Low-complexity Scheduling Algorithms for Multi-channel Downlink Wireless Networks

Shreeshankar Bodas, Sanjay Shakkottai
Department of ECE
The University of Texas at Austin
{shreeshankar,shakkott}@mail.utexas.edu

Lei Ying
Department of ECE
Iowa State University
leiying@iastate.edu

R. Srikant
Department of ECE
Univ. of Illinois at Urbana-Champaign
rsrikant@illinois.edu

***Abstract*—This paper considers the problem of designing scheduling algorithms for multi-channel (e.g., OFDM) wireless downlink networks with n users/OFDM sub-channels. For this system, while the classical MaxWeight algorithm is known to be throughput-optimal, its buffer-overflow performance is very poor (formally, we show it has zero rate function in our setting). To address this, we propose a class of algorithms called iHLQF (iterated Heaviest matching with Longest Queues First) that is shown to be throughput optimal for a general class of arrival/channel processes, and also rate-function optimal (i.e., exponentially small buffer overflow probability) for certain arrival/channel processes. iHLQF however has higher complexity than MaxWeight (n^4 vs. n^2 respectively). To overcome this issue, we propose a new algorithm called SSG (Server-Side Greedy). We show that SSG is throughput optimal, results in a much better per-user buffer overflow performance than the MaxWeight algorithm (positive rate function for certain arrival/channel processes), and has a computational complexity (n^2) that is comparable to the MaxWeight algorithm. Thus, it provides a nice trade-off between buffer-overflow performance and computational complexity. These results are validated by both analysis and simulations.**

***Index Terms*—Scheduling algorithms, large deviations, small buffer, low complexity**

I. Introduction

The emergence of 4G OFDM (Orthogonal Frequency Division Multiplexing) based wireless systems (e.g., WiMax [6] and LTE [1]) has led to an increasing interest in the design of scheduling algorithms for the downlink of wireless networks. In these systems, the downlink bandwidth is partitioned into several (tens to hundreds of) parallel channels. Scheduling decisions are made every timeslot (2 - 5 msec), where in each timeslot, each of the channels is allocated to a potentially different user. The data-rate that a channel can support is timeslot-, user- and channel-dependent. This scenario translates into a multi-user multi-server system, where a given server (channel) can serve only one user per timeslot.

From a networking perspective, the goal is to allocate resources in this multi-channel system in a way that satisfies certain objectives. The fundamental objective is to design scheduling algorithms that provide the maximum network throughput. It is well-known that the MaxWeight algorithm ([16], also see Definition 2) is throughput-optimal for this system. The MaxWeight algorithm has received considerable attention from the researchers, and has been analyzed in a variety of scenarios such as the large queues ([19], [15], [18]) or the heavy-load ([14], [12], [9]) regime.

However, performance (e.g., small per-user queues) is equally important in order to ensure small user-perceived delays. In this paper, we develop low complexity and throughput-optimal algorithms that also provide good performance. A key outcome of our study is that the algorithm design insights from a performance viewpoint are different from the design insights that follow from a throughput-optimality viewpoint, as discussed in the following section.

A. Main Contributions

To put this section in context, we briefly describe the system model. For more details, please see Section II. We consider a multi-queue multi-server queuing system as shown in Figure 1. There are external packet arrivals to the queues, and the channels connecting the queues to the servers support time-varying data-rates. As in an OFDM system, a given server (frequency band) can be allocated to serve only one queue in a given timeslot. The goal is to design a scheduling rule for allocating the servers to the queues that, in addition to throughput-optimality, also guarantees small per-user queues. We look at this "small-queues" or "small-buffer at the base-station" problem from a large deviations perspective, where the number of users in the system and the available bandwidth is large. In [3], we have shown that a class of algorithms called iLQF (iterated Longest Queues First), under certain technical conditions, is rate function optimal for the small buffer overflow event. However, the results regarding iLQF were derived assuming symmetric, i.i.d., ON-OFF traffic and i.i.d. ON-OFF channels. In particular, for more general arrival and channel processes, a number of fundamental questions were left unanswered, such as:

1) Are the algorithms in the iLQF class throughput-optimal for the system?
2) The well-known MaxWeight algorithm [16] is throughput-optimal for the system. What is its performance for the small buffer overflow problem?
3) The iLQF-class algorithms typically have a higher computational complexity than the MaxWeight algorithm. Can we design an algorithm with lower complexity without compromising either throughput-optimality or small-queue performance?

In this paper, we show that the answers to these questions are yes, poor, and yes respectively. The following is a summary of our main contributions in this paper:

- We show that a generalization of the iLQF-class rules, namely iHLQF, is throughput-optimal for very general arrival and channel process models (Section III).
- We show that the classical MaxWeight rule is very poor at keeping the per-user queues small (Section IV). Formally, we show that this rule results in a zero rate function (to be defined) for the small buffer overflow event defined in Section II.
- We propose a new scheduling algorithm called the Server-Side Greedy (SSG) service rule which is an iterative version of the MaxWeight rule, where the queue-lengths are updated after each server (OFDM sub-channel) finishes its service. We show that this rule is throughput-optimal under general arrival and channel processes, results in a strictly positive rate function for the small buffer overflow event (implying small per-user queues), and has complexity comparable to that of the MaxWeight rule, and much less than the iLQF-class rules (Section V).

An important design insight that emerges from the above results is the following: for throughput-optimality, the MaxWeight algorithm argues that the scheduling algorithm should maximize the sum of channel-rate-weighted queue-lengths in each timeslot. However, from a small-queue performance viewpoint, our results indicate that as long as we are "close" to the maximum weighted sum, the scheduling algorithm's objective should shift to equalizing the queues, and that this allocation of channel resources should proceed in an iterative manner.

B. Related Work

Scheduling for multi-user wireless networks is a well-investigated problem [17], [2], [13], [11]. Researchers have analyzed this problem from a variety of angles: heavy traffic limits [14], [12], [9], tail probabilities of queue-lengths [19], [15], [18], and energy-delay tradeoffs for wireless downlink [10]. These results provide very useful guidelines for designing scheduling algorithms, but strictly speaking, a majority of the earlier results are valid only in the large queues regime, i.e., as the queue-lengths tend to infinity. In a recent paper [7], a model similar to our model was considered and optimality results were derived for the two-user case. To the best of our knowledge, the first paper to consider the small buffer overflow problem in a large deviations setting was [3], where we considered a restricted version of the model in this paper and derived rate function optimality results in the many-user, small-queues regime.

II. System Model

We consider a multi-queue, multi-server, discrete-time queuing system as shown in Figure 1. The system has n queues and n servers, connected by time-varying channels that can potentially change from timeslot to timeslot. (We note that for our proof techniques to work and results to hold, it is not necessary that the number of queues and servers be the same, and a constant-factor relation between the two works just as fine.)

