Scheduling in Multi-Channel Wireless Networks: Rate Function Optimality in the Small-Buffer Regime

Shreeshankar Bodas
University of Texas at Austin
Austin, TX 78712, USA
bodas@ece.utexas.edu

Sanjay Shakkottai
University of Texas at Austin
Austin, TX 78712, USA
shakkott@mail.utexas.edu

Lei Ying
Iowa State University
Ames, IA 50011, USA
leiying@iastate.edu

R. Srikant
University of Illinois at
Urbana-Champaign
Urbana, IL 61801, USA
rsrikant@uiuc.edu

ABSTRACT

We consider the problem of designing scheduling algorithms for the downlink of cellular wireless networks where bandwidth is partitioned into tens to hundreds of parallel channels, each of which can be allocated to a possibly different user in each time slot. We prove that a class of algorithms called Iterated Longest Queues First (iLQF) algorithms achieves the smallest buffer overflow probability in an appropriate large deviations sense. The class of iLQF algorithms is quite different from the class of max-weight policies which have been studied extensively in the literature, and it achieves much better performance in the regimes studied in this paper.

Categories and Subject Descriptors

C.2.1 [**Computer-Communication Networks**]: Network Architecture and Design—*Wireless Communication*

General Terms

Algorithms, Performance

Keywords

Scheduling algorithm, large deviations, delay optimality

1. INTRODUCTION

Designing scheduling algorithms is a central problem in wireless networks. In multi-hop wireless networks and the uplink of a cellular network, scheduling is used to resolve contention among competing links while, in the downlink of cellular networks, scheduling is used to achieve maximum throughput subject to Quality of Service (QoS) and fairness constraints. Much of the prior work, with the exception of a few recent papers, has concentrated on achieving 100% throughput without the knowledge of arrival and channel statistics. In this paper, we present scheduling algorithms which achieve the best possible QoS (buffer overflow probability) in the downlink of emerging cellular networks.

We are motivated by the anticipated deployment of 4G systems such as WiMax [6] and LTE [1]. These future systems supporting several tens of users at each base-station employ an OFDM (Orthogonal Frequency Division Multiplexing) based slotted-time air-interface at the base-station. The OFDM air-interface partitions the wireless bandwidth available at the base-station into several hundreds of parallel channels, each of which can be allocated to a (possibly different) user in each timeslot (typically of the order of a few milliseconds). From a network perspective, this system translates into a multi-channel system (with potentially several hundreds of channels), with each channel supporting a *user-dependent* and *time-varying* data rate (user dependence due to the location of the mobile user/handset, and time-variation due to fading and the nature of the wireless channel).

An approach to scheduling over such a system would be to use the MaxWeight algorithm [19]. In each timeslot, the MaxWeight algorithm allocates a single user to each channel based on the product of its queue-length at the base-station (backlog of data that is destined to the mobile user) and the corresponding channel rate. This algorithm has received intense attention [2], and has been shown to be throughput-optimal (i.e., makes the queues stable), along with several performance properties in the large-queue [13, 21, 17, 20] or heavily loaded [16, 14, 10] regimes. However, in a multi-carrier regime with large bandwidth (a scenario that is typically anticipated in 4G systems), one is interested in developing algorithms that ensure small queues at the base-station.

While it is known that the MaxWeight algorithm provides good performance when the queues are large, it is not clear that it provides good small-buffer performance in a multi-carrier setting. For instance, consider a system with 100 channels, each of which can drain one packet per timeslot. Suppose that there are 3 users in the system, with user 1 having 100 packets in its queue and the other two users having 99 packets. It is easy to show that the MaxWeight algorithm will allocate all the available channel resources to user 1. This would result in user 1's queue length decreasing to zero, but the other two queue lengths remaining large. Thus, it intuitively seems better to share the channel resources among all users in order to reduce the peak queue length at the end of the timeslot.

A key observation is that for small-buffer multi-channel systems, scheduling needs to be iterative in each timeslot – as resources (channels) get allocated to users, the effect of this allocation (i.e., that the queue lengths of these users would decrease) needs to factored in when making allocation decisions for the remaining channels. Using this idea, we develop such a class of iterative algorithms (iLQF – iterated Longest Queues First) for scheduling over large multi-carrier systems. We show that for symmetric arrival rates, iLQF algorithms (with certain additional properties) are rate-function optimal in the many-channels regime. Roughly speaking, this means that for a system with a large number of channels (such as a multi-carrier OFDM system), the proposed algorithms "minimize" the probability of the maximum queue length (across users) exceeding *any positive* queue-length threshold b, and where *this threshold b does not scale with system size*. Further, for asymmetric arrival rates (i.e., the arrival rate of each user could be different), a sample-path dominance property established in the paper ensures that the overflow probability under iLQF is upper bounded by a symmetric system whose arrival rate is the same as the largest arrival rate among all the users (please see Section 10).

The main contributions of the paper, along with a summary of the organization of the paper, are provided below:

- In Section 4, we introduce the mathematical abstraction of an OFDM system with many channels, and formally define the problem.
- In Section 5, we present an algorithm-independent lower bound on the probability of a buffer overflow event defined in Section 4.
- In Section 6, we prove certain basic properties of matchings in large bipartite graphs, and exhibit a service rule that is optimal for the problem under consideration in the sense that it achieves the above lower bound in a large deviations sense. However, this service rule results in poor performance when the arrival model is changed even slightly, thus demonstrating the need to carefully design optimal scheduling policies.
- Section 7 presents a class of algorithms called iLQF algorithms and shows that algorithms within this class which possess certain properties are optimal.
- Section 8 describes an algorithm that satisfies the properties required for optimality (described in Section 7) and further, is robust to changes in the arrival model, unlike the algorithm in Section 6.
- In Section 9, we compare the performance of the proposed iLQF class algorithm with the standard MaxWeight algorithm using simulations, and show that the iLQF algorithm yields a much smaller buffer overflow probability than the standard MaxWeight algorithm.
- We conclude with a summary and directions for future work in Section 10.

2. RELATED WORK

Multi-user scheduling in wireless networks has received a lot of interest over the past few years [18, 19, 2, 15, 12, 5, 7]. Recent progress in studying the performance of scheduling algorithms includes the characterizations of the queue-performance in heavy-traffic limits [16, 14, 10], and computations of the tail probability of queue-lengths using the large-deviations analysis [13, 21, 17, 20]. While these results provide very useful insights into the QoS of scheduling algorithms, theoretically, they are valid only when the queue-lengths increase to infinity, i.e., in a large-queue regime. Order-optimality in the number of flows under the MaxWeight algorithm has been explored in [11]. A model similar to the one in this paper has been considered in [8], where the authors use scheduling algorithms based on graph matchings (similar in spirit to the iLQF class of algorithms in this paper) and show delay-optimality in the case of two users, and provide heuristics when more users are present.

To the best of our knowledge, the finite buffer analysis in our paper, for the first time, characterizes the asymptotic buffer overflow performance of OFDM scheduling algorithms in a many-users/servers, small-queue regime.

3. MOTIVATION

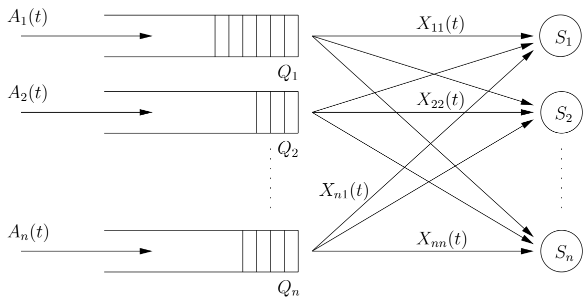


Figure 1: System Model

We consider a discrete time queuing system with n queues and n servers as shown in Figure 1. The following notation is used throughout this paper:

Q_i	$=$	The entity, queue number i
S_i	$=$	The entity, server number i
$Q_i(t)$	$=$	The length of Q_i at the end of timeslot t
$\mathcal{Q}$	$=$	$\{Q_1, Q_2, \ldots, Q_n\}$
$\mathcal{S}$	$=$	$\{S_1, S_2, \ldots, S_n\}$
$A_i(t)$	$=$	The number of arrivals to Q_i at the beginning of timeslot t
$X_{ij}(t)$	$=$	The number of packets in Q_i that can be served by S_j, in timeslot t
a^+	$=$	$\max(a, 0)$

This system model can be used to study an OFDM downlink system (such as WiMax) where each channel (sub-band), consisting of a fixed number of sub-carriers, is a server in Figure 1. There are a fixed number of mobile users, each represented by a queue that corresponds to the backlogged data at the base-station that is destined to the corresponding mobile user. The scheduler operates once every timeslot. During each timeslot, a channel can be assigned to one and at most one user (queue). The state of the channel ($X_{ij}(t)$) to a specific user depends on the location of the user.

