, which leads to the near-optimal delay performance*, as we show in the following theorem.

Theorem 3: Under Assumptions 2 and 3, G-FBS policy achieves a near-optimal rate-function, as given in (3).

The proof of Theorem 3 follows a similar line of argument as in the proof for rate-function delay optimality of FBS (Theorem 2 in [7]). We consider all the events that lead to the delay-violation event $\{W(0) > b\}$, which can be caused by two factors: bursty arrivals and sluggish service. On the one hand, if there are a large number of arrivals in certain period, say of length t time-slots, which exceeds the maximum number of packets that can be served in a period of $t+b+1$ time-slots, then it unavoidably leads to a delay violation. On the other hand, suppose that there is at least one packet arrival at certain time, and that under G-FBS, a success does not occur in any of the following $b+1$ time-slots (including the time-slot when the packet arrives), then it also leads to a delay violation. Each of these two possibilities has a corresponding rate-function for its probability of occurring. Large-deviations theory then tells us that the rate-function for delay violation is determined by the smallest rate-function among these possibilities (i.e., "rare events occur in the most likely way".) We can then show that $I(b) \geq I_U(b-1) \geq I^*(b-1)$ for any integer $b > 0$, where $I(\cdot)$ is the rate-function attained by G-FBS, $I_U(\cdot)$ is the upper bound that we derived in Section III, and $I^*(\cdot)$ is the optimal rate-function, respectively. We provide the detailed proof of Theorem 3 in our online technical report [9].

Remark: Note that the gap between the optimal rate-function and the above near-optimal rate-function is likely to be quite small. For example, in the special case of *i.i.d.* 0-1 arrivals, the near-optimal rate-function implies that $I(b) \geq \frac{b}{b+1} I_U(b) \geq \frac{b}{b+1} I^*(b)$, since we can compute that $I_U(b) = (b+1)\log\frac{1}{1-q}$ for this special case.

Finally, we make use of the following dominance property of D-QSG over G-FBS.

Lemma 2: For any given sample path, by the end of any time-slot, D-QSG has served every packet that G-FBS has served.

We prove Lemma 2 by induction, and provide the proof in our online technical report [9]. Then, the near-optimal rate-function of D-QSG (Theorem 2) follows immediately from Lemma 2 and Theorem 3.

C. Throughput Optimality

In this section, we establish throughput optimality of D-QSG. Note that the rate-function is studied in the asymptotic regime, i.e., when n goes to infinity. Hence, even if the convergence rate of the rate-function is fast (as is typically the case), the throughput performance may be poor for small to moderate values of n. As a matter of fact for a fixed n, a rate-function delay-optimal policy (e.g., FBS) may not even be throughput-optimal. To this end, we are also interested in the throughput performance of scheduling policies in general non-asymptotic regimes (i.e., in a multi-channel system with any fixed value of n.)

It is well-known that the MaxWeight Scheduling policy [10], [13]–[15] that maximizes the weighted sum of the rates (where the weight is either queue length or delay) is throughput-optimal in very general settings, including the multi-channel system that we consider in this paper. Hence, we first discuss a simple extension of the Delay-based MaxWeight Scheduling (D-MWS) policy [8], [10], [14], [15] for our multi-channel system.

Let $\mathcal{S}_j(t)$ denote the set of queues that are connected to server S_j in time-slot t, i.e., $\mathcal{S}_j(t) = \{1 \leq i \leq n \mid C_{i,j}(t) = 1\}$, and let $\Gamma_j(t)$ denote the subset of queues in $\mathcal{S}_j(t)$ that have the largest HOL delay in time-slot t, i.e., $\Gamma_j(t) \triangleq \{i \in \mathcal{S}_j(t) \mid W_i(t) = \max_{l \in \mathcal{S}_j(t)} W_l(t)\}$. We then specify the operations of D-MWS as follows.

Delay-based MaxWeight Scheduling (D-MWS) policy: In each time-slot t, the scheduler assigns server S_j to serve queue $Q_{i(j,t)}$ such that $i(j,t) = \min\{i \mid i \in \Gamma_j(t)\}$. That is, each server is selected to serve a connected queue that has the largest HOL delay, breaking ties by picking the queue with the smallest index when there are multiple such queues.

Remarks: We can prove throughput-optimality of D-MWS in our multi-channel system, using fluid limit techniques by following the same line of analysis used in [10] for a single-channel system. The key insight we obtain from the proof in [10] is that to achieve throughput optimality, it is sufficient for each server to serve a connected queue that has the largest weight in the fluid limits rather than in the original system.