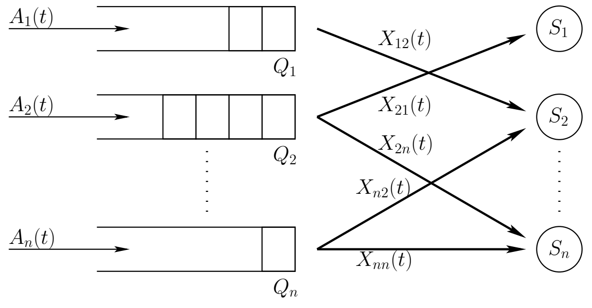

Fig. 1. System Model

The following notation is used throughout this paper.

Q_i	=	The entity, queue number i
S_i	=	The entity, server number i
$\mathcal{Q}$	=	$\{Q_1, Q_2, \ldots, Q_n\}$
$\mathcal{S}$	=	$\{S_1, S_2, \ldots, S_n\}$
$A_i(t)$	=	The number of packet arrivals to Q_i at the beginning of timeslot t
$X_{ij}(t)$	=	The number of packets in Q_i that can potentially be served by S_j, in timeslot t
$Q_i(t)$	=	The length of Q_i at the end of timeslot t
$Q_i^{(k)}(t)$	=	The length of Q_i after $k \geq 1$ rounds of service in timeslot t
$Q_i^{(0)}(t)$	=	$Q_i(t-1) + A_i(t)$, i.e. the length of Q_i after immediately after arrivals, in timeslot t
a^+	=	$\max(a, 0)$
$\Re_+$	=	The set of nonnegative real numbers
$H(x\|y)$	=	$x \log \frac{x}{y} + (1-x) \log \frac{1-x}{1-y}$

We make the following assumptions regarding the arrival and channel processes:

Assumption 1. *(Motivated by [5])*

1) *The channel state process:*
   a) *Let $\mathcal{I}$ denote the set of possible channel states. The channel state process has a stationary distribution $\pi = [\pi_i]_{i \in \mathcal{I}}$, and $\pi_i > 0$ for all $i \in \mathcal{I}$.*
   b) *Let $s[m]$ denote the channel state in timeslot m. Given $\epsilon > 0$ there exists a positive integer M_0 such that for all $M \geq M_0$, all $i \in \mathcal{I}$, and all k, we have*
   $$\mathbb{E}\left[\left|\pi_i - \frac{1}{M}\sum_{m=k}^{k+M-1} \mathbb{1}_{s[m]=i}\right|\right] < \epsilon.$$
   c) *There exists $\hat{\mu} \in \Re_+$ such that $X_{ij}(t) \leq \hat{\mu}$ for all i, j, t.*
2) *The arrival process:*
   a) *The arrivals to each queue Q_i form a stationary process, with mean $\lambda_i := \mathbb{E}[A_i(1)]$.*
   b) *The given set of arrival rates lies inside the throughput region of the system, i.e., there exists a static service split rule that can stabilize the system under the given arrival process.*
   c) *Given $\epsilon > 0$ there exists a positive integer M_1 such that for all $M \geq M_1$, and for all k, we have*

$$\mathbb{E}\left[\left|\lambda_i - \frac{1}{M}\sum_{m=k}^{k+M-1} A_i(m)\right|\right] < \epsilon, \quad \forall i.$$

d) *The arrival process satisfies, for all* i,
$$\lim_{M\to\infty} M^2 \mathbb{P}(A_i(1) > M) = 0. \quad \diamond$$

Our aim is to design a service policy for this system that meets certain performance metrics that will be subsequently defined. A service policy is essentially a rule that allocates the servers to the queues in every timeslot by defining the random variables $Y_{ij}(t)$, where

$$Y_{ij}(t) = \begin{cases} 1 & \text{if } S_j \text{ is allocated to serve } Q_i \text{ in timeslot } t, \\ 0 & \text{otherwise.} \end{cases}$$

We impose the condition that in a given timeslot, a given server can be allocated to serve at most one queue. This condition translates to the following: for all t and all $j \in \{1, 2, \ldots, n\}$, any valid service policy must obey $\sum_{i=1}^{n} Y_{ij}(t) \le 1$. For concreteness, we follow the convention that in a timeslot, there are first arrivals (if any) to the queues, then possible service, and finally, the queue-lengths are measured at the end of the timeslot. Thus, the individual queues in the system evolve according to the following equation: for $1 \le i \le n$,

$$Q_i(t) = \left(Q_i(t-1) + A_i(t) - \sum_{j=1}^{n} X_{ij}(t) Y_{ij}(t)\right)^+.$$

Each queue stores the incoming packets in a buffer of infinite size, so that no packets are ever dropped. The queuing system is started at time $-\infty$. Our objective is to design a service rule that allocates the servers to the queues in each timeslot t based on the information of the entire history of queue-lengths, arrival and channel realizations and allocation decisions, and also the arrivals and channel realizations in the current timeslot. Our first goal is to identify policies other than MaxWeight-type policies which are throughput-optimal under Assumption 1.

The reason for identifying classes of throughput-optimal policies other than the MaxWeight class is to significantly improve the performance of the system. It is difficult to analyze the large-deviations performance under the very general conditions in Assumption 1. Instead, we study the small-queue performance of different algorithms under the following more restrictive set of assumptions:

Assumption 2. *The number of packet arrivals to queue* Q_i *in timeslot* t *is the random variable* $A_i(t)$, *where*

$$A_i(t) = \begin{cases} \bar{K} & \textit{with probability } p, \\ 0 & \textit{with probability } 1-p, \end{cases}$$

where $\bar{K} \ge 1$ *is an integer with* $p\bar{K} \in (0, 1)$. *In timeslot* t, *the server* S_j *can potentially serve* $X_{ij}(t)$ *packets from* Q_i, *where* $X_{ij}(t)$ *are modeled as Bernoulli random variables with*

$$X_{ij}(t) = \begin{cases} 1 & \textit{with probability } q, \\ 0 & \textit{with probability } 1-q, \end{cases}$$

with $q \in (0, 1)$. *All the random variables* $A_i(t)$ *and* $X_{jk}(s)$ *are assumed to be mutually independent for all possible values of the involved parameters.* $\diamond$

Our goal is to design policies that under Assumption 2 and for every integer $b \ge 0$, result in a strictly positive value of

$$\alpha(b) := \liminf_{n\to\infty} \frac{-1}{n} \log \mathbb{P}\left(\max_{1\le i\le n} Q_i(0) > b\right).$$

Further, the complexity of the policy must be $O(n^2)$ computations per timeslot. The reason we choose this benchmark is that, as we later show, the MaxWeight algorithm has a complexity $\Omega(n^2)$.

We refer to the event $\{\max_i Q_i(t) > b\}$ as the *small buffer overflow event* or simply the *overflow event.* The probability term in the above expression can thus be thought of as the probability of the overflow event under the stationary distribution of the queue-length process (provided one exists).

The function $\alpha(b)$ is called the rate function in the large deviations theory. If a scheduling algorithm results in a positive value of $\alpha(b)$, then for large values of n, we have

$$\mathbb{P}\left(\max_{1\le i\le n} Q_i(0) > b\right) \approx e^{-n\alpha(b)}.$$

Thus, the probability of the small buffer overflow event decays to zero very rapidly with n, and therefore it is desirable to have as large a value of $\alpha(b)$ as possible.