Some typical rates (for a 20 MHz WiMax-like system) are as follows: the air-interface is based on OFDM with 50 channels (sub-bands), each of which consists of 25 sub-carriers. Each channel can support 400 kbps and the scheduler operates once every 5 milliseconds. Thus, each good (ON) channel offers 2 kb per timeslot.

Now, the challenge is to develop a high-performance scheduling algorithm for this system. At a first glance, by treating each server as a separate downlink server, the problem is not very different from the scheduling for a traditional downlink network. We can then use the following max-weight scheduling algorithm, which is throughput-optimal.

Max-weight Scheduling: At time slot t, server j serves Q_i^* such that

$$Q_i^* \in \arg\max_{Q_i} X_{ij}(t)Q_i(t). \qquad \diamond$$

While the max-weight scheduling is throughput-optimal, it causes large delays (due to large queues at the base-station). As an example, assume $Q_1(t) = 100, Q_2(t) = Q_3(t) = 95, Q_4(t) = Q_5(t) \ldots = Q_{100}(t) = 10$. Then, all servers S_j such that $X_{1j}(t) = 1$ will serve Q_1. Assume that $X_{ij}(t) = 1$ with probability 0.9, and $X_{ij}(t)$ are mutually independent. Then, roughly 90% of the servers (channels) will be allocated to user '1', and the remaining 10% to users 2 and 3, which will result in large queues for users 2 and 3 at the end of the timeslot.

It can be argued that the MaxWeight algorithm "drives up" all queue lengths to large enough values to ensure the maximum scheduling flexibility. This in turn results in large per-user queue lengths, which can result in large delays. In a multi-carrier system supporting large rates, this problem is further exacerbated because the (mean) queue-lengths under the MaxWeight algorithms grow with the system capacity.

Thus, to have small queues, the first-cut at an algorithm would be to design it in such a way as to serve as many users as possible during each time slot. In Lemma 1, we prove that in a large, balanced, random bipartite graph, a perfect matching (a matching including all queues) exists with high probability. A naive algorithm then is to allocate the channels according to the perfect matching, which we call the perfect-matching scheduling algorithm.

Perfect-matching scheduling: In each time slot, if there exists a perfect matching between the queues and the channels, then serve all queues according to the perfect matching; otherwise, no queue is served. $\diamond$

In Lemma 2, we prove that this perfect-matching scheduling maximizes

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

for $A_i(t) \in \{0,1\}$. However, when $A_i(t) \in \{0,2\}$ (Lemma 3),

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

Hence, the perfect-matching scheduling rule is sensitive to the arrival distribution. This is because the perfect-matching scheduling does not consider queue-length information, and allocates at most one server to a queue.

Thus, a good scheduling algorithm should exploit queue-length information, and allocate an appropriate number of servers to each queue.

In the context of ON-OFF channels, we propose the iLQF (iterated Longest Queues First) class of algorithms. In each timeslot, an algorithm in this class first considers the set of longest queues, and allocates a server to each of them. Then these (used) servers are removed from consideration, and the lengths of the longest-queues are reduced by the amount served. (Note: we have not served any queues at this point, we are simply updating the queue-lengths as though they have been served.) Next, we find the set of the longest queues in the updated system, and allocate one server to each of them. This progresses until we are unable to find a matching between the longest queues and the remaining (unallocated) servers. When formally defining this algorithm, an important issue arises: for a given set of queues, there are many ways to find matching servers (i.e. the matching between the longest queues and available servers may not be unique). Then, *which set of matching servers should we choose during each iteration? Should we explore all possible sets of matching servers?*

Main result: In Section 7, we describe a class of iterative algorithms (iLQF – iterated Longest Queues First) for scheduling over large multi-carrier systems. We show in Theorem 3 that under certain mild conditions, the iLQF algorithms are rate-function optimal in the many-channels regime, i.e. they maximize

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

for any finite threshold $b \ge 0$.

In Section 8, we propose a specific iLQF algorithm that exploits the PullUp technique (to be defined) to efficiently find a good matching. The overall computational complexity of the proposed algorithm is $O(n^4)$.

Finally, we discuss extensions to more general arrival models.

4. SYSTEM MODEL

We consider a multi-channel wireless network as shown in Figure 1. The systems are indexed by the number of servers (and queues), n, and are denoted by Υ_n. For concreteness, in a given timeslot, we assume that arrivals to the queues occur first and then there is the chance for service. The arrivals to each queue are i.i.d. Bernoulli(p), independent across queues and time. In particular,

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

and

$$X_{ij}(t) = \begin{cases} 1 & \text{with probability } q, \\ 0 & \text{with probability } 1-q. \end{cases} \tag{2}$$

All the random variables $A_i(t)$ and $X_{ij}(s)$ are mutually independent. Each queue maintains a buffer of infinite size, so that no packets are ever dropped. If $X_{ij}(t) = 1$, then the server j can potentially serve queue i in timeslot t, reducing the length of queue i by one (unless it is empty). We define the random variables

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

The random variables $Y_{ij}(t)$ are defined by the policy (service rule) that allocates servers to queues. As in an OFDM system, a server can serve at most one queue, but a queue may be served by multiple servers. That is, for all t and all $j \in \{1, 2, \dots, n\}$, we require

$$\sum_{i=1}^{n} Y_{ij}(t) \leq 1.$$

The queue-lengths at the end of a timeslot are defined by the following equation:

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

A finite integer $b \geq 0$ is fixed. The queueing system is started at time $-\infty$. Our objective is to design a service rule that maximizes

$$\liminf_{n \to \infty} \frac{-1}{n} \log \mathbb{P} \left(\max_{1 \leq i \leq n} Q_i(0) > b \right). \tag{3}$$

The above expression is called a rate-function in large deviations theory, and thus our design goal is to find a service rule that is rate-function optimal. We refer to the event $\{\max_i Q_i(t) > b\}$ as the overflow event. We consider only ergodic service policies that make all the queues in the system positive recurrent, so that the probability in (3) is well defined, and equals the fraction of timeslots for which $\{\max_i Q_i(t) > b\}$. Roughly, for large values of n and any fixed b, (3) is equivalent to designing a scheduling policy that results in the *largest* value of $\alpha(b)$ where

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

This means that (for large systems) the algorithm with such a property will result in the smallest buffer overflow probability, for any buffer size b.

5. ALGORITHM-INDEPENDENT LOWER BOUND ON OVERFLOW PROBABILITY

In this section, we present a lower bound on the overflow probability (3). This is an algorithm-independent lower bound, so it holds for any scheduling algorithm. In Section 7, we develop a class of iterative algorithms (iLQF) that achieve this bound.

THEOREM 1. *For the system Υ_n, under any rule for allocating servers to queues, and for all possible values of the parameters $n > 0, 0 < p, q < 1, b \geq 0$,*

$$\mathbb{P} \left(\max_{1 \leq i \leq n} Q_i(0) > b \right) \geq p^{b+1} (1-q)^{n(b+1)}.$$

Consequently, for any $p > 0$,

$$\limsup_{n \to \infty} \frac{-1}{n} \log \mathbb{P} \left(\max_{1 \leq i \leq n} Q_i(0) > b \right) \leq (b+1) \log \frac{1}{1-q}. \tag{4}$$

PROOF. Consider the following event which implies that $\{Q_1(0) > b\}$: for $b+1$ consecutive timeslots before (and including) timeslot 0, there are arrivals to Q_1, and all the channels connecting Q_1 to the servers are OFF in each of the $b+1$ timeslots. The probability of this event is equal to $p^{b+1}((1-q)^n)^{b+1}$ and the result follows. $\square$

6. STABILITY AND PERFECT MATCHINGS

In this section, we first present a stability condition. Then, we study a service rule that is optimal for the problem under consideration but not robust to even small changes in the model.

THEOREM 2. *For given values of $p, q \in (0,1)$, there exists $n_0 = n_0(p,q)$ such that for all $n \geq n_0$, the queuing system Υ_n can be stabilized by some service rule.*

PROOF. Consider a service rule where each server uniformly and randomly picks a queue to which it has an ON channel, and serves it. If that particular chosen queue is empty, then that server does not serve any queue in that timeslot. (Multiple servers can serve the same queue, but there is no co-ordination between the servers.)

