Efficiency and Optimality of Largest Deficit First Prioritization: Resource Allocation for Real-Time Applications

Yuhuan Du and Gustavo de Veciana
Department of Electrical and Computer Engineering, The University of Texas at Austin
Email: dyhuan123@gmail.com, gustavo@ece.utexas.edu

***Abstract*—An increasing number of real-time applications with compute and/or communication deadlines are being supported on shared infrastructure. Such applications can often tolerate occasional deadline violations without substantially impacting their Quality of Service (QoS). A fundamental problem in such systems is deciding how to allocate shared resources so as to meet applications' QoS requirements. A simple framework to address this problem is to, (1) dynamically prioritize users as a possibly complex function of their deficits (difference of achieved vs required QoS), and (2) allocate resources so to expedite users with higher priority. This paper focuses on a general class of systems using such priority-based resource allocation. We first characterize the set of feasible QoS requirements and show the optimality of max weight-like prioritization. We then consider simple weighted Largest Deficit First (w-LDF) prioritization policies, where users with higher weighted QoS deficits are given higher priority. The paper gives an inner bound for the feasible set under w-LDF policies, and, under an additional monotonicity assumption, characterizes its geometry leading to a sufficient condition for optimality. Additional insights on the efficiency ratio of w-LDF policies, the optimality of hierarchical-LDF and characterization of clustering of failures are also discussed.**

I. Introduction

A growing number of real-time applications with compute and/or communication deadlines are being moved onto shared infrastructure, e.g., ranging from embedded systems to efficient cloud infrastructure. Such applications include control, multimedia processing, and/or machine learning components associated with enabling various types of user services as well as wireless, intelligent transportation and energy systems. In many cases such applications can tolerate occasional deadline violations, i.e., have soft constraints, without impacting the application Quality of Service (QoS). For example, applications with feedback can quickly compensate for errors, or humans may tolerate occasional failures in video processing since they can be partially concealed, or wireless base stations can tolerate occasional frame losses, since these can be retransmitted. More generally real-time applications' long-term QoS may depend in a complex manner on what was accomplished on time, e.g., partial completion of a set of tasks, or notions of video quality.

Enabling efficient sharing of compute/communication resources is a challenging problem. On the one hand, even for a single resource, tying the sharing model, e.g., round robin, priority schemes, to QoS metrics is generally hard due to the uncertainty in applications' workloads and possible variations in processing speeds. On the other hand, today's applications leverage complex networks of heterogeneous compute/communication resources, e.g., multi-core computers, embedded network system, or combinations of computation on mobile devices and the cloud. Consider the example in Fig.1. User 1 periodically generates a task that needs to be processed sequentially on Resources A, B, C, D in each period while User 2 generates tasks to be processed on Resource B then C. How should one go about designing resource sharing policies across multiple heterogeneous resources, where parallelism, task preemption and migration are allowed? Furthermore, how can one address heterogeneous QoS requirements associated with real-time applications? For example, User 1's QoS may still benefit from partial completions while User 2 only benefits if all processing is completed. This general class of problems involving both heterogeneous resources and user QoS requirements is the focus of this paper.

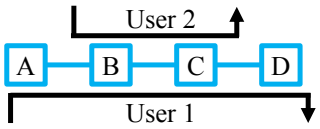

Fig. 1. An example for a network of resources. A, B, C and D represent compute/communication resources.

The design space of possible solutions to this problem is huge and has been explored in many research communities. In this paper we study an approach to resource allocation based on a decomposition of concerns:

- user priorities are dynamically set based on the history outcomes;
- and, resources are allocated so as to favor users with higher priority.

In such a framework there is quite a bit of latitude in choosing how priorities are set, and in turn how these affect the allocation of resources. For example, users' priorities could be set based on measured deficits, the "difference" of the required and achieved QoS, i.e., Largest Deficit First (LDF) prioritization. In turn, for complex systems such as that in Fig.1, resources could be allocated greedily giving preemptive access to tasks associated with higher-priority users.

This research was supported by Huawei Technologies Co. Ltd.

In general an optimal user prioritization strategy could leverage detailed information regarding how these priorities will impact the allocation of resources and completion outcomes to achieve the best possible user QoS. Such strategies require excessive amounts of information regarding the underlying compute/communication resources and resource allocation mechanism, and thus are generally hard to implement. By contrast, LDF-based prioritization is quite intuitive. It requires only tracking of users' possibly heterogeneous QoS deficits, in this sense it is truly decoupling user prioritization from the underlying priority-based resource allocation. Unfortunately, it is known to be suboptimal in certain settings [1], [2], [3].

A theoretical study of the efficiency and, possibly optimality, of LDF-based prioritization systems supporting real-time users with heterogeneous QoS requirements is the main focus of this paper. We note, however, that we do not directly address the design of the underlying priority-based resource allocation, although we consider some natural characteristics it could have to ensure optimality when combined with LDF user prioritization.

Related Work. There have been many efforts on the study of such dynamic prioritization policies in the context of specific resource, workload and/or QoS models.

The authors in [4], [5] propose a framework to model a wireless access point serving a set of clients that in each period generate packets which need to be transmitted by the end of the period. In their model only one client can transmit at a time and thus the access point can be viewed as a single resource. Each client transmits its packets over a unreliable channel which has a fixed probability of success, and thus, the time to successfully transmit a packet can be modeled as a geometric random variable. In this setting the authors show that the LDF policy is "optimal." However, the results are restricted to a single resource shared by users with geometric workloads. In this paper we study the performance of LDF in a more general setting which includes this prior work as a special case. This initial set of papers motivated follow-up work in wireless context, e.g., [6], [7], [8].

The performance of LDF and similar policies has also been studied in [1], [2], [3]. The authors in [1] consider the generalized switch model and were the first to propose the notion of "local pooling" as a sufficient condition for the Longest Queue First (LQF) policy to be throughput optimal. Subsequently, the work in [2] considers a multi-hop wireless network under a node-exclusive interference model and shows that the efficiency ratio of the greedy maximal matching policy, which is essentially LQF, equals to the "local pooling" factor of the network graph. More recently, the authors in [3] consider real-time traffic in ad hoc wireless networks under a link-interference model and also characterize the efficiency ratio of the LDF policy.

The results in [1], [2], [3] depend on the constant service rate model and the specific interference model, i.e., where the set of links/queues that can be scheduled simultaneously is restricted. These models may be appropriate in some wireless/queueing networks but do not necessarily hold in our broader context, e.g., soft real-time applications with stochastic workloads. Also, [1] lacks a performance analysis of LQF when it is not optimal and the works in [2], [3] focus on the efficiency ratio of LDF-like policies but lack a characterization of the full capacity region of these policies. Moreover, when the system can deliver more than the requirements, either the QoS requirements for real-time traffic or the throughput requirements for queueing systems, there is no discussion of how to manage the allocation of the "excess capacity" across users.