III. The iHLQF-class of Algorithms

In this section, we present a class of scheduling rules called iHLQF. We show that any algorithm in this class is throughput-optimal (Theorem 1), and relate the iLQF with PullUp algorithm in [3] to the iHLQF-class.

We consider a class of algorithms called iHLQF (iterated Heaviest matching with Longest Queues First), which is a generalization of the iLQF (iterated Longest Queues First) algorithms presented in [3]. The iLQF-class of algorithms are defined for systems with ON-OFF channels. A particular algorithm in the iLQF-class, called iLQF with PullUp, was shown to be rate function optimal for the small buffer overflow event under Assumption 2 with $\bar{K} = 1$ ([3], Thm. 4), and was shown to have a positive rate function for all $\bar{K} > 0$ ([3], Corollary 2). Here we present its generalization (namely, iHLQF) to multi-rate channels. In every timeslot, the iHLQF rule proceeds in multiple rounds of server allocation as explained below:

Definition 1 (The iHLQF (iterated Heaviest matching with Longest Queues First) rule)**.** *In a timeslot* t,

1) *Update the queue-length vector to account for the arrivals, i.e., compute* $Q_i^{(0)}(t)$ *for all i. Initialize* $k = 1$.
2) *For all* $j \in \{1, 2, \ldots, n\}$, *define*
$$\mathcal{T}_j := \arg\max_{1\le i\le n} Q_i^{(0)}(t) X_{ij}(t),$$
$$\textit{and } c_j := \min\left[\arg\max_{m\in\mathcal{T}_j} X_{mj}(t)\right].$$
For any server S_j *and all* i, *if* $X_{ij}(t) < X_{c_j j}(t)$, *then redefine* $X_{ij}(t) = 0$, *and use this updated value of* $X_{ij}(t)$ *throughout the rest of the timeslot* t. *Let* L *denote the length of the longest queue(s) immediately after arrivals. Throughout the description of this algorithm, let* $\mathcal{V} \subseteq \mathcal{S}$ *denote the set of unallocated servers.*
3) *In round* k, *define a bipartite graph* $G_k(\mathcal{U}_k \cup \mathcal{V}_k, \mathcal{E}_k)$, *where the set of nodes* $\mathcal{U}_k$ *represents the set of queues of length* L *(i.e.* $Q_i^{(k-1)}(t) = L$*), the set of nodes* $\mathcal{V}_k$ *represents the servers in* $\mathcal{V}$, *and* $\mathcal{E}_k$ *is the set of weighted*

edges. The edge between nodes representing queue Q_i and server S_j has a weight equal to $X_{ij}(t)$. Find a maximum weight matching $\mathcal{M}_k$ in the graph G_k, breaking ties arbitrarily. Allocate servers to queues according to the matching $\mathcal{M}_k$, update the queue-lengths to account for service, and remove the used servers from the set $\mathcal{V}$.

4) *If $\mathcal{V} = \emptyset$, stop. Else, decrement L by 1. If $L = 0$, then stop. Else, increment k by 1, and go to Step 3.* ⋄

Here is a description of the algorithm in words: in every timeslot, have multiple rounds of server allocation. In each round, choose the heaviest (edge-weight) matching between the set of *longest* queues and available serves, breaking ties between multiple heaviest matchings arbitrarily. Here, the weight of an edge is the corresponding channel rate. Allocate servers to queues according to the matching, *update the queue-lengths,* remove the allocated servers from further consideration, and proceed to the next round. When choosing the heaviest matching, the iHLQF rule allocates a server to a queue only if its channel to that queue has a high enough rate.

Note that iHLQF is a class of rules and not a single rule, as a result of the arbitrary tie-breaking between largest weight matchings. We first establish a crucial property of the iHLQF-class rules that is useful in proving their throughput-optimality. In words, this property says that in every timeslot, the weight of the schedule chosen by any iHLQF-class rule is at most an additive constant away from that chosen under the throughput-optimal MaxWeight rule.

Lemma 1. *Fix any iHLQF-class rule. Consider any timeslot t, and let the queue-lengths immediately after arrivals in that timeslot be ℓ_i. Let the iHLQF rule assign server S_j to Q_{a_j}, and c_j be as defined in step 2 in the definition of the iHLQF-class rules (Definition 1). Then, $\sum_{j=1}^{n} X_{a_j j}(t)\ell_{a_j} \geq \sum_{j=1}^{n} X_{c_j j}(t)\ell_{c_j} - n^2\hat{\mu}^2$.*

Theorem 1 (Throughput-optimality of iHLQF)**.** *Under Assumption 1 on the arrival and channel processes, any iHLQF-class rule makes the system stable in the mean, i.e.,*

$$\limsup_{p\to\infty} \frac{1}{p}\sum_{k=0}^{p-1} \mathbb{E}\left[\sqrt{\sum_{i=1}^{n} Q_i^2(k)}\right] < \infty.$$

In addition, if the arrival and channel state processes are such that the iHLQF rule makes the queuing system an aperiodic Markov chain with a single communicating class, then the stability in the mean implies that the Markov chain is positive recurrent [8].

The proofs of Lemma 1 and Theorem 1 are analogous to those of Lemma 3 and Theorem 5 in Section V respectively. We present proofs for the claims in Section V alone in order to avoid repetition and also because they admit simpler notation. For a formal proof of Lemma 1, please see [4].

We now turn our attention to the iLQF with PullUp rule, introduced in [3]. This rule is essentially an iHLQF-class rule for the system under Assumption 2. The iLQF with PullUp rule employs a particular form of tie-breaking between the heaviest matchings, and in its original form, it terminates after the round when it cannot find a matching that serves all the queues under consideration (queues of length L). This stopping rule ensures that the complexity of the rule is limited to $O(n^4)$ computations per timeslot. If we instead allow the iLQF with PullUp rule to terminate in the same way as an iHLQF-class rule (namely, when no more servers are left or when all the nonempty queues have been considered for allocation), then it retains its rate-function optimality for the system under Assumption 2 with $\bar{K} = 1$, yields a strictly positive rate function for all $\bar{K} \geq 1$, and is also throughput-optimal for the system under Assumption 1 with Bernoulli (0-1) channels. We refer to this rule as the Modified iLQF with PullUp rule. The following theorem formally summarizes these properties.

Theorem 2 (Properties of Modified iLQF with PullUp)**.** *The Modified iLQF with PullUp belongs to the iHLQF class of rules, and the conclusions of Theorem 1 apply. The Modified iLQF with PullUp rule is rate function optimal for the system under Assumption 2 with $\bar{K} = 1$, and results in*

$$\liminf_{n\to\infty} \frac{-1}{n}\log \mathbb{P}\left(\max_{1\leq i\leq n} Q_i(0) > b\right) = (b+1)\log\frac{1}{1-q},$$

for all integers $b \geq 0$. Further, it yields a strictly positive rate function under Assumption 2 for all $\bar{K} \geq 1$, and can be implemented in $O(n^5)$ computations per timeslot.