Then, the probability that the first server offers its service to the first queue in a particular time slot is

$$\begin{aligned}
&\mathbb{P}(S_1 \text{ offers service to } Q_1 \text{ in timeslot t}) \\
&= \mathbb{P}(S_1 \text{ offers service to } Q_1 \text{ in timeslot t} | X_{11}(t) = 1) \\
&\quad \cdot \mathbb{P}(X_{11}(t) = 1).
\end{aligned}$$

Now, for the service rule under consideration,

$$\begin{aligned}
&\mathbb{P}(S_1 \text{ offers service to } Q_1 \text{ in timeslot t} | X_{11}(t) = 1) \\
&= \sum_{j=0}^{n-1} \mathbb{P}(S_1 \text{ offers service to } Q_1 \text{ in timeslot t} | X_{11}(t) = 1, \\
&\quad \text{Exactly } j \text{ of the rest } n-1 \text{ channels from } S_1 \text{ are ON}) \\
&\quad \cdot \mathbb{P}(\text{Exactly } j \text{ of the rest } n-1 \text{ channels from } S_1 \text{ are ON}) \\
&= \sum_{j=0}^{n-1} \frac{1}{j+1} \binom{n-1}{j} q^j (1-q)^{n-1-j} \\
&= \sum_{j=0}^{n-1} \frac{1}{j+1} \cdot \frac{(n-1)!}{j!(n-1-j)!} q^j (1-q)^{n-1-j} \\
&= \frac{1}{n} \sum_{j=0}^{n-1} \frac{n!}{(j+1)!(n-1-j)!} q^j (1-q)^{n-1-j} \\
&= \frac{1}{qn} \sum_{j=0}^{n-1} \binom{n}{j+1} q^{j+1} (1-q)^{n-1-j} \\
&= \frac{1-(1-q)^n}{qn}.
\end{aligned}$$

Hence, $\mathbb{P}(S_1 \text{ offers service to } Q_1 \text{ in timeslot t}) = \frac{1-(1-q)^n}{n}$, implying that the total service offered to the first queue (or to any other queue, by symmetry) in timeslot t is $1-(1-q)^n$. If $p < 1$ and $q > 0$ are fixed, then $1-(1-q)^n > p$ for large enough $n \geq n_0(p,q)$, where

$$n_0(p,q) := \left\lceil \frac{\log(1-p)}{\log(1-q)} \right\rceil,$$

implying that all the queues are stable (positive recurrent) under the specified policy. $\square$

The above stability result can be generalized easily, assuming $\sup_i p_i < 1$: (a) arrival rates to different queues can be different, (b) the arrival processes can be generalized to allow the number of arrivals to take on values in a finite, non-negative integer set, and (c) the service processes can also be similarly generalized.

We now prove a result regarding perfect matchings in bipartite graphs that is useful for the analysis of the proposed algorithm later in the paper.

Lemma 1. *Consider an undirected bipartite graph $G(\mathcal{U} \cup \mathcal{V}, \mathcal{E})$, where $\mathcal{U} \cup \mathcal{V}$ is the set of vertices with $|\mathcal{U}| = |\mathcal{V}| = n$, and $\mathcal{E}$ is the set of edges. Every edge $e \in \mathcal{E}$ has one of its endpoints in $\mathcal{U}$ and the other in $\mathcal{V}$. For every node $u \in \mathcal{U}$ and $v \in \mathcal{V}$, the edge (u, v) is present in $\mathcal{E}$ with probability q, independently of all other edges. Then, for large n,*

$$(1-q)^n \le \mathbb{P}(G \textit{ has no perfect matching}) \le 3n(1-q)^n,$$

where a perfect matching is defined as a matching of cardinality n.

Proof. For $\mathcal{A} \subseteq \mathcal{U}$, let $\Gamma(\mathcal{A})$ denote the neighborhood $\mathcal{A}$, i.e.

$$\Gamma(\mathcal{A}) := \{b \in \mathcal{V} : (a, b) \in \mathcal{E} \text{ for some } a \in \mathcal{A}\}.$$

We know from Hall's theorem ([9], Thm. 7.40) that if a bipartite graph $G(\mathcal{U} \cup \mathcal{V}, \mathcal{E})$ does not have a perfect matching, then there exists a subset $\mathcal{A} \subseteq \mathcal{U}$ such that $|\Gamma(\mathcal{A})| < |\mathcal{A}|$. Fix a nonempty subset $\mathcal{A} \subseteq \mathcal{U}$ and a subset $\mathcal{B} \subseteq \mathcal{V}$. Let $|\mathcal{A}| = a$. Then, we have

$$\begin{aligned}
&\mathbb{P}(\Gamma(\mathcal{A}) \subseteq \mathcal{B}) \\
&\quad = \mathbb{P}(\text{No node in } \mathcal{A} \text{ connects to any node in } \mathcal{S}\backslash\mathcal{B}) \\
&\quad = (1-q)^{(n-|\mathcal{B}|)a}.
\end{aligned}$$

If the graph has no perfect matching, then by Hall's theorem, there must exist sets $\mathcal{A}$ and $\mathcal{B}$ such that

1. $\mathcal{A} \subseteq \mathcal{U}$, $\mathcal{B} \subseteq \mathcal{V}$,

2. $|\mathcal{B}| = |\mathcal{A}| - 1$,

3. $\Gamma(\mathcal{A}) \subseteq \mathcal{B}$.

Hence, by union bound over all possible subsets $\mathcal{A} \subseteq \mathcal{U}$ and all possible corresponding subsets $\mathcal{B} \subseteq \mathcal{V}$, we have

$$\begin{aligned}
&\mathbb{P}(G \text{ has no perfect matching}) \\
&\quad \le \sum_{a=1}^{n} \binom{n}{a} \cdot \binom{n}{a-1} \cdot (1-q)^{a(n-a+1)} \\
&\quad \le 2 \sum_{a=1}^{\lceil n/2 \rceil} \binom{n}{a} \cdot \binom{n}{a-1} \cdot (1-q)^{a(n-a+1)}, \qquad (5)
\end{aligned}$$

where the last inequality holds with equality if n is even.

We consider the case when n is large, in particular $n > 2$. Now, for $n > 2$ and $1 < a \le \lceil n/2 \rceil$, $a-1 \ge a/2$, $n-a \ge n/3$, and we have

$$\begin{aligned}
&\frac{\binom{n}{a}\binom{n}{a-1}(1-q)^{a(n-a+1)}}{n(1-q)^n} \\
&\le \frac{n^a \cdot n^{a-1} \cdot (1-q)^{a(n-a+1)}}{n(1-q)^n} \\
&\le n^{2a}(1-q)^{(n-a)(a-1)} \\
&\le n^{2a}(1-q)^{na/6} \\
&= \exp\left(2a \log n - \frac{na}{6} \log \frac{1}{1-q}\right) \\
&= \exp\left[-\frac{a}{6}\left\{n \log \frac{1}{1-q} - 12 \log n\right\}\right] \\
&\le \exp\left\{-\frac{a}{6} \cdot \frac{n}{2} \cdot \log \frac{1}{1-q}\right\}, \quad \text{for } n \text{ large enough} \\
&\le \exp\left\{-n \cdot \frac{1}{12} \log \frac{1}{1-q}\right\}, \quad \text{since } a > 1.
\end{aligned}$$

Hence, from (5), we have for any fixed $\epsilon > 0$,

$$\begin{aligned}
&\mathbb{P}(G \text{ has no perfect matching}) \\
&\le 2n(1-q)^n \cdot \left(1 + (\left\lceil \frac{n}{2} \right\rceil - 1) \exp\left\{-n \cdot \frac{1}{12} \log \frac{1}{1-q}\right\}\right) \\
&\le 2n(1-q)^n \cdot (1+\epsilon), \quad \text{for } n \text{ large enough.} \qquad (6)
\end{aligned}$$

Now, fix a node $u_i \in \mathcal{U}$. Let E_i denote the event that u_i is an isolated node. Then, $\mathbb{P}(E_i) = (1-q)^n$. It follows that

$$\mathbb{P}(G \text{ has no perfect matching}) \ge (1-q)^n.$$

Thus, putting $\epsilon = 0.5$ in (6), we have (for large enough n)

$$(1-q)^n \le \mathbb{P}(G \text{ has no perfect matching}) \le 3n(1-q)^n. \quad (7)$$

This completes the proof. □

Next we consider the perfect-matching scheduling.

Definition 1. ***Perfect-matching scheduling:*** *In a timeslot t, let $\mathcal{E} := \{X_{ij}(t) : X_{ij}(t) = 1\}$. If there exists a perfect matching in the bipartite graph $G(\mathcal{Q} \cup \mathcal{S}, \mathcal{E})$, then allocate the servers to serve the respective queues as determined by the perfect matching, else do not allocate any server to the queues.* ◇

Lemma 2. *For the system Υ_n, the perfect-matching scheduling yields*

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

Thus, in conjunction with (4), the perfect matching scheduling rule maximizes (3), and is rate-function optimal.