Using the insight obtained above, we next show that D-QSG is throughput-optimal in general non-asymptotic settings (for a system with any fixed n).

Theorem 4: D-QSG policy is throughput-optimal under Assumption 1.

We prove Theorem 4 using the fluid limit techniques [10], [16]. Different from D-MWS policy under which, each server chooses to serve a connected queue with the largest HOL delay, D-QSG allocates servers to serve the oldest packets first one-by-one in an iterative manner. Hence, we can show that the operations of D-QSG guarantees that each server chooses a connected queue that has a large enough weight, and that in the fluid limits the weight of the queue chosen by each server is equal to that of the queue chosen under D-MWS. Then, we complete the proof of Theorem 4, following a similar line of analysis as in [10]. We provide the detailed proof in our online technical report [9].

So far, we have shown that D-QSG not only achieves a near-optimal rate-function, but also guarantees throughput optimality, with a lower complexity $O(n^3)$ than that of DWM. Interestingly, we will show next that just by switching the order of examining the servers or the queues first, we can obtain another policy that not only achieves the same performance of throughput optimality and rate-function near-optimality as that of D-QSG, but also incurs an even lower complexity $O(n^2)$.

V. Delay-based Server-Side-Greedy (D-SSG)

In this section, we develop another greedy scheduling policy called *Delay-based Server-Side-Greedy (D-SSG)*, under which each server iteratively chooses to serve a connected queue that has the largest HOL delay. We show that D-SSG is equivalent to D-QSG under certain tie-breaking rules, in the sample-path sense, and thus achieves the same performance of *throughput optimality and rate-function near-optimality* as that of D-QSG. Further, *D-SSG has an even lower complexity* $O(n^2)$.

Before we describe the detailed operations of D-SSG, we would like remark on D-MWS due to the similarity between D-MWS and D-SSG. Note that D-MWS is not only throughput-optimal, but also has a low complexity $O(n^2)$. However, we can show that D-MWS suffers from poor delay performance. Specifically, following a similar line of argument as in the proof of Theorem 3 in [3], we can show that D-MWS yields a zero rate-function in certain scenarios (e.g., with *i.i.d.* 0-1 arrivals). We omit the proof here, and explain the intuition behind it as follows. Under D-MWS, each server chooses to serve a connected queue that has the largest HOL delay without accounting for the decisions of the other servers. This way of allocating servers leads to an unbalanced schedule. That is, only a small fraction of the queues get served in each time-slot. This inefficiency essentially leads to poor delay performance.

Now, we describe the operations of our proposed D-SSG policy. D-SSG is similar to D-MWS, in the sense that it also allocates each server to serve a connected queue that has the largest HOL delay. However, there is a key difference. That is, instead of allocating the servers all at once as in D-MWS, D-SSG allocates the servers one-by-one, accounting for the scheduling decisions of the servers that are allocated earlier. We will show that *this critical difference results in a substantial improvement in the delay performance.*

We present some additional notations, and then specify the detailed operations of D-SSG. In each time-slot, there are n rounds, and in each round, one of the remaining servers is allocated. Let $Q_i^k(t)$, $Z_{i,l}^k(t)$ and $W_i^k(t) = Z_{i,1}^k(t)$ denote the length of queue Q_i, the delay of the l-th packet of Q_i, and the HOL delay of Q_i after $k \geq 1$ rounds of server allocation in time-slot t, respectively. In particular, we have $Q_i^0(t) = Q_i(t)$, $Z_{i,l}^0(t) = Z_{i,l}(t)$, and $W_i^0(t) = W_i(t)$. Recall that $\mathcal{S}_j(t) = \{1 \leq i \leq n \mid C_{i,j}(t) = 1\}$. Let $\Gamma_j^k(t)$ denote the set of indices of the queues that are connected to server S_j in time-slot t and have the largest HOL delay at the beginning of the k-th round in time-slot t, i.e., $\Gamma_j^k(t) \triangleq \{i \in \mathcal{S}_j(t) \mid W_i^{k-1}(t) = \max_{l \in \mathcal{S}_j(t)} W_l^{k-1}(t)\}$. Let $i(j,t)$ denote the index of queue that is served by server S_j in time-slot t under D-SSG.