The authors in [9], [10], [11], [12] propose max weight scheduling policies for different queueing systems and show them to be throughput optimal via the approaches summarized in [13], [14]. As we will see in the sequel we too discuss a max weight-like scheduling policy, but it suffers from the usual complexity problems when the decision space is large and it requires excessive amounts of information, motivating us to consider simpler policies.

Additional related work includes modeling and scheduling real-time tasks, see e.g., [15], [16].

Our Contributions. In this paper, we contribute to the theoretical understanding and performance characterization of the Largest Deficit First (LDF) policy with applications to resource allocation to support real-time services. We make three key contributions.

First, we propose a novel general model for a class of systems supporting priority-based resource allocation and study different dynamic prioritization policies. This model is general in terms of the "impact" the priority decisions can have on the QoS payoffs. Specifically, in each period the payoffs under a priority decision are modeled by a random vector, which includes as special cases the single resource model, the geometric/constant workload and/or specific interference model adopted in prior work. For this general model, we propose a general inner bound R_{IB} for the QoS feasibility region of LDF prioritization policy.

Second, with an additional property, *monotonicity in payoffs*, we characterize the geometry of the inner bound R_{IB}. Based on this, we further propose a sufficient condition for the optimality of the LDF policy and characterize the efficiency ratio of LDF. In practice, understanding the geometry of R_{IB} enables us to understand and identify possible bottlenecks in the priority-based resource allocation infrastructure. We also show the LDF/hierarchical-LDF is optimal when there are two users or two classes of exchangeable users.

Finally, we also consider the class of weighted LDF policies, which enable us to explore the allocation of "excess payoffs" when the system has "excess" capacity. Simulation results are exhibited to show the impact of weights and to characterize the clustering of failures.

Paper Organization. The paper is organized as follows: Section II introduces our general model for systems supporting priority-based resource allocation. Section III develops theoretical results and characterizes the performance of the weighted LDF policies while Section IV presents some examples for the optimality of the weighted LDF/hierarchical-LDF policies.

Section V discusses some practical issues while the impact of weights is exhibited via simulation in Section VI. Section VII concludes the paper and points to future work.

II. System Model

We consider applications which periodically generate random workloads with the same period and specify long-term QoS requirements. In the sequel we let a user denote a specific instance of such an application.

We begin by introducing a general model for systems that allocate resources in each period based on the following decomposition: (1) users are assigned priorities dynamically, e.g., at runtime, according to a function of the past history, and (2) the system allocates resources based on these priorities.

For the most part in this paper, the manner in which (2) is carried out will not be our concern. Instead our focus will be on how to perform dynamic user prioritization to achieve optimal (or near-optimal) system performance when combined with a given underlying mechanism for (2).

A. General Model for Systems Supporting Priority-Based Resource Allocation

We consider an abstract system that serves n users indexed from 1 to n. Let $N = \{1, 2, \cdots, n\}$ be the user set. The system operates in discrete time, over periods $t = 1, 2, \cdots$. In each period, it picks a user *priority decision* $\mathbf{d} = (d_1, d_2, \ldots, d_n)$ where d_m is the index of the user with m^{th} highest priority. We let D denote the set of all possible priority decisions and let $|D|$ represent the number of possible decisions, thus, $|D| = n!$

In each period, given the priority decision $\mathbf{d}$ passed to the underlying resources, since there are intrinsic uncertainties in users' workloads, each user i achieves a non-negative random QoS payoff, denoted by $V_i(\mathbf{d})$. We let $\mathbf{V}(\mathbf{d}) = (V_1(\mathbf{d}), V_2(\mathbf{d}), \cdots, V_n(\mathbf{d}))$. We assume the payoffs are independent across periods. The distribution of $\mathbf{V}(\mathbf{d})$ depends on the selected priority decision $\mathbf{d}$ and the expected payoff vector given $\mathbf{d} \in D$ is denoted by $\mathbf{p}(\mathbf{d}) = \mathrm{E}[\mathbf{V}(\mathbf{d})]$. We assume all possible payoff vectors form a finite rational set.

Each user requires a long-term average QoS payoff $q_i \geq 0$ as the QoS requirement. We let $\mathbf{q} = (q_1, q_2, \cdots, q_n)$ and assume q_i's are rational[1]. We denote by $\mathbf{d}(t)$ the priority decision at period t. To keep track of the deficits between required and achieved QoS payoffs, for each user $i \in N$ and period $t+1$, we define[2]

$$X_i(t+1) = [X_i(t) + q_i - V_i(\mathbf{d}(t+1))]^+, \qquad (1)$$

where $[x]^+ = \max[x, 0]$.

The goal is thus to devise user prioritization policies which will meet users' long-term payoff requirements.

Definition 1: A **user prioritization policy** is a stationary policy that picks a priority decision $\mathbf{d}(t+1) \in D$ at period $t+1$ based on the following:

- users' payoff requirement vector $\mathbf{q}$;
- expected payoff vectors $P = \{\mathbf{p}(\mathbf{d}) | \mathbf{d} \in D\}$;
- and, the deficits $\mathbf{X}(t) = (X_1(t), X_2(t), \cdots, X_n(t))$.

The process $\{\mathbf{X}(t)\}_{t \geq 1}$ is a Markov chain under any such policy. We assume the initial state $\mathbf{X}(0)$, the requirements $\mathbf{q}$, the set of all possible payoff vectors and the user prioritization policy make $\{\mathbf{X}(t)\}_{t \geq 1}$ an irreducible Markov chain.

Definition 2: A payoff requirement vector $\mathbf{q}$ is said to be **feasible** if there exists a user prioritization policy η under which the Markov chain $\{\mathbf{X}(t)\}_{t \geq 1}$ is positive recurrent. We also say this policy fulfills this requirement vector.

The expected payoff vectors $P = \{\mathbf{p}(\mathbf{d}) | \mathbf{d} \in D\}$ could in principle be statistically inferred from the history events or by repeated experiments. However, in a practical setting this can be challenging and it is of interest to find a policy that performs well and uses little a-priori information regarding the exponential set of expected payoff vectors P.

Note that this model is general in the sense that the "impact" of priority decisions $\mathbf{d} \in D$ on the QoS payoff vectors P is at this point general, whereas the specific resource and workload models in prior work, e.g., [1], [2], [3], implicitly impose properties on P and therefore restrict the results significantly.

B. Example: Centralized Computing System for Real-Time Applications

Our model can for example capture a centralized computing infrastructure supporting Soft Real-Time (SRT) applications where the n users share compute resources. In a cloud-based collaborative video conferencing context, a user might correspond to an individual end user and the period length might correspond to the length of a group of video frames.