Proof. Fix the number of queues (and servers), n, large enough for Theorem 1 to hold, and consider the evolution of Q_1 under the above service rule. $Q_1(t)$ evolves according to a Markov chain with the following state-transition probabilities:

$$\begin{aligned}
p_0 = \mathbb{P}(Q_1(t+1) = Q_1(t) + 1) &\le p \cdot 3n(1-q)^n, \\
q_0 = \mathbb{P}(Q_1(t+1) = Q_1(t) - 1 \ge 0) &\ge (1-p)(1 - 3n(1-q)^n), \\
\mathbb{P}(Q_1(t+1) = Q_1(t) + m) &= 0, \qquad (8)
\end{aligned}$$

for all $m \notin \{0, 1, -1\}$. Further, the evolution of Q_1 is independent of the states of, and arrivals to, all the other queues. Figure 2 shows the transition probabilities for $Q_1(t)$.

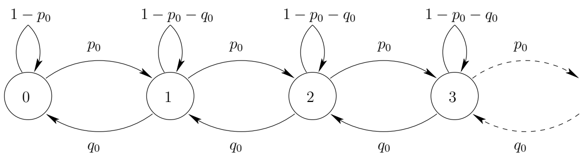


Figure 2: Markov chain for the evolution of the first queue

For $\rho := p_0/q_0 < 1$, the steady-state distribution of the Markov chain in Figure 2 is given by

$$\mathbb{P}(Q_1(t) = b) = (1-\rho)\rho^b, \quad \forall\, b \geq 0,$$

implying $\mathbb{P}(Q_1(t) > b) = \rho^{b+1}$. Using (8), we get

$$\begin{aligned}\mathbb{P}(Q_1(t) > b) &\leq \left(\frac{3pn(1-q)^n}{(1-p)(1-3n(1-q)^n)}\right)^{b+1} \\ &\leq \left(\frac{6pn(1-q)^n}{1-p}\right)^{b+1},\end{aligned}$$

for n large enough. The same calculation applies to each one of the queues from Q_2 to Q_n, since the number of packets served from a queue Q_i is independent of all other queues and their respective arrivals; it is a function of the random variables $X_{jk}(t)$, $Q_i(t-1)$ and $A_i(t)$. Therefore, for the service rule under consideration,

$$\begin{aligned}\mathbb{P}\left(\max_{1\leq i\leq n} Q_i(0) > b\right) &\leq \sum_{i=1}^{n} \mathbb{P}(Q_i(0) > b) \\ &= n\mathbb{P}(Q_1(0) > b) \\ &\leq n\left(\frac{6pn(1-q)^n}{1-p}\right)^{b+1}.\end{aligned}$$

Hence,

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

which, combined with (4), proves that the service rule under consideration maximizes (3). □

While the perfect-matching scheduling is rate-function optimal for $A_i(t) \in \{0,1\}$, this algorithm is sensitive to the arrival processes.

Definition 2. *The arrival process to a queuing system is said to be L×Bernoulli(p) if it satisfies*

$$A_i^{(n)}(t) = \begin{cases} L & \text{with probability } p, \\ 0 & \text{with probability } 1-p, \end{cases}$$

with $pL < 1$. If $L = 1$, then the process is said to be Bernoulli(p). ◇

Lemma 3. *If the arrival process to the system Υ_n is changed from Bernoulli(p) to 2×Bernoulli(r) for any $r \in (0, 0.5)$, then the perfect matching scheduling rule results in the overflow event having at least a constant probability, implying that the expression* (3) *equals* 0.

Proof. Consider the evolution of Q_1, starting from any state $Q_1(t)$. The following event leads to $\{Q_1(t+b+1) > b\}$, irrespective of the channel realizations: for $b+1$ consecutive timeslots $(t+1, \ldots, t+b+1)$, there are arrivals to Q_1. This event leads to $Q_1(t+b+1) > b$, since in a given timeslot, at most 1 packet can be served from any given queue. The probability of this event is r^{b+1}. Therefore, under the perfect matching scheduling rule,

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

taking steps similar to that in the proof of Theorem 1. □

This motivates us to study (in the rest of this paper) a queue-length based scheduling policy which provides rate-function optimality (in Equation (3)) for the Bernoulli(p) arrival process, and also achieves a nonzero rate function for more general arrival processes (see Corollary 2, Section 8.2).

7. CHARACTERISTICS OF OPTIMAL SERVICE RULES

In this section, we consider a special class of service rules - iLQF (iterated Longest Queues First), and present sufficient conditions for an iLQF scheduling policy to be rate-function optimal. In the next section, we present an algorithm in this class that maximizes (3), and also is robust to the arrival processes.

Definition 3. ***Iterated Longest Queues First (iLQF):*** *A service rule is said to belong to the class iLQF if, in every timeslot, it allocates servers to queues in multiple rounds as follows:*

1. *In a given round, the service rule finds a largest cardinality matching in the bipartite graph whose node-sets are the set of longest queues and set of available servers, and the edges are defined by the channel realizations (an edge from Q_i to S_j is present if $X_{ij} = 1$), and allocates the servers to the (longest) queues as determined by the matching. If the cardinality of the matching thus found equals the cardinality of the set of the longest queues, then the algorithm is required to serve all the (longest) queues. If, in the given round, none of the longest queues are connected to any of the servers, then the set of the next longest queues may be considered for server allocation, but it is not required to be considered.*

2. *The service rule updates the lengths of all the queues (to take into account the service received by a subset of the longest queues in the particular round) and the set of available servers (to take into account the servers allocated to some of the queues) and proceeds to the next round.* ◇

Note that the class iLQF contains more than one scheduling algorithm, since the following parameters are unspecified:

1. The number of rounds to be performed, i.e., the termination condition.

2. The tie-breaking rule if there exist multiple largest cardinality matchings among the longest queues.

This class of algorithms is interesting because it gives priority to the longer queues, thereby trying to minimize the probability of the overflow event.

LEMMA 4. *For any algorithm in the iLQF class, and for n large enough,*

$$\mathbb{P}\left(\max_{1\leq i\leq n} Q_i(t+1) > \max_{1\leq i\leq n} Q_i(t)\right) \leq 3n(1-q)^n.$$

PROOF. Consider the bipartite graph $G(\mathcal{Q}\cup\mathcal{S},\mathcal{E})$, where $\mathcal{E} := \{X_{ij}(t) : X_{ij}(t) = 1\}$. If G has a perfect matching (i.e., a matching of cardinality n), then for an algorithm in the *iLQF* class, $\max_{1\leq i\leq n} Q_i(t+1) \leq \max_{1\leq i\leq n} Q_i(t)$. Further, by Theorem 1, the graph G has a perfect matching with probability at least $1 - 3n(1-q)^n$ for large n. □

DEFINITION 4. ***Dominance property of an iLQF rule*** Λ***:*** *Consider the queuing system with $\mathcal{Q} = \{Q_i\}_{i=1}^n$ as the queues, and $\mathcal{S} = \{S_i\}_{i=1}^n$ as the servers. Let $A_i(t)$ and $X_{ij}(t)$ be the arrival process and channel processes respectively (see (1), (2)). Now, a new queueing system with queues $\mathcal{R} = \{R_i\}_{i=1}^n$ and servers $\mathcal{S} = \{S_i\}_{i=1}^n$ is obtained as follows: at each time t, the queues $R_i, i = 1, 2, \ldots, n$ see the same arrivals as those incoming to $Q_i, i = 1, 2, \ldots, n$ and the channel states of the servers are identical to those of system $\mathcal{Q}$ (i.e. the arrival processes and channel states in the system $\mathcal{R}$ are sample-path coupled with the system $\mathcal{Q}$).* In addition, *there are extra packet arrivals (an arbitrary, finite number) that occur to system $\mathcal{R}$ immediately after service, and at arbitrary timeslots $T_1, T_2, \ldots$ (see Figure 3). The service policy used in the queuing system $\mathcal{R}$ is the same iLQF policy (Λ) that is used in the system $\mathcal{Q}$ (also the process $\mathcal{R}$ is defined over the same probability space as $\mathcal{Q}$).*

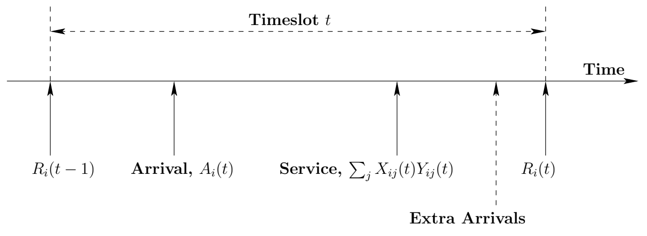


Figure 3: Service model for the queuing system $\mathcal{R}$

A rule Λ in the iLQF class is said to have the dominance property if the following holds: for all timeslots t, all $b \geq 0$, and over all possible choices of extra arrivals, we have that

$$\mathbb{P}\left(\max_{1\leq i\leq n} R_i(t) > b\right) \geq \mathbb{P}\left(\max_{1\leq i\leq n} Q_i(t) > b\right). \qquad \diamond$$

Intuitively, the dominance property requires that *adding extra packets* to the queueing system driven by the iLQF policy Λ does *not decrease* the maximum queue length. This property is extremely useful, because this property allows us to "carefully" add packets so that the resulting queuing system can be explicitly analyzed and whose rate function can be computed in closed-form. The dominance property ensures that the rate function so obtained provides a lower-bound on the rate-function of the original system.