Proof: Omitted to avoid repetition. For proofs of rate function optimality and computational complexity, we refer the reader to [3] and [4] respectively. ■

Thus, the answer to question 1 in Section I-A is "yes, the iLQF rules (and their generalization, iHLQF, for multi-rate channels) are throughput-optimal for the system."

IV. The MaxWeight Rule

In this section, we show that the classic MaxWeight scheduling rule yields a zero rate function for the small buffer overflow event (Theorem 3), and in fact yields no decay in the probability of the small buffer overflow event as the system size increases (Theorem 4). We then establish a lower bound (computations per timeslot) on the complexity of the MaxWeight rule (Lemma 2).

The classic MaxWeight rule [16] results in the following service allocation rule for our system, with a particular tie-breaking rule for the sake of concreteness:

Definition 2 (The MaxWeight Rule, [16])**.** *In every timeslot, each server independently picks a queue that maximizes the product of queue-length and channel rate, breaking ties in favor of the smallest queue-index.* ⋄

This service allocation rule is throughput-optimal for the system. But as the next theorem shows, it yields a zero rate function for the small buffer overflow event.

Theorem 3 (MaxWeight gives zero rate function)**.** *Under Assumption 2 with $\bar{K} = 1$, with the MaxWeight rule for allocating servers to queues, and for any fixed integer $b \geq 0$,*

$$\limsup_{n\to\infty} \frac{-1}{n}\log \mathbb{P}\left(\max_{1\leq i\leq n} Q_i(0) > b\right) = 0.$$

The main idea behind the proof is to show that under the MaxWeight rule, the overflow event has at least a constant probability even for n large.

Proof: Fix any integer T, and consider the queues at the end of timeslot T. Define $\gamma_n := 1 - e^{-\sqrt{n}}$, and $p' := p/2$.

Timeslot $T+1$:

By the Chernoff bound, there exists an integer n_1 such that for all $n \geq n_1$, with probability at least γ_n, at least np' queues see arrivals in timeslot $T+1$. Define $\alpha := -2/\log(1-q)$. Fix an integer N such that for all $n \geq N$, we have $\alpha \log n \geq 1$. Define $\beta := \alpha \log n$, and consider the first β queues in the order of priority for service (after arrivals). In particular, the first queue in the order of priority is the longest queue with the smallest index, the second one is the longest queue with the second smallest index or the second longest queue with the smallest index, and so on. Let the set of these queues be $\mathcal{Q}^\star := \{Q_{i_1}, Q_{i_2}, \ldots, Q_{i_\beta}\}$. Let E_j denote the event that server S_j is not connected with any of the queues in $\mathcal{Q}^\star$. Then, since $\mathbb{P}(E_j) = (1-q)^\beta = (1-q)^{\alpha \log n}$, we have

$$\mathbb{P}\left(\bigcup_{j=1}^n E_j\right) \leq \sum_{j=1}^n \mathbb{P}(E_j) = n(1-q)^{\alpha \log n} = \frac{1}{n}.$$

Thus, with probability at least $1 - 1/n$, each one of the servers is connected to a queue in $\mathcal{Q}^\star$. By the definition of the MaxWeight rule, a server connected to one of the queues in $\mathcal{Q}^\star$ is allocated to one of the queues in $\mathcal{Q}^\star$. Since $|\mathcal{Q}^\star| = \beta$ and at least np' queues had packet arrivals, it follows that at the end of timeslot $T+1$, the system has at least $np' - \alpha \log n$ queues at length 1 or more. By the union bound (for $n \geq \max(N, n_1)$), the probability of this event is at least $1 - (1/n + e^{-\sqrt{n}})$. Let this set of queues (of length at least 1) be called $\mathcal{A}_1$.

Timeslot $T+2$:

The arrivals in the timeslot $T+2$ are independent of all the random variables involved in the definition of the set $\mathcal{A}_1$. Thus, by appropriately using the Chernoff bound, there exists an integer n_2 such that for all $n \geq n_2$, with probability at least γ_n, at least p' fraction of queues in the set $\mathcal{A}_1$ see arrivals in timeslot $T+2$. By an argument similar to that for Timeslot $T+1$, it follows that, with probability at least $1 - 1/n$, no more than $\alpha \log n$ of the queues receive service. Combining the results for the timeslots and using the union bound, we have the following conclusion: for all $n \geq \max(N, n_1, n_2)$, with probability at least $1 - 2(1/n + e^{-\sqrt{n}})$, there exists a set $\mathcal{A}_2$ of queues such that:

- $|\mathcal{A}_2| \geq p'(np' - \alpha \log n) - \alpha \log n \geq np'^2 - 2\alpha \log n$.
- Each queue in $\mathcal{A}_2$ has a length at least 2.

Continuing this way (formally, by induction), the following claim holds: at the end of timeslot $T + b + 1$, for $n \geq \max(N, n_1, n_2, \ldots, n_{b+1})$, with probability at least $1 - (b+1)(1/n + e^{-\sqrt{n}})$, there exists a set $\mathcal{A}_{b+1}$ of queues such that:

- $|\mathcal{A}_{b+1}| \geq p'(np'^b - b\alpha \log n) - \alpha \log n$, implying $|\mathcal{A}_{b+1}| \geq np'^{b+1} - (b+1)\alpha \log n$.
- Each queue in $\mathcal{A}_{b+1}$ has a length at least $b+1$.

There exists an integer n_0 such that for all $n \geq n_0$, we have $np'^{b+1} - (b+1)\alpha \log n \geq 1$ and $(b+1)(1/n + e^{-\sqrt{n}}) \leq 1/2$. Hence, for a system with $n \geq \max(N, n_0, n_1, \ldots, n_{b+1})$, with at least a probability $1/2$ and starting with any initial configuration of queue-lengths, we have a queue of length $b+1$ at the end of a further $b+1$ timeslots. If we consider the $b+1$ timeslots leading to and including timeslot 0, i.e., consider $T = -(b+1)$, then the above result shows that even for large n, the small buffer overflow event occurs with at least a constant probability, and the proof is complete. ■

The result of Theorem 3 can be strengthened to the following:

Theorem 4. *Consider any function $f : \Re_+ \to \Re_+ \backslash \{0\}$ such that $\lim_{x\to\infty} f(x) = \infty$. Then, under Assumption 2, with the MaxWeight rule for allocating servers to queues, and for any fixed integer $b \geq 0$,*

$$\limsup_{n\to\infty} \frac{-1}{f(n)} \log \mathbb{P}\left(\max_{1\leq i\leq n} Q_i(0) > b\right) = 0.$$

Proof: The result for the case $\bar{K} = 1$ follows from the proof of Theorem 3, which shows that under Assumption 2 with $\bar{K} = 1$, the MaxWeight rule results in at least a constant probability for the small buffer overflow event, for n large enough. The proof for the case $\bar{K} > 1$ is almost identical. ■

Thus, the answer to question 2 in Section I-A is "the MaxWeight algorithm is very inefficient at keeping the per-user queues small." The main reason behind these negative results is that the MaxWeight rule potentially assigns all the available servers to serve *the* longest queue, essentially treating a slightly shorter queue as if it were empty. When a large number of servers are available, this results in draining the longest queue(s) much more than is warranted by good load-balancing, and also leads to the following situation: when the MaxWeight rule runs into a state when a significant fraction of the queues is long (note that such a state is reached infinitely often, almost surely, because the system is positive recurrent under MaxWeight), then it is very difficult to leave this state "quickly." Therefore, the MaxWeight rule is not effective in keeping the queues *really* small.