The users generate streams of tasks periodically. Specifically in each period a user generates several tasks. A task may further consist of a graph of possibly dependent sub-tasks with (possibly) random processing requirements, i.e., workloads. These tasks/sub-tasks need to be fully completed before the end of the period. For real-time services, it is generally useless to process a task after its deadline. For example, in the video conferencing context it is not desirable to present an out-of-date frame. Therefore, we assume tasks/sub-tasks not completed on time are dropped.

In each period t, the user prioritization policy picks a user priority decision $\mathbf{d}(t)$, based on which compute resources are allocated to process tasks. Given the task processing results, a payoff $V_i(\mathbf{d}(t))$ is achieved for each user i based on whether the tasks were successfully processed, or how much of the task graphs were completed. In general, $V_i(\mathbf{d}(t))$ may represent any user-specific QoS payoff per period, that can be averaged over time, e.g., the quality/resolution of video frame processing, or the number of task completions. Accordingly the vector $\mathbf{q}$ represents the long-term average QoS requirements.

C. General Model for Complex Resources and Applications

As indicated in the introduction, our model also applies to a complex network of heterogeneous compute and communication resources, as long as users periodically and synchronously

[1]All the results in this paper can be generalized to models with irrational values. For simplicity in the proof we do not consider that level of generality.

[2]We truncate the deficit at 0 for the convenience of defining feasibility in the sequel. Removing the truncation won't change the results in the paper.

generate tasks that require timely processing on diverse resources and moving around in the network, e.g., as shown in Fig.1.

Given the priority decision in each period, the network of resources coordinate according to some priority-based resource allocation mechanism to accelerate the processing of tasks with high priorities, by reducing the communication/queueing delays, processing with higher processor speed, allocating more shared resources, etc.

Again, different users can define their payoffs in different ways and specify their QoS requirements accordingly.

III. Performance Analysis

In this section we shall develop theoretical results for such systems. To save space we have deferred proofs of these results to the extended version of this paper available at [17]. Some of these results are similar to prior work but in the more general model while other results are completely new. For completeness we shall develop a self-contained theoretical framework.

A. System Feasibility Region and Feasibility Optimal Policy

The set of all feasible long-term payoff requirement vectors will be referred to as the *system feasibility region* F. We let F_η denote the feasibility region of a user prioritization policy η. To characterize F we introduce some further notation.

A vector $\mathbf{x}$ is said to be *dominated* by a vector $\mathbf{y}$ if $x_i \leq y_i$ for all i and is denoted by $\mathbf{x} \preceq \mathbf{y}$. We define $\mathbf{x} \prec \mathbf{y}$, $\mathbf{x} \succeq \mathbf{y}$ and $\mathbf{x} \succ \mathbf{y}$ in a similar manner.

Given the set of priority decisions D and the expected payoff vectors $P = \{\mathbf{p}(\mathbf{d}) | \mathbf{d} \in D\}$, we let C be the set of requirement vectors $\mathbf{q} \in \mathbb{R}_+^n$ which are dominated by a vector in the convex hull of P denoted Conv(P), i.e.,

$$C \equiv \{\mathbf{q} \in \mathbb{R}_+^n \mid \exists \mathbf{x} \in \text{Conv}(P) \text{ such that } \mathbf{q} \preceq \mathbf{x}\}. \quad (2)$$

Fig.2 exhibits C for a two-user (left figure) and three-user (right figure) setting. In the two-user setting, the points labeled $\mathbf{p}(\mathbf{d}_1)$ and $\mathbf{p}(\mathbf{d}_2)$ are the expected payoff vectors of two priority decisions, i.e., where User 1 or User 2 has higher priority, respectively. The shadowed area represents C. In the three-user setting, the circles represent the 6 possible expected payoff vectors, and the region dominated by their convex hull is C. Note that in a n-user scenario where $n \geq 3$, as displayed the expected payoff vectors need not be on a hyperplane in the n-dimensional space. As we will see this is essentially the source of complexity in studying such systems.

Clearly, for any requirement vector $\mathbf{q}$ in the interior of C, denoted by int(C), one can achieve $\mathbf{q}$ if one is allowed to do probabilistic time sharing among priority decisions by picking decisions according to a pre-computed probability distribution whose mean payoff dominates $\mathbf{q}$. Therefore, int(C) $\subseteq F$. We can also show the following result.

***Lemma** 1:* The system feasibility region F is such that

$$F \subseteq \text{cl}(C),$$

where cl(C) is the closure of C.

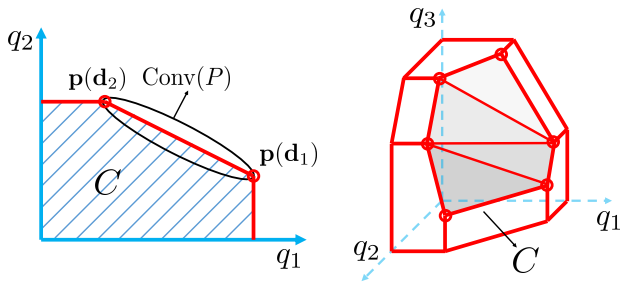

Fig. 2. Examples of set C when $n = 2$ and $n = 3$.

Intuitively, if $\mathbf{q}$ is feasible, it is fulfilled by some user prioritization policy that in the long-term picks each priority decision some fraction of the time and thus, $\mathbf{q}$ is dominated by some point in the convex hull of P. This is similar to prior work, e.g., [9]. See the extended version of this paper [17] for the proof. In other words, C is different from F by at most a boundary, and therefore, characterizes F for practical purposes. Thus, in the sequel we will also refer to C as the system feasibility region.

Ideally, it is desirable to devise an "optimal" policy that can fulfill all feasible requirements. More formally, a user prioritization policy η is said to be *feasibility optimal* if int(C) $\subseteq F_\eta \subseteq$ cl(C). Similar to prior work [9], [10], the following max weight-like policy is one such feasibility optimal policy.

***Definition** 3:* The **deficit-based max weight (MW)** prioritization policy is such that, at period $t+1$, given the deficit vector $\mathbf{X}(t)$ computed by (1), it picks a priority decision $\mathbf{d}(t+1)$ that satisfies

$$\mathbf{d}(t+1) \in \arg\max_{\mathbf{d} \in D} \langle \mathbf{X}(t), \mathbf{p}(\mathbf{d}) \rangle, \quad (3)$$

where $\langle \mathbf{x}, \mathbf{y} \rangle$ is the inner product of two vectors.

***Theorem** 1:* The feasibility region of the MW policy F_{MW} is such that

$$\text{int}(C) \subseteq F_{MW} \subseteq \text{cl}(C),$$

and therefore, the MW policy is feasibility optimal.