DEFINITION 5. ***Drain property of a scheduling rule*** Λ***:*** *A scheduling rule Λ (not necessarily from the iLQF-class) is said to have the drain property if there exists a constant k_0 independent of n such that, for all n large enough and all integers t,*

$$\mathbb{P}\left(\max_{1\leq i\leq n} Q_i(t+k_0) < \max_{1\leq i\leq n} Q_i(t) \,\middle|\, \max_{1\leq i\leq n} Q_i(t) > 0\right) \geq \frac{1}{2}. \qquad \diamond$$

THEOREM 3. *Suppose a service rule in the iLQF class has the drain and dominance properties. Then, this iLQF service rule results in*

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

Further, by Theorem 1, no other service rule can give a larger value for the left hand side of the above expression.

PROOF. The proof of the theorem proceeds according the following steps:

1. By adding dummy packets and using the drain property, we construct a Markov chain $\underline{\tilde{B}}(t)$ such that for some $k_0 > 0$,

$$\begin{aligned}
\mathbb{P}(\tilde{B}^\star(t+1) = m-1 | \tilde{B}^\star(t) = m) &= \frac{1}{2} \\
\mathbb{P}(\tilde{B}^\star(t+1) = m+r | \tilde{B}^\star(t) = m) &= \binom{k_0}{r}(3n(1-q)^n)^r \\
&\quad \text{for } r = 1, \ldots, k_0 \\
\mathbb{P}(\tilde{B}^\star(t+1) > m+k_0 | \tilde{B}^\star(t) = m) &= 0.
\end{aligned}$$

2. We then compute the bounds on the stationary distribution of the new Markov chain, and show that

$$\liminf_{n\to\infty} \frac{-1}{n} \log \mathbb{P}(\tilde{B}^\star(0) > b) \geq (b+1)\log\frac{1}{1-q}.$$

   Under the dominance property, this is also a lower bound on the rate function of the original Markov chain. Finally, according to Theorem 1, we can conclude that this is the rate-function for iLQF algorithms with the drain and stochastic dominance properties.

The details of the proof are provided in [3]. □

8. A SPECIFIC ALGORITHM

We now focus on constructing an algorithm in the iLQF class that satisfies the requirements in the statement of Theorem 3. The algorithm employs a particular tie-breaking rule (PullUp) when there exist multiple largest-cardinality matchings in the bipartite graph between the set of queues and servers, where the edges are defined by the ON links.

Before we explain the intuition behind this tie-breaking rule, consider two queuing systems denoted by $\mathcal{Q}$ and $\mathcal{R}$ with both these systems operating under the same iLQF rule. Further, suppose that at some timeslot along a fixed sample-path, the set of longest-queues under $\mathcal{Q}$ is a subset of the set of longest queues under $\mathcal{R}$ and the set of available channels in system $\mathcal{Q}$ is "larger" than that in system $\mathcal{R}$ (more precisely, the bipartite graph connecting the queues to the servers in system $\mathcal{Q}$ has more servers and edges than in the system $\mathcal{R}$, where ordering is defined by set inclusion). This is a scenario where system $\mathcal{R}$ is less "flexible" than system $\mathcal{Q}$ (in terms of scheduling flexibility) in the sense that any allocation of servers in the system $\mathcal{R}$ can be mimicked by system $\mathcal{Q}$.

Now, the intuition behind PullUp can be explained as follows. Consider two multichannel queueing systems with identical initial conditions and suppose we add packets at arbitrary times to one of them (say, the second system). Then, we would like the first system to have more "flexibility" at each time slot under iLQF in the sense of the previous paragraph. (We will see that such a property is key to showing the stochastic dominance in Theorem 3.) The PullUp-based iLQF algorithm described below ensures that such a property holds.

Definition 6. ***PullUp:*** *Consider a bipartite graph $G(\mathcal{U} \cup \mathcal{V}, \mathcal{E})$, where the sets of nodes, not necessarily of the same cardinality, are $\mathcal{U} = \{u_1, u_2, \ldots, u_m\}$ and $\mathcal{V} = \{v_1, v_2, \ldots, v_n\}$. Given a matching $\mathcal{M}$ in G, $\mathcal{M}' := PullUp(G, \mathcal{M}, \mathcal{V})$ is a new matching obtained by the following steps, which we call PullUp:*

1. *Mark all the edges in $\mathcal{M}$ as forward edges (i.e. from $\mathcal{U}$ to $\mathcal{V}$), and all the other edges in $\mathcal{E}$ as backward edges, to get a directed graph G_1. Define $\mathcal{M}_1 := \mathcal{M}$.*

2. *Obtain $\mathcal{M}_{k+1}$ from $\mathcal{M}_k$ as follows: If the node v_k has an incoming edge, then define $\mathcal{M}_{k+1} := \mathcal{M}_k$, $G_{k+1} := G_k$. Otherwise, in the directed graph G_k, find the set $\mathcal{N}_k$ of all nodes reachable from v_k. Let $\Gamma(G_k, v_k) := \mathcal{N}_k \cap \mathcal{U}$ and $\Delta(G_k, v_k) := \mathcal{N}_k \cap \mathcal{V}$. Find the smallest index $l > k$ such that $v_l \in \Delta(G_k, v_k)$. If no such l exists, then define $\mathcal{M}_{k+1} := \mathcal{M}_k$ and $G_{k+1} := G_k$. If such an l exists, then reverse the directions of all the edges on a path from v_k to v_l, to obtain a graph G_{k+1}. Define $\mathcal{M}_{k+1}$ to be the set of all forward edges in G_{k+1}.*

3. *Return the matching $\mathcal{M}' := \mathcal{M}_{n+1}$.* ◇

An example of the PullUp operation is shown in Figure 4.

Lemma 5. *The output $\mathcal{M}'$ of $PullUp(G, \mathcal{M}, \mathcal{V})$ is a matching, and $|\mathcal{M}| = |\mathcal{M}'|$.*

Proof. Please see [3]. □

The objective of the PullUp operation is to efficiently find a good matching. Based on the PullUp technique, we can construct an iLQF algorithm that is rate-function optimal.

Definition 7. ***iLQF with PullUp:***
Input:

1. *The queue lengths, $Q_1(t-1), Q_2(t-1), \ldots, Q_n(t-1)$.*

2. *The channel realizations, $X_{ij}(t)$ for $1 \le i, j \le n$.*

3. *The arrivals to the queues, $A_i(t)$ for $1 \le i \le n$.*

Steps:

1. *Update the queue-lengths $Q_i(t-1)$ to account for arrivals, that is, the new length of Q_i is defined to be $Q_i(t-1) + A_i(t)$. Hereafter, the length of a queue always refers to its most current updated length, accounting for arrivals and service. Find the length of the longest queue, $\hat{Q}$. Define $L = \hat{Q}$. To begin with, all servers are unallocated.*

2. *Let $\mathcal{Q}_L$ denote the set of queues whose length is exactly L. Let G_L denote the (undirected) bipartite graph with nodes $\mathcal{Q}_L \cup \mathcal{S}$, and the edges as defined by the channel realizations. Here, $\mathcal{S}$ denotes the set of unallocated servers. More specifically, an edge (Q_i, S_j) is present in G_L if $Q_i \in \mathcal{Q}_L$, $S_j \in \mathcal{S}$ and $X_{ij} = 1$. Find a largest cardinality matching $\mathcal{M}_L$ in G_L.*

   (a) If $|\mathcal{M}_L| = |\mathcal{Q}_L|$, define $\mathcal{M}' := PullUp(G_L, \mathcal{M}_L, \mathcal{S})$.

   (b) If $|\mathcal{M}_L| < |\mathcal{Q}_L|$, define $\mathcal{M}_1 := PullUp(G_L, \mathcal{M}_L, \mathcal{S})$. Obtain $\mathcal{M}_{k+1}$ from $\mathcal{M}_k$ as follows: if k is odd, then define $\mathcal{T} := \mathcal{Q}_L$; otherwise $\mathcal{T} := \mathcal{S}$, and $\mathcal{M}_{k+1} := PullUp(G_L, \mathcal{M}_k, \mathcal{T})$. Continue obtaining $\mathcal{M}_{k+1}$ from $\mathcal{M}_k$ until $\mathcal{M}_{i+1} = \mathcal{M}_i$ for some i. Define $\mathcal{M}' := \mathcal{M}_i$.