Delay-based Server-Side-Greedy (D-SSG) policy: In each time-slot t,

1) Initialize $k = 1$.
2) In the k-th round, allocate server S_k to serve queue $Q_{i(k,t)}$, where $i(k,t) = \min\{i \mid i \in \Gamma_k^k(t)\}$. That is, in the k-th round, server S_k is allocated to serve the connected queue that has the largest HOL delay, breaking ties by picking the queue with the smallest index if there are multiple such queues. Then, update the length of $Q_{i(k,t)}$ to account for service, i.e., set $Q_{i(k,t)}^k(t) = \left(Q_{i(k,t)}^{k-1}(t) - C_{i(k,t),k}(t)\right)^+$ and $Q_i^k(t) = Q_i^{k-1}(t)$ for all $i \neq i(k,t)$. Also, update the HOL delay of $Q_{i(k,t)}$ to account for service, i.e., set $W_{i(k,t)}^k(t) = Z_{i(k,t),1}^k(t) = Z_{i(k,t),2}^{k-1}(t)$ if $Q_{i(k,t)}^k(t) > 0$, and $W_{i(k,t)}^k(t) = 0$ otherwise, and set $W_i^k(t) = W_i^{k-1}(t)$ for all $i \neq i(k,t)$.
3) Stop if k equals n. Otherwise, increase k by 1 and repeat step 2.

Remark: Note that both D-SSG and D-QSG aim to allocate each server to a queue with the largest HOL delay. The key difference between D-SSG and D-QSG is that D-SSG iterates over the servers first while D-QSG iterates over the packets/queues first. This key difference leads to the fact that D-SSG is simpler to implement and has an even *lower* *complexity* $O(n^2)$. Specifically, there are n rounds, and in each round, it takes at most n times for a server to find a connected queue with the largest HOL delay.

It should be noted that the queue-length-based counterpart of D-SSG, called Q-SSG, has been studied in [3], [4]. Under Q-SSG, each server iteratively chooses to serve a connected queue that has the largest length. It has been shown that Q-SSG not only achieves throughput optimality, but also guarantees a *positive (queue-length)* rate-function. However, their results have the following limitations: 1) a positive rate-function may not be good enough, since the gap between the guaranteed rate-function and the optimal is unclear; 2) good queue-length performance does not necessarily translate into good delay performance; 3) their analysis was only carried out for restricted arrival processes that are *not only i.i.d. across users, but also in time*. In contrast, in the following theorem, we show that D-SSG achieves a rate-function that is *not only positive but also near-optimal* (in the sense of (3)) for more general arrival processes, while guaranteeing throughput optimality.

Theorem 5: D-SSG policy is throughput-optimal under Assumption 1, and achieves a near-optimal rate-function as given in (3) under Assumptions 2 and 3.

Theorem 5 follows immediately from the following lemma, which states that D-SSG is equivalent to D-QSG under the tie-breaking rules specified in this paper.

Lemma 3: For the same sample path, i.e., same realizations of arrivals and channel connectivity, D-QSG and D-SSG pick the same schedule in every time-slot.

We prove Lemma 3 by induction, and provide the proof in our online technical report [9]. Note that under D-SSG, in each round, when a server has multiple connected queues that have the largest HOL delay, we break ties by picking the queue with the smallest index. Presumably, one can take other arbitrary tie-breaking rules. However, it turns out that directly analyzing the rate-function for a greedy policy from the server side (like D-SSG) is much more difficult than that for a greedy policy from the queue side (like D-QSG). For example, as we mentioned earlier, the authors of [3], [4] were only able to prove a positive (queue-length) rate-function for Q-SSG in more restricted scenarios. Hence, our choice of the above simple tie-breaking rule is in fact quite important to leading to the equivalence property in Lemma 3, which plays a key role in proving the rate-function near-optimality for D-SSG. Nevertheless, we would expect that one can choose arbitrary tie-breaking rules in practice.

So far, we have shown that our proposed low-complexity greedy policies achieve both throughput optimality and rate-function near-optimality. In the next section, we will show through simulations that these greedy policies *not only exhibit a near-optimal rate-function, but also empirically are virtually indistinguishable from the delay-optimal policy DWM in many scenarios.*

VI. Simulation Results

In this section, we conduct simulations to compare scheduling performance of our proposed greedy policies with DWM, D-MWS, and Q-SSG. We simulate these policies in Java and compare the empirical probabilities that the largest HOL delay in the system in any given time-slot exceeds an integer threshold b, i.e., $\mathbb{P}(W(0) > b)$.