See the extended version of this paper [17] for the proof.

However, the MW policy and time sharing policies require full knowledge of P which is challenging in complex practical systems. Moreover, these policies are hard to implement since they involve solving fairly complex optimization problems, i.e., Eq (3). Changes in the user set or payoff requirement vector $\mathbf{q}$ will also impact the realization of these policies. In summary, the requirements in terms of a-priori knowledge, the computational complexity and lack of flexibility to changes make them hard to use in practice. This motivates the policies considered in the next subsection.

For ease of reference, Table I provides a summary of the notation used to denote various regions used in the rest of the paper—some of these are introduced in the sequel.

TABLE I
NOTATION OF REGIONS.

Regions	Description
F	System Feasibility Region.
Conv(P)	Convex hull of the expected payoff vectors.
C	Region dominated by Conv(P).
$F_{\mathbf{w}\text{-LDF}}$	Feasibility region of the $\mathbf{w}$-LDF policy.
R_{IB}	An inner bound for $F_{\mathbf{w}\text{-LDF}}$
B	Dominant of the convex hull.
R	Region characterizing the geometry of R_{IB}.

B. Weighted LDF Policies and Associated Feasibility Regions

The LDF user prioritization policies require no a-priori knowledge of the system, are simple to implement and adapt easily to changes in $\mathbf{q}$ or the user set. In particular we shall characterize the feasibility regions of these policies by providing an inner bound.

***Definition** 4:* Given a vector $\mathbf{w} = (w_1, w_2, \cdots, w_n) \succ \mathbf{0}$, the **weighted Largest Deficit First ($\mathbf{w}$-LDF)** user prioritization policy is such that, at period $t+1$, given the deficit vector $\mathbf{X}(t)$, it picks a priority decision $\mathbf{d}$ that satisfies

$$w_{d_1} X_{d_1}(t) \geq w_{d_2} X_{d_2}(t) \geq \cdots \geq w_{d_n} X_{d_n}(t),$$

with ties broken arbitrarily (possibly randomly). In other words, it sorts the weighted deficits of users and assigns priorities accordingly. Let $\mathbf{1} \equiv (1, 1, \cdots, 1)$. We refer to the policy with $\mathbf{w} = \mathbf{1}$ the **Largest Deficit First (LDF)** policy.

Clearly, the $\mathbf{w}$-LDF prioritization policies do not require knowledge of the expected payoff vectors P. Note that we still use deficit feedback to stabilize the system. In terms of computational complexity, solving (3) is $O(n!)$ while sorting weighted deficits only requires $O(n \log n)$. It also allows us to further differentiate the performance across users by assigning different weights. The impact of weights is discussed in Section VI.

Prior work has established that the LDF policy need not be feasibility optimal. Therefore, a key question is whether the feasibility regions for the $\mathbf{w}$-LDF policies are acceptable and to characterize the gap between their feasibility regions and the system feasibility region F. To that end, we first provide an inner bound, denoted by R_{IB}, for the feasibility region of any $\mathbf{w}$-LDF policy.

***Theorem** 2:* For any $\mathbf{w} \succ \mathbf{0}$, an inner bound for the feasibility region of the $\mathbf{w}$-LDF policy $F_{\mathbf{w}\text{-LDF}}$ is given by $\text{int}(R_{\text{IB}}) \subseteq F_{\mathbf{w}\text{-LDF}}$, where

$$R_{\text{IB}} \equiv \{\mathbf{q} \in \mathbb{R}_+^n \mid \exists \boldsymbol{\alpha} \succ \mathbf{0} \text{ such that } \forall S \subseteq N, \sum_{i \in S} \alpha_i q_i \leq \min_{\mathbf{d} \in D(S)} \sum_{i \in S} \alpha_i p_i(\mathbf{d})\} \quad (4)$$

where $D(S)$ denotes the set of all priority decisions that assign the highest $|S|$ priorities to users in S.

In other words, if $\mathbf{q} \in R_{\text{IB}}$, it is feasible under all $\mathbf{w}$-LDF policies except perhaps boundary points. The underlying intuition for this bound is as follows. A vector $\mathbf{q}$ is in R_{IB} if there is a weight vector $\boldsymbol{\alpha} \succ \mathbf{0}$ such that for any subset of users S, and decisions giving users in S the highest priorities, the weighted sum of payoff requirement $\sum_{i \in S} \alpha_i q_i$ will not exceed the least sum weighted payoff $\sum_{i \in S} \alpha_i p_i(\mathbf{d})$. Based on $\boldsymbol{\alpha}$, we can construct an appropriate Lyapunov function to show feasibility for $\mathbf{q}$ and each $\mathbf{w}$. See the extended version of this paper [17] for the proof.

Understanding the geometry of R_{IB} enables us to characterize the performance gap between $\mathbf{w}$-LDF and feasibility optimal policies. Let us informally consider the geometry of R_{IB} for the two special cases in Fig.2. In the two-user case in Fig.2, R_{IB} is the same as C and thus, the $\mathbf{w}$-LDF policies are always feasibility optimal. However, in the three-user case in Fig.2, this need not be true. Indeed, in this setting, the region R_{IB} corresponds to C minus the convex hull of P, modulo some boundary points. This is exhibited in Fig.3. In the next subsection, we will formalize these observations and show under what conditions they hold true.

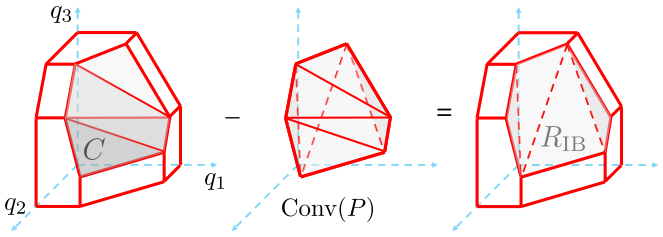

Fig. 3. Visualizing R_{IB} for the three-user scenario in Fig.2. In this example, $R_{\text{IB}} = \text{cl}(C - \text{Conv}(P))$.

C. Geometry of R_{IB} under Monotonicity in Payoffs

In order to formally characterize the geometry of R_{IB} we will add a further natural requirement to the general model.

We define $S_i(\mathbf{d})$ to be the set of users that have higher priorities than user i under decision $\mathbf{d}$.

***Definition** 5:* The system with expected payoff vectors $P = \{\mathbf{p}(\mathbf{d}) | \mathbf{d} \in D\}$ is said to satisfy **monotonicity in individual expected payoff** if, for any two priority decisions $\mathbf{d}_1$ and $\mathbf{d}_2$ and any user i such that $S_i(\mathbf{d}_1) \subseteq S_i(\mathbf{d}_2)$, it is true that $p_i(\mathbf{d}_1) \geq p_i(\mathbf{d}_2)$. We call this **monotonicity in payoffs** for short.