Lemma 2 (Complexity of MaxWeight)**.** *Under Assumption 2, implementing the MaxWeight rule requires $\Omega(n^2)$ computations per timeslot.*

Proof: Let Assumption 2 hold. In a given timeslot t, the MaxWeight rule needs to find, for every server S_j, a queue Q_i that maximizes the product of the instantaneous channel rate $X_{ij}(t)$ and the length of queue Q_i after packet arrivals (if any). Thus, for every server, the rule needs to perform n multiplications. All the n multiplications are necessary: not performing even one of these multiplications (in the worst case) can lead to an incorrect allocation. Further, since all the channels are mutually independent, each server needs n separate computations, i.e., the calculations not involving $X_{\cdot j}(t)$ are useless for S_j. Thus, any implementation of the MaxWeight rule requires $\Omega(n^2)$ computations per timeslot. ■

Note that Lemma 2 holds under much weaker assumptions on the channel process. This motivates us to design a scheduling rule that, in addition to throughput-optimality, also guarantees a good delay performance, and has a computational complexity comparable to that of the MaxWeight rule.

V. The SSG Scheduling Rule

In this section, we propose the Server-Side Greedy (SSG) service rule for the problem stated in Section II. This service rule can be thought of as a recursive version of the MaxWeight rule, where the queue-lengths are updated after each server finishes its service. We show that the SSG rule is throughput-optimal for the system (Theorem 5). Under Assumption 2, it results in a strictly positive value of the rate function, $\alpha(b)$ for every integer $b \geq 0$ (Theorems 6, 7). It can be implemented in $O(n^2)$ computations per timeslot (Theorem 8).

This (SSG) rule proceeds in multiple rounds of service allocation in every timeslot, as explained below.

Definition 3 (The SSG rule)**.** *In every timeslot t,*

1) *Update the queue-lengths to account for arrivals, i.e., compute $Q_i^{(0)}(t)$ for all i. Initialize $k = 1$.*
2) *In round k, allocate server S_k to serve a queue Q_w that maximizes the product $Q_i^{(k-1)}(t)X_{ik}(t)$, breaking ties in favor of the smallest queue-index. Update the length of Q_w to account for service, i.e., $Q_w^{(k)}(t) := \left(Q_w^{(k-1)}(t) - X_{wk}(t)\right)^+$ and $Q_i^{(k)}(t) := Q_i^{(k-1)}(t)$ for all $i \neq w$.*
3) *If $k = n$, stop. Else, increment k by 1, go to step 2.* $\diamond$

Unlike the MaxWeight rule, the SSG rule updates queue-lengths after each server finishes its service. Now we analyze the SSG service rule in detail. Our first aim is to establish the throughput-optimality of the SSG rule. We first establish that in every timeslot, the weight of the service schedule selected by the SSG rule is at most an additive constant away from the maximum possible. That is, under the same queue-lengths at the beginning of the timeslot and the same arrivals to the queues, the sum of the channel-rate-weighted queue-lengths selected for service under the MaxWeight rule is at most an additive constant larger than that under the SSG rule.

Lemma 3. *Fix any timeslot t, and let the queue-lengths immediately after arrivals in that timeslot be ℓ_i. Let the SSG rule assign server S_j to serve Q_{a_j}. Under the same initial queue-lengths and the same arrivals to the queues, let the MaxWeight rule assign S_j to Q_{b_j}. Then, $\sum_{j=1}^n X_{a_j j}(t)\ell_{a_j} \geq \sum_{j=1}^n X_{b_j j}(t)\ell_{b_j} - n^2\hat{\mu}^2$.*

Proof: Fix any $j \in \{1, 2, \ldots, n\}$. If $a_j = b_j$, then we have $X_{a_j j}(t)\ell_{a_j} = X_{b_j j}(t)\ell_{b_j}$ and corresponding terms cancel from both sides. Hence, we focus on the case $a_j \neq b_j$. In this case, Q_{b_j} is connected to server S_j, but does not get served by S_j under the SSG rule. It follows that

$$\begin{aligned} X_{a_j j}\ell_{a_j} &\overset{(a)}{\geq} X_{a_j j}Q_{a_j}^{(j-1)}(t) \overset{(b)}{\geq} X_{b_j j}Q_{b_j}^{(j-1)}(t) \\ &\overset{(c)}{\geq} X_{b_j j}(\ell_{b_j} - n\hat{\mu}) \geq X_{b_j j}\ell_{b_j} - n\hat{\mu}^2. \end{aligned}$$

Here, inequality (a) holds because $\ell_{a_j} = Q_{a_j}^{(0)}(t) \geq Q_{a_j}^{(j-1)}(t)$, since queue-lengths can only monotonically decrease as the successive rounds proceed (recall that the arrivals occur before round 1). Inequality (b) holds because in round j, the SSG rule allocates server S_j to a queue that maximizes the product of the channel-rate and queue-length. Inequality (c) holds because any given queue can receive at most $\hat{\mu}$ units of service in a given round (of SSG), implying that the length of Q_{b_j} after $j-1$ rounds is at least $\ell_{b_j} - (j-1)\hat{\mu} \geq \ell_{b_j} - n\hat{\mu}$. Therefore,

$$\sum_{j=1}^n X_{a_j j}(t)\ell_{a_j} \geq \sum_{j=1}^n X_{b_j j}(t)\ell_{b_j} - n^2\hat{\mu}^2.$$

Thus, the proof is complete. ∎

Theorem 5 (Throughput-optimality of SSG)**.** *Under Assumption 1 on the arrival and channel processes, the SSG rule makes the system stable in the mean, i.e.,*

$$\limsup_{p\to\infty} \frac{1}{p}\sum_{k=0}^{p-1} \mathbb{E}\left[\sqrt{\sum_{i=1}^n Q_i^2(k)}\right] < \infty.$$

In addition, if the arrival and channel state processes are such that the SSG rule makes the queuing system an aperiodic Markov chain with a single communicating class, then the stability in the mean implies that the Markov chain is positive recurrent [8].