   Finally, as defined by the matching $\mathcal{M}'$, allocate the servers to queues. For example, if $(Q_x, S_y) \in \mathcal{M}'$, then allocate S_y to serve Q_x, remove S_y from $\mathcal{S}$, decrease the length Q_x by 1.

3. *If at the end of step 2, we have $|\mathcal{M}_L| < |\mathcal{Q}_L|$, then stop. If $|\mathcal{S}| = 0$ or $L = 1$, then stop. Else, decrease the value of L by 1, go to step 2.* ◇

The above description of the algorithm may be a bit difficult to follow. So, we provide a brief description in words here: the algorithm first finds a largest cardinality matching in the bipartite graph consisting of the longest queues and all servers connected to these queues. Then, it performs the PullUp operation on this matching. If the number of links in the resultant matching is less than the number of longest queues, the algorithm terminates after using this matching in the schedule. Else, it removes packets from the longest queues as dictated by the matching and repeats the process by finding a largest cardinality matching among the new set of longest queues. We note that for implementing the iLQF with PullUp algorithm, the base station does not need to know (nor learn) the arrival or channel process statistics.

Let every execution of step 2 be called a round. If in step 2 we have $|\mathcal{M}_L| = |\mathcal{Q}_L|$, then that round is called a perfect matching round, else a maximal matching round.

Theorem 4. *The iLQF with PullUp is rate-function optimal, i.e., it gives*

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

Furthermore, the algorithm can be implemented in $O(n^4)$ computations per timeslot.

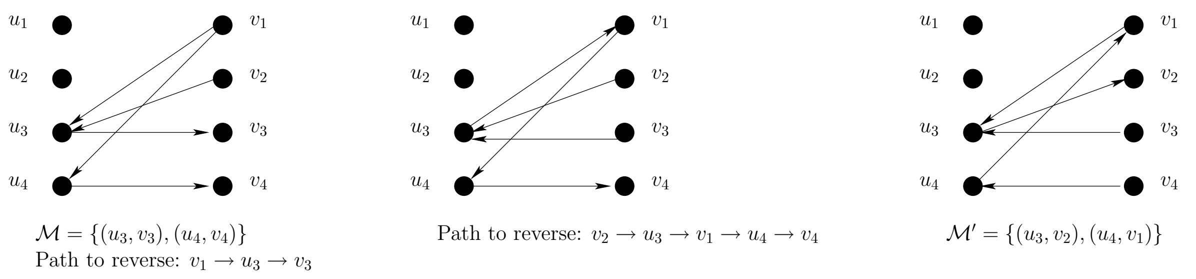
Proof. We prove that the iLQF with PullUp satisfies the drain property (Theorem 6) and the dominance property (Theorem 5). The first part of the theorem holds according to Theorem 3. The computational complexity result follows from Lemma 6. □

8.1 Computational complexity

We first analyze the computational complexity of the iLQF with PullUp.

Lemma 6. *The proposed algorithm can be implemented in $O(n^4)$ computations per timeslot.*

Proof. Please see [3]. □



u_1
u_2
u_3
u_4
v_1
v_2
v_3
v_4
$\mathcal{M} = \{(u_3, v_3), (u_4, v_4)\}$
Path to reverse: $v_1 \to u_3 \to v_3$
Path to reverse: $v_2 \to u_3 \to v_1 \to u_4 \to v_4$
$\mathcal{M}' = \{(u_3, v_2), (u_4, v_1)\}$

Figure 4: An example of the PullUp operation

8.2 Rate-function optimality

We establish the rate-function optimality of the iLQF with PullUp by proving that the algorithm has the drain property and the dominance property as required by Theorem 3. The following is a technical lemma that will be used in the proof of Theorem 5.

LEMMA 7. *In the graph G_{n+1}, if a node v_a has no incoming edge, then there does not exist a (directed) path from v_a to any node v_b with $b > a$. Consequently, if PullUp$(G, \mathcal{M}, \mathcal{V}) = \mathcal{M}'$, then PullUp$(G, \mathcal{M}', \mathcal{V}) = \mathcal{M}'$.*

PROOF. Please see [3]. □

THEOREM 5. *(Sample-path-wise Dominance) Consider two queuing systems $\underline{Q}$ and $\underline{R}$ with queues $\mathcal{Q} = \{Q_1, Q_2, \ldots, Q_n\}$ and $\mathcal{R} = \{R_1, R_2, \ldots, R_n\}$ respectively, with the property that $Q_i(t-1) \le R_i(t-1)$ for all i, for some t. Let the two systems have identical channel realizations, $X_{ij}(t)$ and identical arrivals, $A_i(t)$ for $1 \le i, j \le n$. Both the queuing systems implement the algorithm described in Section 8, i.e. iLQF with PullUp. Then, $Q_i(t) \le R_i(t)$ for all i.*

Note that this theorem immediately implies that the iLQF with PullUp algorithm has the dominance property as required by Theorem 3.

PROOF. The following notation will be used throughout this proof:

$\mathcal{M}_r$	=	The set of queues served in the r^{th} round, in the system $\underline{R}$
$\mathcal{Y}_r$	=	The set of servers allocated in the r^{th} round, in the system $\underline{R}$
$R_i^{(r)}$	=	The length of R_i after r rounds of service
$\mathcal{N}_r$	=	The set of queues served in the r^{th} round, in the system $\underline{Q}$
$\mathcal{Z}_r$	=	The set of servers allocated in the r^{th} round, in the system $\underline{Q}$
$Q_i^{(r)}$	=	The length of Q_i after r rounds of service

By definition, $R_i^{(0)} := R_i(t-1) + A_i(t)$ and $Q_i^{(0)} := Q_i(t-1) + A_i(t)$. Let $\hat{R} := \max_i R_i^{(0)}$, $\hat{Q} := \max_i Q_i^{(0)}$ and $w := \hat{R} - \hat{Q}$. Let there exist n_R and n_Q rounds of perfect matchings in the system $\underline{R}$ and $\underline{Q}$ respectively.

Case 1: $n_R < w$.

If a queue R_i was served even once in the n_R rounds, then at the end of n_R rounds, $R_i^{(n_R)} = \hat{R} - n_R > \hat{R} - w = \hat{Q}$. Since there are exactly n_R rounds of perfect matching in the system $\underline{R}$,

$$R_i(t) \ge R_i^{(n_R)} - 1 \ge \hat{Q} \ge Q_i(t).$$

If R_i was not served even once in the first n_R rounds of perfect matching, but was served in the last round of maximal matching, then

$$R_i(t) = \hat{R} - (n_R + 1) \ge \hat{R} - w = \hat{Q} \ge Q_i(t).$$

Finally, if the queue R_i was not served at all, then

$$R_i(t) = R_i(t-1) + A_i(t) \ge Q_i(t-1) + A_i(t) \ge Q_i(t),$$

and the claim is true in this case.

Case 2: $n_R = w$.

We have $R_i^{(n_R)} \ge \hat{R} - n_R = \hat{Q}$, with equality holding if and only if $R_i^{(0)} \ge \hat{Q}$. Let $\mathcal{R}_{last} = \{R_{i_1}, R_{i_2}, \ldots, R_{i_a}\}$ denote the set of longest (i.e. of length $\hat{Q}$) queues at the beginning of the maximal matching round for the system $\underline{R}$, with $i_1 < i_2 < \cdots < i_a$. Let $\mathcal{Q}_{first} = \{Q_{j_1}, Q_{j_2}, \ldots, Q_{j_b}\}$ denote the set of longest queues in the system $\underline{Q}$, at the beginning of the first round, with $j_1 < j_2 < \cdots < j_b$. Then, $\{j_1, j_2, \ldots, j_b\} \subseteq \{i_1, i_2, \ldots, i_a\}$. If the first round in the system $\underline{Q}$ is a perfect matching round (i.e. $n_Q > 0$), then all of the queues in $\mathcal{Q}_{first}$ are served, and only some of $\mathcal{R}_{last}$, and the claim is true because the queues in the system $\underline{R}$ are not served for more than $n_R + 1$ rounds.

Now, let $n_Q = 0$. Let a queue R_{i_c} be served by a server S_a in the $(n_R + 1)^{th}$ round, but Q_{i_c} is not served in the 1^{st} (largest matching) round. Then, S_a must serve a queue Q_{i_d} with $d < c$, otherwise the size of the largest matching can be strictly increased ($\because X_{i_c a} = 1$), or there exists a directed path $Q_{i_c} \to S_a \to Q_{i_d}$, contradicting Lemma 7. The queue R_{i_d} must be served by a server S_e, otherwise there exists a directed path $R_{i_d} \to S_d \to R_{i_c}$, again contradicting Lemma 7. The server S_e must serve a queue Q_{i_f} with $f < c$, otherwise the size of the largest matching in $\underline{Q}$ can be strictly increased (by allocating S_e to Q_{i_d}, S_a to Q_{i_c}), or there exists a directed path $Q_{i_c} \to S_a \to Q_{i_d} \to S_e \to Q_{i_f}$ and $f > c$, contradicting the specifications of the algorithm and in particular, Lemma 7. This process of finding newer servers and queues in the two systems can be continued indefinitely, contradicting the finiteness of the number of queues and servers in the system. Therefore, if a queue R_{i_c} is served in the largest matching round of the system $\underline{R}$, then so is Q_{i_c} in the system $\underline{Q}$, and the claim holds in this case.