In other words, a user i can expect to get a higher payoff if some users with higher priority are re-assigned lower priorities. This property characterizes in a broad sense how priorities impact the expected payoffs when the underlying system allocates resources. It is a natural condition but need not hold in general.

We shall define B to be the set of payoff requirement vectors $\mathbf{q}$ which dominate a vector in the convex hull of P, i.e.,

$$B \equiv \{\mathbf{q} \in \mathbb{R}_+^n \mid \exists \mathbf{x} \in \text{Conv}(P) \text{ such that } \mathbf{q} \succeq \mathbf{x}\}.$$

We call B the *dominant of the convex hull*. Contrast this to the definition of C in (2).

For the special cases in Fig.2 and 3, $B \cap C$ equals to Conv(P), but in general it can be larger than Conv(P). Fig.4

shows a conceptual picture of what could happen. The three circles represent three possible expected payoff vectors. Here, the whole shadowed area $B \cap C$ is larger than the region Conv(P) which is the triangle formed by the three circles. Note that this is only a conceptual example to help visualize $B \cap C$ in higher dimensions. In reality for two dimensions, i.e., systems with two users, we know there are only 2 expected payoff vectors as shown in Fig.2.

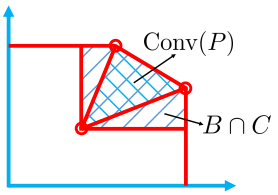

Fig. 4. An example where $B \cap C$ is larger than Conv(P).

In the sequel we will see that given monotonicity in payoffs, R_{IB} is obtained by "removing" $B \cap C$, rather than just Conv(P) from C. To develop this result we need some further notation associated with each subset of users $S \subseteq \{1, 2, \cdots, n\}$.

The projection of a vector $\mathbf{x}$ on the subspace of S is denoted by $\mathbf{x}^S$, i.e.,

$$x_i^S = \begin{cases} x_i & \text{if } i \in S \\ 0 & \text{otherwise.} \end{cases}$$

We let $P^S \equiv \{\mathbf{p}^S(\mathbf{d}) | \mathbf{d} \in D(S)\}$ represent the projections of expected payoff vectors corresponding to decisions in $D(S)$, i.e., which assign the highest priorities to users in S.

Given a subset S and P^S, we define the feasibility region C^S and the dominant of the convex hull B^S as follows.

$$C^S \equiv \{\mathbf{q}^S \in \mathbb{R}_+^n \mid \exists \mathbf{x}^S \in \text{Conv}(P^S) \text{ such that } \mathbf{q}^S \preceq \mathbf{x}^S\},$$
$$B^S \equiv \{\mathbf{q}^S \in \mathbb{R}_+^n \mid \exists \mathbf{x}^S \in \text{Conv}(P^S) \text{ such that } \mathbf{q}^S \succeq \mathbf{x}^S\}.$$

Note that C^S and B^S are not necessarily the same as projecting C and B on the subspace of S, respectively. This is because in the definitions of C^S and B^S, we only focus on a subset of decisions $D(S)$ rather that the full decision set D.

Let us now define a region R which will help characterize the geometry of the inner bound R_{IB}.

***Definition** 6:* Let R be defined as follows:

$$R \equiv \{\mathbf{q} \in \mathbb{R}_+^n \mid \forall S \subseteq N, \mathbf{q}^S \in C^S \setminus B^S\},$$

where $C^S \setminus B^S = \{\mathbf{q}^S | \mathbf{q}^S \in C^S, \mathbf{q} \notin B^S\}$. In other words, any $\mathbf{q} \in R$ satisfies that for any user subset S, its projection on the subspace of S belongs to the set $C^S \setminus B^S$, which is the feasibility region C^S minus the dominant of the convex hull B^S.

One can visualize obtaining the set R as a process of removing $B^S \cap C^S$ from C^S in all subspaces corresponding to all subsets S. The geometry of R_{IB} is then captured as follows.

***Theorem** 3:* If the system satisfies monotonicity in payoffs, then the inner bound region R_{IB} is such that

$$\text{int}(R) \subseteq R_{\text{IB}} \subseteq \text{cl}(R).$$

See the extended version of this paper [17] for this somewhat technical proof.

D. Sufficient Condition for w*-LDF's Optimality*

By Theorem 2 and Theorem 3, we immediately get

$$\text{int}(R) \subseteq \text{int}(R_{\text{IB}}) \subseteq F_{\mathbf{w}\text{-LDF}}. \tag{5}$$

Since R is obtained by removing $B^S \cap C^S$ from C^S for each S, if what is removed is nothing more than a boundary, the difference between R and C is at most a boundary and thus w-LDF policies are feasibility optimal. It is easy to see this happens when vectors in P^S lie on a hyperplane for each subset of users S. This can be formalized as follows.

***Definition** 7:* The system with expected payoff vectors $P = \{\mathbf{p}(\mathbf{d}) | \mathbf{d} \in D\}$ is said to satisfy **subset payoff equivalence** if for each subset of users S the vectors in $P^S = \{\mathbf{p}^S(\mathbf{d}) | \mathbf{d} \in D(S)\}$ lie on a hyperplane, i.e., there exists a nonzero $\boldsymbol{\alpha}^S \succeq \mathbf{0}$ such that for all $\mathbf{d}_1, \mathbf{d}_2 \in D(S)$,

$$\langle \boldsymbol{\alpha}^S, \mathbf{p}^S(\mathbf{d}_1) \rangle = \langle \boldsymbol{\alpha}^S, \mathbf{p}^S(\mathbf{d}_2) \rangle.$$

***Theorem** 4:* If the system satisfies monotonicity in payoffs and subset payoff equivalence, then

$$\text{int}(C) \subseteq F_{\mathbf{w}\text{-LDF}} \subseteq \text{cl}(C),$$

and therefore, the w-LDF policies are feasibility optimal.

The conditions in this theorem are akin but not equivalent to the conditions introduced in [1] for the generalized switch model. See the extended version of this paper [17] for the proof of the theorem and a discussion of the difference between our proposed conditions and those implicitly imposed by the model in [1].

If the system has only two users, then clearly subset payoff equivalence is satisfied since the two expected payoff vectors are always on a line. Therefore, we get the following corollary.

***Corollary** 1:* If the system has two users and satisfies monotonicity in payoffs, then w-LDF policies are feasibility optimal.

Note that in a two-user scenario, the property of monotonicity in payoffs simply means a user gets higher payoff under the higher priority than its payoff under the lower priority. In Section IV-B we will consider systems serving two classes of exchangeable users and use this corollary to show the optimality of LDF-like policies.