Proof: For any server allocation rule R, in any timeslot, let the channel-rate-weighted sum of the queues be called the weight of the service rule in that timeslot. In our notation, the weight of a rule R in timeslot t is

$$W^R(t) := \sum_{i=1}^n \sum_{j=1}^n (Q_i(t-1) + A_i(t))X_{ij}(t)Y_{ij}(t).$$

The proof of this Theorem is on the same lines as that of Theorem 1 in [5], which establishes the following: consider a service rule R that, in every timeslot, picks a schedule whose weight is an arbitrarily high fraction of that of the MaxWeight schedule whenever the sum of the queue lengths is large enough. Then, the rule R makes the system stable in the mean, and positive recurrent under the stated conditions. That proof applies almost unchanged when the weight of the schedule selected by the candidate policy R is at most an additive constant smaller than the maximum possible, provided that this constant is independent of the queue-lengths and is a function of the system parameters only. This result and Lemma 3 complete the proof. ∎

Remark 1. *Lemma 3 and Theorem 5 hold even if the SSG rule chooses an arbitrary tie-breaking rule in Step 2.*

Next, we show that if two queuing systems have sample-path coupled arrivals and channels, both implement the SSG rule, and at the end of a timeslot, one system has queues that are respectively longer than the corresponding queues in the second system, then this property continues to hold for all the future timeslots.

Lemma 4 (Sample-path dominance)**.** *Under Assumption 2, consider two queuing systems $\underline{Q}$ and $\underline{R}$ with queues $\mathcal{Q} = [Q_1, Q_2, \ldots, Q_n]$ and $\mathcal{R} = [R_1, R_2, \ldots, R_n]$, such that at the end of some timeslot t, we have $Q_i(t) \leq R_i(t)$ for all i. Both the systems have the same arrivals and channel realizations for all times, and in particular for the timeslot $t+1$. Both implement the SSG rule. Then, $Q_i(t+1) \leq R_i(t+1)$ $\forall i$.*

Proof: We need to prove that $Q_i^{(n)}(t+1) \le R_i^{(n)}(t+1)$ for all i. Suppose that for some $k \in \{1, 2, \ldots, n\}$, we have $Q_i^{(k-1)}(t+1) \le R_i^{(k-1)}(t+1)$ for all i. We need to prove that $Q_i^{(k)}(t+1) \le R_i^{(k)}(t+1)$ for all i, and the proof would be complete by the principle of mathematical induction.

Let in the system $\underline{R}$, server S_k be allocated to serve queue R_j. If (in the system $\underline{Q}$) the server S_k is allocated to serve Q_j, then there is nothing to prove. If S_k is not allocated to serve Q_j, it must be because of one of the following reasons:

1) $Q_j^{(k-1)}(t+1) < Q_r^{(k-1)}(t+1)$ for some r, and S_k is connected to Q_r, i.e., $X_{rk}(t+1) = 1$.
2) $Q_j^{(k-1)}(t+1) = Q_m^{(k-1)}(t+1)$ for some $m < j$, and Q_j, Q_m are among the longest queues connected to S_k, so according to the tie-breaking rule, queue Q_m is served in the k^{th} round.

In case 1, we have (by hypothesis)

$$Q_j^{(k-1)}(t+1) < Q_r^{(k-1)}(t+1) \le R_r^{(k-1)}(t+1),$$

and $R_r^{(k-1)}(t+1) \le R_j^{(k-1)}(t+1)$ by the definition of the SSG rule. (Otherwise, S_k would have been allocated to serve R_r because $X_{rk}(t+1) = 1$.) Hence, irrespective of the allocation of S_k (in the system $\underline{Q}$), we have $Q_j^{(k)}(t+1) \le R_j^{(k)}(t+1)$, and consequently $R_i^{(k)}(t+1) \ge Q_i^{(k)}(t+1)$ for all i.

In case 2, we must have $Q_j^{(k-1)}(t+1) < R_j^{(k-1)}(t+1)$, because if $Q_j^{(k-1)}(t+1) = R_j^{(k-1)}(t+1)$, then

$$R_j^{(k-1)}(t+1) = Q_j^{(k-1)}(t+1) = Q_m^{(k-1)}(t+1) \le R_m^{(k-1)}(t+1),$$

and $m < j$ implies that in the system $\underline{R}$, server S_k must have been allocated to R_m and not R_j. Therefore, we have $Q_j^{(k-1)}(t+1) < R_j^{(k-1)}(t+1)$, and hence $Q_j^{(k)}(t+1) \le R_j^{(k)}(t+1)$, implying $R_i^{(k)}(t+1) \ge Q_i^{(k)}(t+1)$ for all i. ■

As in the case of the proof of Theorem 3 in [3], this sample-path property is the key to obtaining rate function positivity results. The next technical lemma provides a sufficient condition for all the longest queues in the system (after arrivals) to receive at least one unit of service.

Lemma 5. *Under Assumption 2, let the set of longest queues after arrivals be of cardinality k. If in that timeslot, each one of the longest queues is connected to at least k servers, then all the longest queues are served at least once under SSG.*

Proof: Let $\{Q_{i_1}, Q_{i_2}, \ldots, Q_{i_k}\}$ be the set of longest queues. For $1 \le j \le k$, if Q_{i_j} is connected to server S_m, then it is a longest queue connected to S_m. Since Q_{i_j} is connected to at least k servers, there are only $k-1$ other longest queues in the system, and the queue-lengths are updated after service, Q_{i_j} is served by at least one server. ■

We now show that under the SSG rule, the probability that the max. queue in the system increases in a given timeslot is extremely small for n large.

Lemma 6. *Let Assumption 2 hold with $\bar{K} = 1$. Fix any $p' \in (p, 1)$ and $\delta \in (0, \frac{q(1-p')}{2-q})$. In particular, let $p' = (1+p)/2$ and $\delta = \frac{q(1-p')}{2}$. Then, under the SSG rule, for n large enough, and for any given t,*

$$\mathbb{P}\left(\max_{1\le i\le n} Q_i(t+1) > \max_{1\le i\le n} Q_i(t)\right) \le n\delta \exp\left(\frac{-2n\delta H(\frac{q}{2}|q)}{q}\right)$$
$$+ n(1-q)^{n\delta} + \exp\left(-nH(p'|p)\right). \qquad (1)$$

Proof: Consider a situation where at the beginning of timeslot $t+1$, all the queues have length $= m$. The third term on the RHS of Equation (1) upper-bounds the probability that in a given timeslot t, there are more than np' queues with arrivals, and follows from the Chernoff bound. Conditioned on the (high probability) event of no more than np' arrivals (and adding dummy packets if necessary to ensure exactly np' queues with arrivals, since by doing so, we only get a "worse" system, a system with sample-pathwise longer queues for all future times, thanks to Lemma 4), the queue-length profile after arrivals is:

Queue-length	$m+1$	m
Number of queues	np'	$n(1-p')$

The second term on the RHS of Equation (1) upper-bounds the probability of the complement of the following event: $n(p'-\delta)$ different longest queues are served in the first $n(p'-\delta)$ rounds of server allocations. The first term, in conjunction with Lemma 5, upper-bounds the probability of the complement of the following event: the (remaining) $n\delta$ longest queues are (each) served by some of the remaining servers. The result follows by the union bound and Lemma 4. A formal proof is omitted due to lack of space, and available in [4]. ■

Next, we establish that under the SSG rule and for n large, the max. queue-length in the system decreases in a constant number of timeslots with at least a constant probability.