For the arrival processes, we consider bursty arrivals that are driven by a two-state Markov chain and that are correlated over time. (We obtained similar results for *i.i.d.* arrivals, and do not report them here due to space constraints.) We adopt the same parameter settings as in [7]. For each user, there are 5 packet-arrivals when the Markov chain is in state 1, and there is no arrivals when it is in state 2. The transition probability of the Markov chain is given by the matrix $[0.5, 0.5; 0.1, 0.9]$, and the state transitions occur at the end of each time-slot. The arrivals for each user are correlated over time, but they are independent across users. For the channel model, we first assume *i.i.d.* ON-OFF channels with unit capacity, and set $q = 0.75$. We later consider more general scenarios with heterogeneous users and bursty channels that are correlated over time. We run simulations for a system with n servers and n users, where $n \in \{10, 20, \ldots, 100\}$. The simulation period lasts for 10^7 time-slots for each policy and each system.

The results are summarized in Fig. 2, where the complexity of each policy is also labeled. In order to compare the rate-function $I(b)$ as defined in Eq. (2), we plot the probability over the number of channels or users, i.e., n, for a fixed value of threshold b. The negative of the slopes of the curves can be viewed as the rate-function for each policy. In Fig. 2, we report the results only for $b = 4$, and the results are similar for other values of threshold b. From Fig. 2, we observe that both D-QSG and D-SSG are virtually indistinguishable from DWM, which is known to be rate-function delay-optimal. This not only supports our theoretical results that both D-QSG and D-SSG guarantee a near-optimal rate-function, but also implies that both D-QSG and D-SSG empirically perform very well while enjoying a lower complexity. Further, we observe that D-SSG consistently outperforms its queue-length-based counterpart, Q-SSG, despite that in [3], it has been shown through simulations that Q-SSG *empirically* achieves near-optimal queue-length performance. This provides a further evidence that good queue-length performance does not necessarily translate into good delay performance. The results also show that D-MWS yields a zero rate-function, as expected.

Further, we evaluate scheduling performance of different policies in more realistic scenarios, where users are *heterogeneous* and channels are *correlated over time*. Specifically, we consider channels that can be modeled as a two-state Markov chain, where the channel is "ON" when the Markov chain is in state 1, and is "OFF" when it is in state 2. This type of channel model can be viewed as a special case of the Gilbert Elliot model that is widely used for describing bursty channels. We assume that there are two classes of users: users with an odd index are called *near-users*, and users with an even index are called *far-users*. Different classes of users see different channel conditions: near-users see better channel condition, and far-users see worse channel condition. We assume that the transition probability matrices of channels

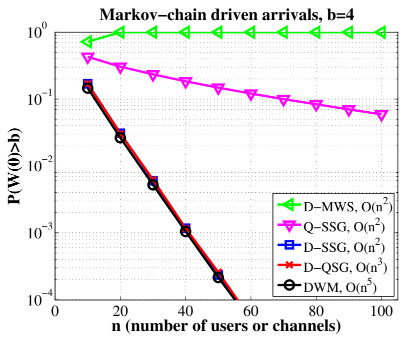

Fig. 2. Performance comparison of different scheduling policies in the case with homogeneous *i.i.d.* channels, for $b = 4$.

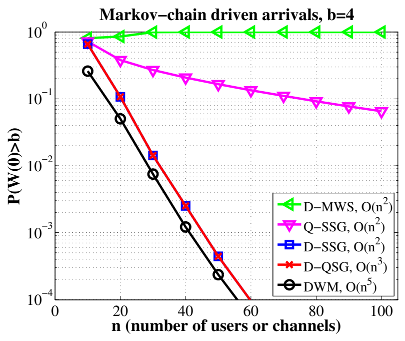

Fig. 3. Performance comparison of different scheduling policies in the case with heterogeneous users and Markov-chain driven channels, for $b = 4$.

for near-users and far-users are $[0.833, 0.167; 0.5, 0.5]$ and $[0.5, 0.5; 0.167, 0.833]$, respectively. The arrival processes are assumed to be the same as in the previous case.

The results are summarized in Fig. 3. We observe similar results as in the previous case with homogeneous users and *i.i.d.* channels in time. In particular, D-QSG and D-SSG exhibit a rate-function that is the same as that of DWM, although their delay performance is slightly worse. Note that in this scenario, a rate-function delay-optimal policy is *not* known yet. Hence, for future work, it would be interesting to understand how to design rate-function delay-optimal or near-optimal policies in general scenarios.

VII. Conclusion

In this paper, we developed low-complexity greedy scheduling policies that not only achieve throughput optimality, but also guarantee a near-optimal delay rate-function, for multi-channel wireless networks. Our studies reveal that throughput optimality is relatively easier to achieve in such multi-channel systems, while there exists an explicit trade-off between complexity and delay performance. If one can bear a minimal drop in the delay performance, lower-complexity scheduling policies can be exploited.