E. Efficiency Ratio Analysis

When the conditions in Theorem 4 do not hold, one can still study the efficiency ratio, see e.g., [2], to evaluate the performance of w-LDF policies.

***Definition** 8:* The **efficiency ratio** of the w-LDF policy is defined as

$$\gamma_{\mathbf{w}\text{-LDF}} = \sup\{\gamma | \gamma F \subseteq F_{\mathbf{w}\text{-LDF}}\}.$$

Clearly $\gamma_{\mathbf{w}\text{-LDF}}$ equals to 1 if and only if the w-LDF policy is feasibility optimal.

If a system does not satisfy subset payoff equivalence, i.e., for some subset of users S the vectors in P^S are not on the same hyperplane, we can characterize the "heterogeneity" of these vectors based on the following notion.

***Definition** 9:* Given a subset of users $S \subseteq N$, the **subset payoff ratio** σ_S for S is defined as

$$\sigma_S = \max_{\substack{\boldsymbol{\alpha}^S \succeq \mathbf{0} \\ \boldsymbol{\alpha}^S \neq \mathbf{0}}} \frac{\min_{\mathbf{d} \in D(S)} \langle \boldsymbol{\alpha}^S, \mathbf{p}^S(\mathbf{d}) \rangle}{\max_{\mathbf{d} \in D(S)} \langle \boldsymbol{\alpha}^S, \mathbf{p}^S(\mathbf{d}) \rangle}. \tag{6}$$

The optimal $\boldsymbol{\alpha}^S$ is such that the projections of the vectors in P^S on $\boldsymbol{\alpha}^S$ are as close to each other as possible.

Clearly if the vectors in P^S are on the same hyperplane, then $\sigma_S = 1$ and the optimal $\boldsymbol{\alpha}^S$ is the normal vector to the hyperplane. Intuitively, σ_S characterizes the degree to which the vectors in P^S deviate from being on the same hyperplane.

This notion enables us to characterize the efficiency ratio of $\mathbf{w}$-LDF for a given system.

***Theorem** 5:* If the system satisfies monotonicity in payoffs, the efficiency ratio of the $\mathbf{w}$-LDF policy is such that

$$\gamma_{\mathbf{w}\text{-LDF}} \geq \min_{S \subseteq N} \sigma_S.$$

See the extended version of this paper [17] for the proof. Intuitively, the bottleneck of the efficiency ratio is the subset S where σ_S is the smallest.

Note that by picking any $\boldsymbol{\alpha} \succ \mathbf{0}$, we can get lower bounds on σ_S for all subsets $S \subseteq N$ by placing its projection $\boldsymbol{\alpha}^S$ into (6). Thus, any $\boldsymbol{\alpha} \succ \mathbf{0}$ enables us to construct a lower bound on $\gamma_{\mathbf{w}\text{-LDF}}$. A trivial option is $\boldsymbol{\alpha} = \mathbf{1}$, where for each subset S the value of $\langle \mathbf{1}^S, \mathbf{p}^S(\mathbf{d}) \rangle$ represents the sum payoff of users in S under decision $\mathbf{d}$.

We have shown that the efficiency and optimality of the $\mathbf{w}$-LDF policies is related to R_{IB}. Understanding and analyzing the geometry of R_{IB} can in principle enable us to provide feedback to the designers of priority-based resource allocation mechanisms regarding which specific priority decision or set of priority decisions are problematic and bottlenecks for the system so that the designers can focus on improving the resource allocation for these problematic decisions. For example, in the conceptual setting shown in Fig.4, the priority decision corresponding to the lower left circle is the "bottleneck" of the system and should be targeted to make the dominant of the convex hull as small as possible. This is of particular interest for some practical systems where it is possible to get explicit knowledge of P which reflect the underlying priority-based resource allocation, e.g., by collecting data over a long time.

A priority decision is problematic if the associated underlying resource allocation suffers from resource contention, blocking among users/applications, or even deadlocks on compute resources, etc. Based on feedback regarding the bottlenecks, the designer could improve the associated resource allocation schemes, e.g., by increasing the processing speed of the certain computing resources, spending more energy, reducing the contention, and/or resolving the blocking/deadlock, and thus, improve the efficiency of the overall system under the $\mathbf{w}$-LDF prioritization policies.

IV. Examples for $\mathbf{w}$-LDF's Optimality

Theorem 4 gives a sufficient condition for $\mathbf{w}$-LDF to be feasibility optimal. One example system that satisfies these conditions is the model considered in prior work [4] which, as mentioned in Section I, can be viewed as a single-resource geometric-workload model. In this section we consider more system settings and show how our results provide useful insights in practice.

A. Exchangeable Expected Payoffs

We start by considering systems with underlying symmetry.

***Definition** 10:* A subset of users S is said to have **exchangeable expected payoffs** if, for all priority decisions $\mathbf{d} \in D$ and all $i, j \in S$, if we switch the priorities of user i and j and use $\mathbf{d}'$ to represent the resulting new priority decision, then

$$p_k(\mathbf{d}') = \begin{cases} p_k(\mathbf{d}) & \text{if } k \neq i, j \\ p_j(\mathbf{d}) & \text{if } k = i \\ p_i(\mathbf{d}) & \text{if } k = j. \end{cases}$$

In other words, exchanging the priorities of two users in S will simply exchange their expected payoffs without impacting that of other users. This would be true if the priority-based resource allocation were symmetric for users in S and the users generate tasks with identically distributed or exchangeable workloads.

If the user set N have exchangeable expected payoffs, we can verify the property of subset payoff equivalence by picking $\boldsymbol{\alpha}^S = \mathbf{1}^S$ for each subset of users S. Therefore, by Theorem 4 we get the following corollary.

***Corollary** 2:* If the set of users N have exchangeable expected payoffs and the system satisfies monotonicity in payoffs, then the $\mathbf{w}$-LDF policies are feasibility optimal.

See the extended version of this paper [17] for the proof.

B. Multiple Classes of Exchangeable Users and Hierarchical-LDF

In this subsection, we first consider a system supporting two classes of exchangeable users. Formally, a class of users is *exchangeable* if they have exchangeable expected payoffs and the same QoS requirement. The users in different classes may have distinct payoffs and QoS requirements. In some contexts it is of practical interest to first prioritize the classes and then prioritize users in each class, respectively. We refer to such schemes as using *class-based hierarchical prioritization*.

In practice, depending on whether the priorities of classes can change dynamically, there are two types of class-based hierarchical prioritization: Type 1 where the class priorities are fixed, and Type 2 where one is allowed to dynamically prioritize classes of users.