Case 3: $n_R > w$.

We prove the following statement $f(r)$, for $0 \le r \le n_R - w$, by induction:

$$f(r) : \mathcal{N}_r \subseteq \bigcup_{j=1}^{r+w} \mathcal{M}_j, \mathcal{Z}_r \subseteq \bigcup_{j=1}^{r+w} \mathcal{Y}_j, \text{ and } Q_i^{(r)} \le R_i^{(r+w)} \; \forall i.$$

Base case: We need to prove that $f(0)$ is true. Since $\mathcal{N}_0 = \emptyset$ and $\mathcal{Z}_0 = \emptyset$, we only need to prove that $Q_i^{(0)} \le R_i^{(w)}$.

Now, if R_i was not served during the first w rounds, then $R_i^{(w)} = R_i^{(0)} \geq R_i^{(0)}$. If R_i was served in at least one of the first w rounds of service, then $R_i^{(w)} \geq \hat{R} - w = \hat{Q} \geq Q_i^{(0)}$. Hence, $f(0)$ is true.

Induction step: Suppose $f(0), \ldots, f(r-1)$ are true for some $r \geq 1$. We need to prove $f(r)$. Let $R_i \in \mathcal{M}_{r+w}$. We prove that if $R_i^{(r-1+w)} = Q_i^{(r-1)}$, then $Q_i \in \mathcal{N}_r$. Since $R_i \in \mathcal{M}_{r+w}$, it was, at the beginning of that round, a longest queue.

Let $R_i \in \mathcal{M}_{r+w}$ be allocated a server S_a in the $(r+w)^{th}$ round. Therefore, $X_{ia} = 1$. Since (by induction hypothesis)

$$\bigcup_{i=1}^{r-1} \mathcal{Z}_i \subseteq \bigcup_{i=1}^{r-1+w} \mathcal{Y}_i, \text{ and } S_a \notin \bigcup_{i=1}^{r-1+w} \mathcal{Y}_i,$$

we have $S_a \notin \bigcup_{i=1}^{r-1} \mathcal{Z}_i$, so the server S_a is available to serve Q_i in the r^{th} round. Therefore, if there exists a perfect matching in the system $\underline{R}$ in the $(r+w)^{th}$ round, then there exists a perfect matching in the r^{th} round in the system $\underline{Q}$, and $Q_i \in \mathcal{N}_r$, implying that $Q_i^{(r)} \leq R_i^{(r+w)}$.

Now, for the purpose of obtaining a contradiction, let $S_c \in \mathcal{Z}_r$, and $S_c \notin \mathcal{Y}_1 \cup \cdots \cup \mathcal{Y}_{r+w}$. Let Q_i be served by S_c in the r^{th} round, while R_i was served by S_d in $(r+w)^{th}$ round. Hence, $d < c$. S_d must serve some queue Q_e in the system $\underline{Q}$ in r^{th} round, because otherwise it can replace S_c to serve $\underline{Q}_i$ and the server S_d was unused (in the system $\underline{Q}$) until the beginning of the r^{th} round by induction hypothesis. R_e, in turn, must be served by a server S_f in the $(r+w)^{th}$ round in the system $\underline{R}$. We must have $f < c$, otherwise there exists a connecting path $S_c \rightarrow R_i \rightarrow S_d \rightarrow R_e \rightarrow S_f$ and S_c cannot remain unused in the system $\underline{R}$, according to Lemma 7. This process can be continued indefinitely, contradicting the fact that the number of queues and servers is finite. Hence, $\mathcal{Z}_r \subseteq \mathcal{Y}_1 \cup \mathcal{Y}_2 \cup \cdots \cup \mathcal{Y}_{r+w}$, and the induction is complete.

Hence, if we compare the state of the system $\underline{R}$ after n_R rounds of perfect matching (i.e. at the beginning of the maximal matching round) and $\underline{Q}$ at the end of $n_Q - w$ rounds of perfect matching, we have the following:

1. The set of unallocated servers available in the system $\underline{Q}$ is a superset of the set of unallocated servers available in the system $\underline{R}$.

2. The set of longest queues in the system $\underline{Q}$ is a subset of the set of longest queues in the system $\underline{R}$.

As before, let $\mathcal{R}_{last} = \{R_{i_1}, R_{i_2}, \ldots, R_{i_a}\}$ denote the set of longest queues at the beginning of the maximal matching round for the system $\underline{R}$, with $i_1 < i_2 < \cdots < i_a$. Let $\mathcal{Q}_{first} = \{Q_{j_1}, Q_{j_2}, \ldots, Q_{j_b}\}$ denote the set of longest queues in the system $\underline{Q}$, at the beginning of the $(n_R - w + 1)^{th}$ round, with $j_1 < j_2 < \cdots < j_b$. Then, $\{j_1, j_2, \ldots, j_b\} \subseteq \{i_1, i_2, \ldots, i_a\}$. If the $(n_R - w + 1)^{th}$ round in the system $\underline{Q}$ is a perfect matching round (i.e. $n_Q > n_R - w$), then all of the queues in $\mathcal{Q}_{first}$ are served, and only some of $\mathcal{R}_{last}$, and the claim is true because the queues in the system $\underline{R}$ are not served for more than $n_R + 1$ rounds.

Now, let $n_Q = n_R - w$. We need to prove that if a queue R_i is served in the largest matching round of the system $\underline{R}$, then so is Q_i in the system $\underline{Q}$. The proof is almost identical to that of the case $n_R = w$, and is skipped to avoid repetition. Therefore, the proof of the theorem is complete. □

Corollary 1. *The iLQF with PullUp algorithm has the dominance property defined in Section 7.*

The corollary follows by repeated applications of Theorem 5. The queuing system is started at time $-\infty$, and we are interested in the probability that the length of the longest queue exceeds a constant b at a finite time t. By applying the result of Theorem 5 to timeslots $T_1, T_2, \ldots$ (in the definition of the Dominance property), it follows that the packet-added system has sample-path wise longer queues than the original system. The probabilistic dominance is an immediate consequence of this sample-path dominance.

We now demonstrate a property of the PullUp operation which is useful in proving that the proposed algorithm has the Drain property as required by Theorem 3.

Lemma 8. *Let a bipartite graph $G(\mathcal{U} \cup \mathcal{V}, \mathcal{E})$ and a matching $\mathcal{M}$ be given, with*

$$\mathcal{U} = \{u_1, u_2, \ldots, u_n\}, \mathcal{V} = \{v_1, v_2, \ldots, v_n\}.$$

Suppose there exists a matching $\mathcal{M}_\star$ in G with the following properties:

1. *$|\mathcal{M}| = |\mathcal{M}_\star|$.*

2. *If $u \in \mathcal{U}$ is an endpoint of some edge $e \in \mathcal{M}$, then u is an endpoint of some edge $e' \in \mathcal{M}_\star$.*

3. *Mark all the edges in $\mathcal{M}_\star$ as forward edges (i.e. from $\mathcal{U}$ to $\mathcal{V}$), and all the edges in $\mathcal{E} \backslash \mathcal{M}_\star$ as backward edges, to get a directed graph $G^\ddagger(\mathcal{U} \cup \mathcal{V}, \mathcal{E})$. Then, in the graph $G^\ddagger$, if a node $v_i \in \mathcal{V}$ has no incoming edge, then there does not exist a directed path from v_i to any v_j, $j > i$.*

4. *For some $a \leq n$, no node $v_b \in \mathcal{V}$, $b > a$ is an endpoint of any edge in $\mathcal{M}_\star$.*

Let $\mathcal{M}' = PullUp(G, \mathcal{M}, \mathcal{V})$. Then, $\mathcal{M}'$ does not have, as an endpoint of some edge, any node in $\mathcal{V}$ with index larger than a.

Proof. Please see [3]. □

Let ℓ_T denote the length of the longest queue at the end of the timeslot T. We next prove that the iLQF with PullUp satisfies the drain property.

Theorem 6. *(The Drain property) For the proposed algorithm, there exists a constant $k = k(p) = \left\lceil \frac{3}{1-p} \right\rceil$ such that, for all n large enough, all $m > 0$ and all T,*

$$\mathbb{P}(\ell_{T+k} < m | \ell_T = m) \geq \frac{1}{2}.$$

Proof. We provide a sketch of the proof here. WLG let $T = 0$ and $\ell_0 = m$ in a queuing system $\underline{Q}$. Consider a queuing system $\underline{Q'}$ where $Q'_i(0) = m$ for all i, implying $Q'_i(0) \geq Q_i(0)$ for all i, and this property continues to hold for all further timeslots if the arrivals and the channel realizations are identical for the two systems (Theorem 5). We analyze the system $\underline{Q'}$.