For future work, it would be interesting to explore whether one can find low-complexity scheduling policies that can guarantee both throughput and delay optimality. Further, it is still unclear how to design scheduling policies (even with a high complexity) that can guarantee optimal or near-optimal delay performance in more realistic scenarios. Therefore, it is important to investigate the scheduling problem in such multi-channel systems with more general models, e.g., accounting for multi-rate channels that are correlated over time, instead of *i.i.d.* ON-OFF channels, as well as heterogeneous users and channels with different statistics.

Acknowledgment

The authors would like to thank Manu Sharma at Qualcomm Inc. for his valuable comments on this work.

References

[1] S. Kittipiyakul and T. Javidi, “Delay-optimal server allocation in multiqueue multiserver systems with time-varying connectivities,” *IEEE Transactions on Information Theory*, vol. 55, no. 5, pp. 2319–2333, 2009.

[2] S. Bodas, S. Shakkottai, L. Ying, and R. Srikant, “Scheduling in multi-channel wireless networks: Rate function optimality in the small-buffer regime,” in *ACM Proceedings of the eleventh international joint conference on Measurement and modeling of computer systems (SIGMETRICS)*, 2009, pp. 121–132.

[3] ——, “Low-complexity scheduling algorithms for multi-channel downlink wireless networks,” in *The IEEE International Conference on Computer Communications (INFOCOM)*. IEEE, 2010, pp. 1–9.

[4] ——, “Scheduling for small delay in multi-rate multi-channel wireless networks,” in *The IEEE International Conference on Computer Communications (INFOCOM)*. IEEE, 2011, pp. 1251–1259.

[5] S. Bodas and T. Javidi, “Scheduling for multi-channel wireless networks: Small delay with polynomial complexity,” in *2011 International Symposium on Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks (WiOpt)*. IEEE, 2011, pp. 78–85.

[6] V. Venkataramanan and X. Lin, “On wireless scheduling algorithms for minimizing the queue-overflow probability,” *IEEE/ACM Transactions on Networking*, vol. 18, no. 3, pp. 788–801, 2010.

[7] M. Sharma and X. Lin, “OFDM downlink scheduling for delay-optimality: Many-channel many-source asymptotics with general arrival processes,” in *The IEEE Information Theory and Applications Workshop (ITA)*, 2011.

[8] B. Ji, C. Joo, and N. B. Shroff, “Delay-based Back-Pressure Scheduling in Multi-Hop Wireless Networks,” in *The IEEE International Conference on Computer Communications (INFOCOM)*, 2011, pp. 2579–2587.

[9] B. Ji, G. R. Gagan, X. Lin, and N. B. Shroff, “Performance of Low-Complexity Greedy Scheduling Policies in Multi-Channel Wireless Networks: Optimal Throughput and Near-Optimal Delay,” *Arxiv preprint arXiv:1212.1638*, December 2012. [Online]. Available: http://arxiv.org/abs/1212.1638

[10] M. Andrews, K. Kumaran, K. Ramanan, A. Stolyar, R. Vijayakumar, and P. Whiting, “Scheduling in a queuing system with asynchronously varying service rates,” *Probability in the Engineering and Informational Sciences*, vol. 18, pp. 191–217, 2004.

[11] M. Bramson, “Stability of queueing networks,” *Probability Surveys*, vol. 5, no. 1, pp. 169–345, 2008.

[12] M. Fredman and R. Tarjan, “Fibonacci heaps and their uses in improved network optimization algorithms,” *Journal of the ACM (JACM)*, vol. 34, no. 3, pp. 596–615, 1987.

[13] L. Tassiulas and A. Ephremides, “Stability properties of constrained queueing systems and scheduling policies for maximum throughput in multihop radio networks,” *IEEE Transactions on Automatic Control*, vol. 37, no. 12, pp. 1936–1948, 1992.

[14] A. Eryilmaz, R. Srikant, and J. Perkins, “Stable scheduling policies for fading wireless channels,” *IEEE/ACM Transactions on Networking*, vol. 13, no. 2, pp. 411–424, 2005.

[15] B. Sadiq and G. de Veciana, “Throughput optimality of delay-driven MaxWeight scheduler for a wireless system with flow dynamics,” in *Proceedings of the 47th Annual Conference on Communication, Control and Computing (Allerton)*, 2009.

[16] J. Dai, “On positive Harris recurrence of multiclass queueing networks: a unified approach via fluid limit models,” *The Annals of Applied Probability*, pp. 49–77, 1995.