Lemma 7. *Under the SSG rule, under Assumption 2 with $\bar{K} = 1$, there exists a constant integer k, independent of n, such that for all n large enough, and for all t,*

$$\mathbb{P}\left(\max_{1\le i\le n} Q_i(t+k) < \max_{1\le i\le n} Q_i(t) \,\middle|\, \max_{1\le i\le n} Q_i(t) > 0\right) \ge \frac{1}{2}.$$

Proof: The intuition behind the proof is that in any given timeslot, there are approximately np arrivals to the system, and because a given server can potentially serve any one of approximately nq of the queues (with preference to the longer queues), the system has a service capacity for almost n packets. Thus, there is a net "drain" in the number of packets in the system. A formal proof is omitted due to lack of space, and available in [4]. ■

As a consequence of Lemmas 6 and 7, the maximum queue-length in the system, $Q_{max}(t)$ has the following behavior: over a constant number of timeslots, it increases by a finite amount with very low probability, and decreases with at least a constant probability. Thus, it is reasonable to expect that the stationary distribution of $Q_{max}(t)$ is strongly concentrated around 0. Indeed, this is the essence of our next claim.

Theorem 6 (Positive rate function under SSG)**.** *Let Assumption 2 hold with $\bar{K} = 1$. Fix any $p' \in (p, 1)$ and $\delta \in (0, \frac{q(1-p')}{2-q})$. In particular, let $p' = (1+p)/2$ and $\delta = \frac{q(1-p')}{2}$. Fix any constant integer $b \geq 0$. If the system uses the SSG rule for allocating servers to queues, then*

$$\liminf_{n\to\infty} \frac{-1}{n} \log \mathbb{P}\left(\max_{1\leq i\leq n} Q_i(0) > b\right) \geq (b+1)\min\left\{H(p'|p), \delta \log\frac{1}{1-q}, \frac{2\delta H(\frac{q}{2}|q)}{q}\right\} > 0.$$

Proof: The proof is similar to that of Theorem 3 in [3]. In particular, under the SSG rule, the system has the sample-path dominance property (Lemma 4), an exponentially decaying probability of $Q_{max}(t)$ increasing in a timeslot (Lemma 6), and a constant probability of $Q_{max}(t)$ decreasing in a constant number of timeslots (Lemma 7). We omit the details. ■

Next, we show that the SSG rule returns a positive rate function for the small buffer overflow event under a bursty arrival process (Assumption 2), for any fixed integer $\bar{K} \geq 1$.

Theorem 7. *Let Assumption 2 hold. Fix any $p' \in (p, 1/\bar{K})$ and $\delta \in (0, \frac{q(1-p'\bar{K})}{\bar{K}(2-q)})$. In particular, let $p' = (p+1/\bar{K})/2$ and $\delta = \frac{q(1-p'\bar{K})}{2\bar{K}}$. Fix any constant integer $b \geq 0$. If the system uses the SSG rule for allocating servers to queues, then*

$$\liminf_{n\to\infty} \frac{-1}{n} \log \mathbb{P}\left(\max_{1\leq i\leq n} Q_i(0) > b\right) \geq \left\lceil \frac{b+1}{\bar{K}} \right\rceil \min\left\{H(p'|p), \delta \log\frac{1}{1-q}, \frac{2\delta H(\frac{q}{2}|q)}{q}\right\} > 0.$$

Proof: The proof is very similar to the proof of Theorem 6 and has been omitted. ■

An immediate strengthening of the above results is obtained by maximizing the RHS with respect to p' and δ over the appropriate ranges. We now turn our attention to the computational complexity of implementing the SSG rule.

Theorem 8 (Complexity of the SSG rule)**.** *The SSG rule can be implemented in $O(n^2)$ computations per timeslot.*

Proof: Consider the following implementation of the SSG rule: the system maintains a vector of queue-lengths $[Q_1, Q_2, \ldots, Q_n]$ and updates it as arrivals and service occur. It also maintains a vector that records the queue-index that server S_k is allocated to, and updates it in every round. Updating the queue-length vector after arrivals (Step 1 in the definition of SSG) involves n additions and can be implemented in $O(n)$ computations. In Step 2, any given round can be implemented in $O(n)$ computations, and therefore the entire Step 2 (n rounds) can be implemented in $O(n^2)$ computations. Step 3 can be implemented with $O(1)$ computations. Hence, the overall complexity is $O(n^2)$ computations per timeslot. ■

Thus, we have designed a scheduling algorithm (the SSG rule) that is throughput-optimal, yields a positive rate function for the small buffer overflow event, and has a computational complexity $O(n^2)$, which is no larger than the MaxWeight rule. This answers question 3 in Section I-A in the affirmative. In view of these results, the new intuition that emerges from this work is that we should not allocate service to maximize the channel-weighted sum of queue-lengths. We should allocate resources in an iterative fashion, taking into account the effects of prior allocations. This results in good performance (small per-user queues) in addition to throughput-optimality.

VI. Simulation Results

We compare the performance of the proposed SSG rule with the MaxWeight rule and the Modified iLQF with PullUp rule studied in [3]. We run the simulations for 10^6 timeslots each.

In the first set of simulations, we consider a system with bursty arrivals as per Assumption 2 with $\bar{K} = 10$ and $p = 0.095$, i.e., a system with 95% of the maximum symmetric load. The channel ON probability is set to $q = 0.75$. We vary the number (n) of queues and servers in the system, and study the empirical probability of buffer overflow and the empirical delay distribution of packets. The results are summarized in Figures 2 and 3. It can be seen that depending upon the system size, the MaxWeight algorithm needs about 3 to 7 times as much buffer as SSG. Further, the performance of the MaxWeight algorithm actually *degrades* with the system size, while that of the SSG rule improves. Similar conclusions hold for the per-packet delay under the two algorithms.

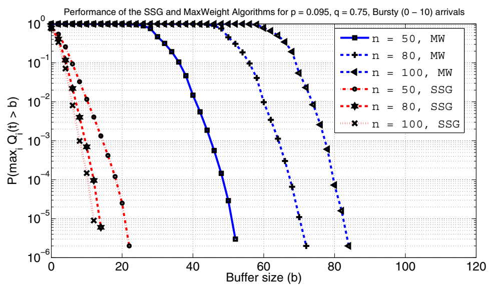

Fig. 2. SSG v/s MaxWeight: Buffer overflow probabilities

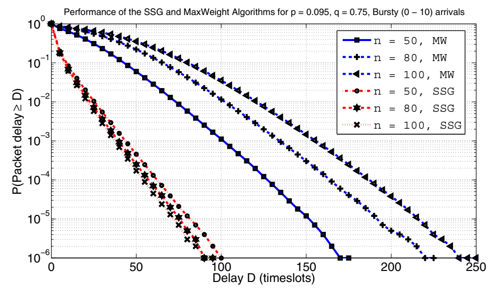

Fig. 3. SSG v/s MaxWeight: Packet delay profiles

In the second set of simulations (Figure 4), we compare the performance of the SSG algorithm against the Modified iLQF with PullUp algorithm. We analyze a system with asymmetric arrival rates to the queues. Of the $n = 20$ queues, we choose three queues to receive much higher mean loads (L) compared to the others. In our simulations, queues Q_{11}, Q_{15} and Q_{19}

receive, in every timeslot, a random number of packets that is uniformly distributed in $[0, 2L]$, while the other queues each receive a packet with probability 0.12, all independently of each other. We set the channel ON probability to $q = 0.4$ to ensure that the system is stable but heavily loaded (about 95.7% for $L = 5$). As can be seen from the plot (Figure 4), the proposed SSG algorithm and the Modified iLQF with PullUp algorithm give very similar performance. In fact, we can hardly distinguish the two curves in each pair. Even if the Y-axis scale is changed from logarithmic to linear, the two algorithms result in almost overlapping curves.