Fix $\tilde{p} \in (p, 1)$. The probability that there are $n\tilde{p}$ or more arrivals to the system in a given timeslot is very small (Sanov's Theorem, Thm. 2.10 in [4]). By union bound, the same is true for k timeslots for any constant k independent of n. We condition the rest of the argument on this (high probability) event. Further, if necessary, we add dummy packets to the system to ensure that in each of the k timeslots, the number of queues that see arrivals is *exactly* $n\tilde{p}$.

Let the queue-length distribution at the beginning of a timeslot be:

Queue-length	Number of queues
m	x
$m-1$	$n-x$

Then, at the end of the timeslot, w.h.p., either all queues are shorter than m, or the queue-length distribution is

Queue-length	Number of queues
m	x'
$m-1$	$n-x'$

where $x - x' \geq n\delta$ for some δ. This is because of the following: after arrivals in the timeslot, once the queues of length $m+1$ (if any) are served in the first round, at most x servers are consumed, and there are at least $n-x$ servers available to serve the queues of length m. By Lemma 8, these $n-x$ servers and the remaining queues of length m exhibit a matching independently of the allocations in the first round of service. As a result, if the length of the longest queue equals m at the end of the timeslot, then (w.h.p.) the difference between the number of packet arrivals and packets served is at least a constant fraction of n, providing the negative drift. For a detailed proof, please see [3]. $\square$

Recall that the arrival model used in the proofs are i.i.d. Bernoulli ON-OFF processes. If the arrival process is generalized to any ON-OFF bursty i.i.d. process (i.e., taking values on $\{0, L\}$ for any fixed positive integer L and with ON probability equal to p) and subject to stability (i.e., $pL < 1$), the proofs presented in this paper can be generalized to show that there is a strictly positive rate-function for any $b \geq 0$, as summarized below.

COROLLARY 2. *For ON-OFF bursty i.i.d. (across time and users) arrival processes with $pL < 1$, the iLQF with PullUp algorithm results in a strictly positive value for (3). In other words, for any $b \geq 0$,*

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

We refer to [3] for the proof details.

9. SIMULATION RESULTS

We have compared the performance of the proposed iLQF-class algorithms with the standard MaxWeight (MW) algorithm [19] under a number of conditions. We have considered a system with $n = 20$ queues and 20 servers, with the channel between a queue and a server being *ON* with probability $q = 0.4$. We have run the simulations for 500000 timeslots, based on which the empirical probabilities that the maximum queue-length exceeds a constant b are computed.

In the first set of simulations (Figure 5), we have run the algorithms for a system Υ_n described in Section 4, with $\{0, 1\}$ i.i.d. arrivals.

In the second set of simulations (Figure 6), we study bursty arrivals – in a given timeslot, every queue sees either 0 or 4 arrivals. The reason the iLQF algorithm results in a small probability of buffer overflow from $b = 2$ onwards is that the rate function for bursty arrivals is smaller than that for the $\{0, 1\}$ arrivals.

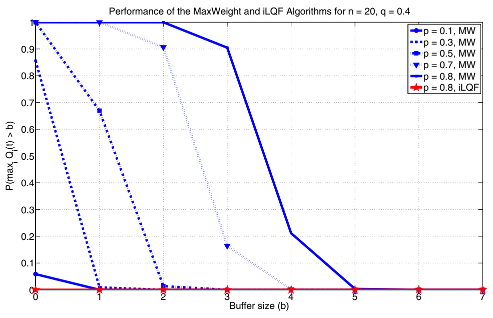


Figure 5: Arrivals as per the system model, Υ_n

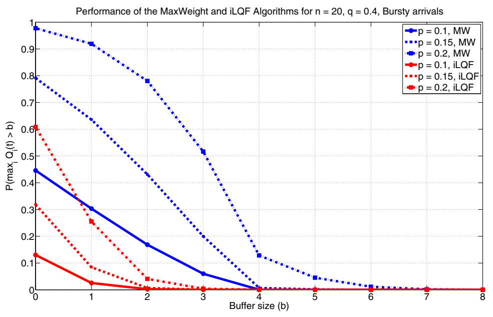


Figure 6: Bursty arrivals

In the third set of simulations (Figure 7), we have considered time-correlated $\{0, 1\}$ arrivals to the queues. We considered an arrival process that formed a Markov chain, with the following transition probabilities (for different values of the parameter p_0):

$$\begin{aligned}\mathbb{P}(A_i(t) = 1 | A_i(t-1) = 0) &= p_0 \\ \mathbb{P}(A_i(t) = 1 | A_i(t-1) = 1) &= 0.8\end{aligned}$$

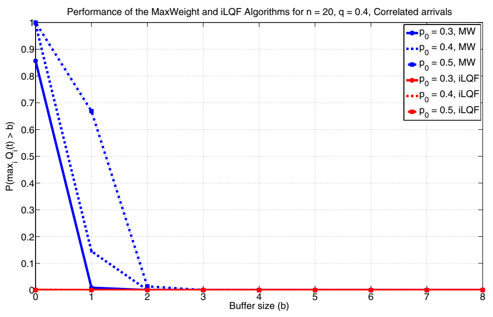


Figure 7: Correlated arrivals

The results are summarized in the accompanying plots. As can be seen, the iLQF algorithm performs much better than the MaxWeight algorithm as far as the finite buffer overflow probabilities are concerned. The intuition for this is that iLQF balances the servers among the long-queues, whereas the traditional MaxWeight focuses on a single longest-queue. When the buffers are large, this does not affect stability. However, for small buffer performance, there is a marked improvement as seen from the plots.

10. DISCUSSION AND CONCLUSIONS

We considered the problem of designing scheduling algorithms for the downlink of cellular networks where the number of available channels is large. Our goal was to minimize the buffer overflow probability in an appropriate large-deviations sense. We showed that a class of algorithms called iLQF minimizes the buffer overflow probability if the algorithm satisfies certain properties. We identified one iLQF algorithm that possesses the desired properties.

In some special cases, the set of algorithms which minimize the probability of overflow may not be singleton. However, we provided an example to show that not all optimal algorithms have the following key robustness property: the algorithm should continue to perform well even when the system model is changed slightly. Interestingly, our proposed iLQF with PullUp algorithm has this key robustness property.

In this paper, we derived rate function optimality results for symmetric ON-OFF arrival processes and stated results (strictly positive rate function, Corollary 2) when the arrival process are i.i.d., bursty. Further, we have limited extensions for this case where the arrival processes are asymmetric where the arrival rate of each user could be different (please see [3]). Suppose that the arrival probability to user i when n users are in the system is given by $p_i^{(n)}$, and further, $\limsup_{n\to\infty} \max_{1\leq i\leq n} p_i^{(n)} = \beta \in (0,1)$. Then, we can use the sample-path dominance property established in this paper to ensure that the overflow probability under iLQF is upper bounded by a symmetric system whose arrival rate is β. This enables us to establish rate-function optimality properties for such an asymmetric case. However, this technique does not permit generalizations to the case where there are "large differences" in the arrival rates to users (where some users could have an arrival rate exceeding '1' and other with less than '1' such that the overall system is still stabilizable). Future work will focus on this and other related issues.

Acknowledgments

This work was partially supported by NSF Grants CNS-0347400, CNS-0519535, CNS-0634898, CNS-0721380, and CNS-0831756, the Darpa ITMANET program and the DTRA grant HDTRA1-08-1-0016.

11. 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. Technical report, The University of Texas at Austin, WNCG, 2009.

[4] A. Dembo and O. Zeitouni. *Large Deviations Techniques and Applications*. Springer-Verlag New York, Inc., second edition, 1998.

[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] A. Ganti, E. Modiano, and J. Tsitsiklis. Optimal transmission scheduling in symmetric communication models with intermittent connectivity. *IEEE Trans. Inform. Theory*, 53:998–1008, March 2007.

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

[9] Jon Kleinberg and Éva Tardos. *Algorithm Design.* Pearson Education, 2006.

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

[11] M. J. Neely. Delay Analysis for Max Weight Opportunistic Scheduling in Wireless Systems. In *Forty-Sixth Annual Allerton Conference On Communication, Control, and Computing*, Sep. 2008.

[12] 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.

[13] S. Shakkottai. Effective capacity and QoS for wireless scheduling. *IEEE Trans. Automat. Contr.*, 53(3):749–761, February 2008.

[14] 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.

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

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

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

[18] 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.

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

[20] 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.

[21] 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.