The first type of hierarchical prioritization might correspond to a setting where the users/applications are separated into human-interactive/high-QoS and background-processing/low-QoS categories [18], and it is always desirable to first process high-QoS users. In this setting, the problem is reduced to a collection of independent user prioritization problems similar to the one considered in this paper. By Corollary 2, $\mathbf{w}$-LDF is feasibility optimal to prioritize users in each class.

The second type of dynamic hierarchical prioritization might be of interest in systems where switching between processing different user classes involves overheads, and/or

where it is inefficient to mix the processing of different user classes, probably because of resource contention or deadlocks.

In this setting, we propose a *hierarchical-LDF* policy that in each period works in two steps by (1) prioritizing classes by LDF based on the aggregate deficits, i.e., the sum of deficits for users in the same class, and (2) prioritizing users in each class according to LDF based on individual users' deficits. Note that here LDF can be replaced by $\mathbf{w}$-LDF for any $\mathbf{w} \succ \mathbf{0}$ and the following result would hold.

***Theorem** 6:* In a system with two classes of exchangeable users, if the property of monotonicity in payoffs is satisfied, the hierarchical-LDF policy is feasibility optimal among all possible class-based hierarchical prioritization policies.

Intuitively, by Corollary 1 we know the class-based LDF policy is optimal to set priorities amongst the two classes and by Corollary 2 we know the LDF-based user prioritization is also optimal for the exchangeable users in each class. We omit the proof here to save space.

More generally, for systems serving multiple (more than two) classes of exchangeable users, one can view each class as a "super user", and define the aggregate payoff and QoS requirement for a super user to be the sum of payoffs and QoS requirements for users in that class, respectively. Then the dynamic prioritization of super users can be viewed as the problem considered in this paper. Therefore, by Theorem 4, if the system with the super users' expected aggregate payoffs satisfies monotonicity in payoffs and subset payoff equivalence, the LDF policy is feasibility optimal to prioritize super users and thus, the hierarchical-LDF policy is feasibility optimal among all class-based hierarchical prioritization policies. Indeed, all the results we have introduced, e.g., Theorem 1-5, still hold for the prioritization of these super users.

V. Some Practical Issues

In practice, besides meeting minimum payoff requirements, users may be willing to pay for additional payoffs, e.g., better video quality in the video conferencing setting, albeit at possibly different prices. Given the requirements $\mathbf{q}$ and the achieved average payoffs $\mathbf{p} = (p_1, p_2, \cdots, p_n)$, we call $p_i - q_i$ the *excess payoff* for each user i. While using $\mathbf{w}$-LDF policies to fulfill users' payoff requirements, we also want to manage the allocation of excess payoffs across users, perhaps with the aim of maximizing the benefits to the system or users.

However, the non-negative definition of deficit (1) makes it hard to track excess payoffs. For example, consider a model with 2 users and suppose the payoff is always 1 for the high priority user and 0 for the low priority user. Suppose the payoff requirement vector is $\mathbf{q} = (0.1, 0.5)$. Since $1 > 0.1 + 0.5$, we know $\mathbf{q}$ is feasible and the system can deliver 0.4 excess payoff. Suppose we use the LDF policy, starting from $\mathbf{X}(0) = (0, 0)$ it is easy to verify[3] that the system will switch giving high priority to these two users, and thus the achieved average payoff vector is $\mathbf{p} = (0.5, 0.5)$. Clearly User 1 gets 0.4 excess payoff while User 2 gets nothing. This happens because $X_1(t)$ and $X_2(t)$ are frequently forced to 0 from different negative values, which causes the "unfairness" between these two users.

To solve this problem, we modify the deficit definition for each user i and period $t+1$ as follows,

$$X_i'(t+1) = X_i'(t) + q_i - V_i(\mathbf{d}(t+1)), \tag{7}$$

i.e., we allow $X_i'(t)$ to be negative.

Now for the simple example above, if we adopt LDF but based on the possibly negative deficits $\mathbf{X}'(t) = (X_1'(t), X_2'(t), \cdots, X_n'(t))$, we can get achieved average payoff vector $\mathbf{p} = (0.3, 0.7)$. We observe that the two users equally split the excess payoff.

Intuitively, for each user i the modified deficit $X_i'(t)$ changes roughly linearly as t increases with the slope being $q_i - p_i$. Since $\mathbf{w}$-LDF policy aims to balance weighted deficit $w_i X_i'(t)$, we know $w_i(p_i - q_i)$ is roughly the same for all users. We will verify this observation in the simulation section and based on this we can manage the excess payoffs across users by picking the appropriate weight vector $\mathbf{w}$.

Note that for completeness we will need to modify the feasibility definition since the process $\{\mathbf{X}'(t)\}_{t\geq 1}$ is no longer positive recurrent as it may keep decreasing or increasing. Refer to the extended version of this paper [17] for details.

VI. Simulations

In this section we explore via simulation the impact of weights of $\mathbf{w}$-LDF policies.

Consider an illustrative system with single computing resource serving 3 soft real-time users. In each period of length $\delta = 10$, each user generates one task that need to complete by end of the period. We let the non-negative workload, i.e., task service time, distributions for three users be Gamma$(12, 0.5)$, Gamma$(4, 1)$ and Gamma$(10, 0.1)$, respectively. We pick these workload distributions to make them general and heterogeneous. In each period, the payoff for user i is 1 if user i's task completes and is 0 otherwise. Accordingly, user i's QoS requirement q_i represents the long-term task completion ratio.

We start with initial deficit $\mathbf{X}'(0) = (0, 0, 0)$. In each period, we independently generate task workloads for users and simulate the $\mathbf{w}$-LDF policy based on $\mathbf{X}'(t)$ to pick a priority decision. The single resource sequentially processes users' tasks from highest to lowest priority. Tasks not completed on time are dropped. All simulations are run for 30000 periods. A requirement vector $\mathbf{q}$ is feasible if it is dominated by the achieved task completion ratio vector $\mathbf{p}$ over the 30000 periods. The vectors $\mathbf{q}$ and $\mathbf{w}$ are specified in various settings in the sequel.

Note that in this setting monotonicity in payoffs is satisfied while subset payoff equivalence is not.

A. Impact of Weights on Long-Term Completion Ratios

In Table II we consider a requirement vector $\mathbf{q}$ that is feasible under the $\mathbf{w}$-LDF policies and display the achieved $\mathbf{p}$ under two different weight vectors $\mathbf{w}$. For each weight vector $\mathbf{w}$, we verify that $w_i(p_i - q_i)$ is the same for all three users.

[3]Since the payoffs are deterministic, we can verify this by evaluating the deficits for the first few periods and we will observe that the process $\{\mathbf{X}(t)\}_{t\geq 1}$ evolves in a periodic pattern.