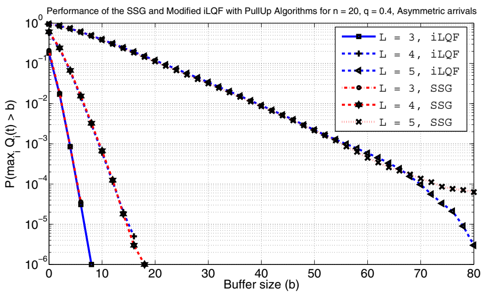

Fig. 4. SSG v/s Modified iLQF with PullUp: Asymmetric arrivals

Due to space constraints, we have not reported the simulation results comparing SSG and Modified iLQF with PullUp for bursty, symmetric arrivals, but the two algorithms continue to give almost identical performance for the system under this and a number of other configurations, and consistently better than MaxWeight. Also, we have compared the performance of the SSG algorithm against the MaxWeight algorithm with a different tie-breaking rule: in every timeslot, the rule allocates servers sequentially (S_1 to S_n). A server S_j is allocated to a longest connected queue that has so far received the least amount of service. While the queue-lengths are not updated after each allocation, this rule results in a minimal wastage of service. Even with this modification, the SSG rule performs much better than the MaxWeight rule. Thus, we have verified that the results presented here are representative. For more simulation results, please see [4].

VII. Conclusion

We considered the problem of designing scheduling algorithms for OFDM-based wireless downlink from the point of view of minimizing per-user delay, which is closely related to having small per-user queues at the base-station. We first considered a class of algorithms called iHLQF and showed that all the algorithms in this class are throughput-optimal for the system. In addition, under certain technical conditions, these algorithms are rate-function optimal for the small buffer overflow event. However, the computational complexity of these algorithms is somewhat large. We then considered the classic MaxWeight algorithm and showed that it results in a very poor small-buffer performance for our system. We proposed a new algorithm called SSG (Server-Side Greedy) that is throughput optimal for the system, results in a positive rate function for the small buffer overflow event under symmetric, i.i.d., ON-OFF traffic, and has a small computational complexity. We verified the results through simulations.

The new intuition that emerges from our work is that maximizing the sum of channel-rate-weighted queue-lengths, although throughput-optimal, is not good for small per-user queues. Instead, approximately maximizing this sum while paying attention to the finer queuing dynamics results in good small-buffer performance in addition to throughput-optimality.

Acknowledgments

This work was partially supported by NSF Grants CNS-0721286, CNS-0347400, CNS-0721380, HDTRA1-09-1-005, and the Darpa ITMANET program.

References

[1] 3GPP TR 25.913. Requirements for Evolved UTRA (E-UTRA) and Evolved UTRAN (E-UTRAN). March 2006.

[2] M. Andrews, K. Kumaran, K. Ramanan, A.L. Stolyar, R. Vijayakumar, and P. Whiting. CDMA data QoS scheduling on the forward link with variable channel conditions. *Bell Labs Tech. Memo*, April 2000.

[3] S. Bodas, S. Shakkottai, L. Ying, and R. Srikant. Scheduling in Multi-Channel Wireless Networks: Rate Function Optimality in the Small-Buffer Regime. In *Proc. Ann. ACM SIGMETRICS Conf.*, Jun. 2009.

[4] S. Bodas, S. Shakkottai, L. Ying, and R. Srikant. SSG Tech Report. http://users.ece.utexas.edu/~bodas/infocom10-long.pdf, 2009.

[5] A. Eryilmaz, R. Srikant, and J. Perkins. Stable scheduling policies for fading wireless channels. *IEEE/ACM Trans. Network.*, 13:411–424, April 2005.

[6] WiMax Forum. Mobile WiMAX Part I: A technical overview and performance evaluation. March 2006. White Paper.

[7] S. Kittipiyakul and T. Javidi. Delay-Optimal Server Allocation in Multi-Queue Multi-Server Systems with Time-Varying Connectivities. *Technical Report, UCSD*, 2008.

[8] P. R. Kumar and Sean P. Meyn. Stability of queueing networks and scheduling policies. *IEEE Trans. Automat. Contr.*, 40:251–260, Feb. 1995.

[9] S.P. Meyn. Stability and asymptotic optimality of generalized maxweight policies. *SIAM J. Control and Optimization*, 2008. to appear.

[10] M. J. Neely. Optimal energy and delay tradeoffs for multiuser wireless downlinks. *IEEE Trans. Inform. Theory*, 53(9):3095–3113, Sept. 2007.

[11] M. J. Neely, E. Modiano, and C. E. Rohrs. Power and server allocation in a multi-beam satellite with time varying channels. In *Proc. IEEE Infocom*, volume 3, pages 1451–1460, New York, NY, June 2002.

[12] S. Shakkottai, R. Srikant, and A. Stolyar. Pathwise optimality of the exponential scheduling rule for wireless channels. *Ann. Appl. Prob.*, 36(4):1021–1045, December 2004.

[13] S. Shakkottai and A. Stolyar. Scheduling for multiple flows sharing a time-varying channel: The exponential rule. *Ann. Math. Statist.*, 207:185–202, 2002.

[14] A. Stolyar. MaxWeight scheduling in a generalized switch: State space collapse and workload minimization in heavy traffic. *Ann. Appl. Prob.*, 14(1), 2004.

[15] A. Stolyar. Large deviations of queues sharing a randomly time-varying server. *Queueing Systems*, 59:1–35, 2008.

[16] L. Tassiulas and A. Ephremides. Stability properties of constrained queueing systems and scheduling policies for maximum throughput in multihop radio networks. *IEEE Trans. Automat. Contr.*, 4:1936–1948, December 1992.

[17] L. Tassiulas and A. Ephremides. Dynamic server allocation to parallel queues with randomly varying connectivity. *IEEE Trans. Inform. Theory*, 39:466–478, March 1993.

[18] V. J. Venkataramanan and X. Lin. Structural properties of LDP for queue-length based wireless scheduling algorithms. In *Proc. Ann. Allerton Conf. Communication, Control and Computing*, Monticello, Illinois, September 2007.

[19] L. Ying, R. Srikant, A. Eryilmaz, and G. Dullerud. A large deviations analysis of scheduling in wireless networks. *IEEE Trans. Inform. Theory*, 52(11):5088–5098, November 2006.