Contrasting the two lines in Table II, we can see that for a system which can deliver more than required, changing the weight vector reallocates the excess payoffs and gives more excess payoffs to users with smaller weights.

TABLE II
ACHIEVED COMPLETION RATIO VECTORS UNDER TWO WEIGHT VECTORS.

$\mathbf{q}$	$\mathbf{w}$	Achieved $\mathbf{p}$	$w_i(p_i - q_i)$
$0.8, 0.6, 0.4$	$(1, 1, 1)$	$0.85, 0.65, 0.45$	0.05
	$(10, 1, 1)$	$0.809, 0.69, 0.49$	0.09

B. Characterization of Clustering of Failures and Impact of Weights

If a user's task is not completed in a period, we call it a failure event. The requirement vector $\mathbf{q}$ focuses on long-term task completion ratio, but it would likely be undesirable for a user to experience consecutive or clustered failure events. Fig.5 gives an example of failure events. In this subsection we consider the same $\mathbf{q} = (0.8, 0.6, 0.4)$ used above and explore the clustering of failures under two $\mathbf{w}$-LDF policies.

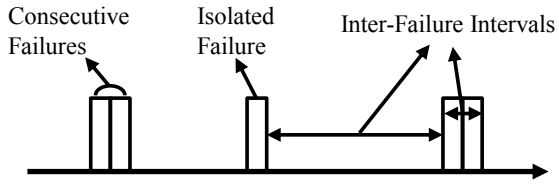

Fig. 5. Characteristics of clustering of failures.

We consider Inter-Failure Intervals (IFIs) between typical failures. IFI is supported on the set $\{1, 2, 3, \cdots\}$. To quantitatively evaluate the clustering of the failures, we focus on the standard deviation (SD) of the IFIs for each user. One extreme case is that failures happen strictly periodically and therefore, the SD is 0. Intuitively, a user with a smaller IFI SD implies that the user experiences less clustered failures.

Next we introduce an evaluation benchmark. For each user i, we know $1-p_i$ represents the time-averaged failure ratio. If the failure happens in each period independently with probability $1-p_i$, the IFI can be modeled by a geometric random variable supported on the set $\{1, 2, 3, \cdots\}$ with the parameter being $1 - p_i$. We use the SD of such a geometric random variable as a benchmark.

Under some $\mathbf{w}$-LDF policy, we define *SD ratio* of user i to be the ratio of user i's IFI SD to the SD of the geometric random variable with parameter $1 - p_i$. Table III shows the SD ratios of three users under two different weight vectors $\mathbf{w}$. Under $\mathbf{w} = (1, 1, 1)$, the ratios are less than 1, indicating that the failures under the LDF policy are less clustered compared to the scheme where failure event happens i.i.d. in each period. The last two columns in Table III indicates that increasing the weight of user i reduces the degree of failure clustering for user i but at the price of other users' more clustered failures. Thus, the users' sensitivities to clustered failures is another factor to consider when one assigns weights to users.

TABLE III
CHARACTERIZATION OF CLUSTERING OF FAILURES.

	SD ratio under $\mathbf{w} = (1, 1, 1)$	**SD ratio under** $\mathbf{w} = (10, 1, 1)$
User 1	88%	39%
User 2	77%	97%
User 3	92%	107%

VII. CONCLUSION

Resource allocation in complex systems supporting real-time users with general QoS requirements can be "easy". One can in principle design the system to allow priority-based resource allocation and adopt simple $\mathbf{w}$-LDF policies to dynamically prioritize users/applications. Our theory provides guidance towards understanding the suboptimality and even optimality of such solutions and how to improve the system design. For future work, it would be interesting to explore the management of real-time users across systems and/or sharing with non real-time traffic.

REFERENCES

[1] A. Dimakis and J. Walrand, "Sufficient Conditions for Stability of Longest-Queue-First Scheduling: Second-Order Properties Using Fluid Limits," *Advances in Applied Probability*, vol. 38, no. 2, June 2006.

[2] C. Joo, X. Lin, and N. B. Shroff, "Performance Limits of Greedy Maximal Matching in Multi-hop Wireless Networks," in *IEEE Conference on Decision and Control*, 2007, pp. 1128–1133.

[3] X. Kang, W. Wang, J. J. Jaramillo, and L. Ying, "On the Performance of Largest-Deficit-First for Scheduling Real-Time Traffic in Wireless Networks," in *Proc. of MobiHoc 2013*.

[4] I.-H. Hou and P. R. Kumar, "Queueing systems with hard delay constraints: a framework for real-time communication over unreliable wireless channels," *Queueing Systems*, vol. 71, 2012.

[5] ——, *Packets with Deadlines: A Framework for Real-Time Wireless Networks*. Morgan & Claypool Publishers, May 2013.

[6] ——, "Scheduling Heterogeneous Real-Time Traffic over Fading Wireless Channels," *IEEE Trans. on Networking*, vol. 22, 2014.

[7] S. Munir *et al.*, "Addressing burstiness for reliable communication and latency bound generation in wireless sensor networks," in *IPSN*, 2010.

[8] J. J. Jaramillo and R. Srikant, "Optimal Scheduling for Fair Resource Allocation in Ad Hoc Networks With Elastic and Inelastic Traffic," *IEEE Trans. on Networking*, vol. 19, 2011.

[9] L. Tassiulas and A. Ephremides, "Stability Properties of Constrained Queueing Systems and Scheduling Policies for Maximum Throughput in Multihop Radio Networks," *IEEE Trans. on Automatic Control*, 1992.

[10] ——, "Dynamic Server Allocation to Parallel Queues with Randomly Varying Connectivity," *IEEE Trans. on Information Theory*, 1993.

[11] N. McKeown *et al.*, "Achieving 100% Throughput in an Input-Queued Switch," *IEEE Trans. on Communications*, 1999.

[12] A. L. Stolyar, "Maxweight Scheduling in a Generalized Switch: State Space Collapse and Workload Minimization in Heavy Traffic," *The Annals of Applied Probability*, vol. 14, no. 1, February 2004.

[13] D. Down and S. Meyn, "A Survey of Markovian Methods for Stability of Networks," in *11th International Conference on Analysis and Optimization of Systems*, 1994.

[14] ——, "Piecewise linear test functions for stability and instability of queueing networks," *Queueing Systems*, 1997.

[15] L. Sha *et al.*, "Real Time Scheduling Theory: A Historical Perspective," *Real-Time Systems*, 2004.

[16] R. I. Davis and A. Burns, "A Survey of Hard Real-Time Scheduling for Multiprocessor Systems," *ACM Computing Surveys*, vol. 43, 2011.

[17] Extended version. [Online]. Available: http://arxiv.org/abs/1601.06331

[18] Personal communication with Alan Gatherer.