Scheduling for Cloud-Based Computing Systems to Support Soft Real-Time Applications

YUHUAN DU and GUSTAVO DE VECIANA, The University of Texas at Austin

Cloud-based computing infrastructure provides an efficient means to support real-time processing workloads, e.g., virtualized base station processing, and collaborative video conferencing. This paper addresses resource allocation for a computing system with multiple resources supporting heterogeneous soft real-time applications subject to Quality of Service (QoS) constraints on failures to meet processing deadlines. We develop a general outer bound on the feasible QoS region for non-clairvoyant resource allocation policies, and an inner bound for a natural class of policies based on dynamically prioritizing applications' tasks by favoring those with the largest (QoS) deficits. This provides an avenue to study the efficiency of two natural resource allocation policies: (1) priority-based greedy task scheduling for applications with variable workloads, and (2) priority-based task selection and optimal scheduling for applications with deterministic workloads. The near-optimality of these simple policies emerges when task processing deadlines are relatively large and/or when the number of compute resources is large. Analysis and simulations show substantial resource savings for such policies over reservation-based designs.

CCS Concepts: • **Computer systems organization** → **Real-time system architecture**; • **Networks** → *Cloud computing*; • **Theory of computation** → *Scheduling algorithms*;

Additional Key Words and Phrases: Soft real-time applications, cloud-computing, non-clairvoyant resource allocation, feasibility region, largest deficit first, greedy task scheduling, task selection and optimal scheduling, efficiency ratio

ACM Reference format:
Yuhuan Du and Gustavo de Veciana. 2017. Scheduling for Cloud-Based Computing Systems to Support Soft Real-Time Applications. *ACM Trans. Model. Perform. Eval. Comput. Syst.* 1, 1, Article 1 (March 2017), 31 pages.
DOI: 10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

The shift towards delivering compute platforms/services via cloud-based infrastructure is well on its way. An increasing number of the applications/services migrating to the cloud involve real-time computation with processing deadlines and where failure to meet the deadlines degrades user's Quality of Service (QoS). Such infrastructure allows one to reap the significant benefits of cloud computing, e.g., reduced cost of sharing computing, hoteling and cooling resources, along with increased reliability and energy efficiency. In this paper,

This research was supported by Huawei Technologies Co. Ltd. We thank Alan Gatherer for his support and feedback on this work.

DOI: 10.1145/nnnnnnn.nnnnnnn

we focus on Soft Real-Time (SRT) applications which can tolerate occasional violations of processing deadlines but still need to meet QoS or Service Level Agreements (SLA).

An example of such a platform is the Cloud-based Radio Access Network (CRAN) (Bernardos et al. 2014; China Mobile 2011; Du and de Veciana 2014) being considered for next generation cellular deployments. Instead of co-locating dedicated compute resources next to base station antennas, they virtualize compute resources for baseband processing. To do so, the received uplink signals associated with wireless subframes are sampled and sent from antennas to the cloud for timely decoding and processing such that downlink signals requiring timely channel measurements, acknowledgements, etc., can be sent back to antennas for transmission. This process must happen within several milliseconds as determined by the cellular system standards. In this setting shared compute resources may occasionally fail to complete subframe processing on time, but this must happen infrequently, i.e., QoS/SLA requirements must be met. In fact, different tasks may have different QoS/SLA requirements. For example, failures in subframe baseband processing should be very infrequent whereas failures for tasks associated with channel measurement/estimation might be acceptable once every few subframes (Gatherer 2015). Other SRT applications including multi-party collaborative video conferencing, multimedia processing, real-time control systems, augmented reality platforms, etc., have similar characteristics.

The computing infrastructure, e.g. (Verma et al. 2015), to support such applications may involve a large number of heterogeneous servers, e.g., various generations of processors, which themselves have multiple cores, special purpose hardware, shared memories/caches, etc. In other words, a complex collection of resources must be orchestrated to efficiently meet applications' SRT requirements. In this paper we focus on a single computing system, e.g., managed server/center, shared by a set of users, corresponding to SRT applications, that periodically generate workloads. The traditional management approach is to allocate dedicated resources to users to meet their QoS requirements. However, given the typical uncertainty in users' workloads and "interference" across shared resources, doing so typically involves over-provisioning.

Computing systems today are engineered so as to permit prioritization of one user over another, e.g., production vs. non-production tasks, which in turn translates to priority in accessing shared compute resources and/or memory. In this paper we consider resource allocation policies which can *dynamically* prioritize users in each period. Such dynamic prioritization of users would typically reduce the required resources vs. static allocations, and is further flexible to changes in users' workload characteristics or QoS requirements.

Given a set of users and a computing system, here are some key questions of interest:

- What QoS requirements are feasible?
- Can we design simple efficient resource allocation policies meeting users' QoS requirements and characterize the performance of these policies?
- Compared with dedicated resource allocation, what kinds of reductions in resource requirements can one expect from enabling dynamic resource sharing?

In the sequel we will address these basic questions and more, but we first turn to related work.

Related Work. There is a substantial body of work on scheduling real-time tasks. Starting with (Liu and Layland 1973), the community has established theoretical frameworks to study the scheduling of real-time applications where tasks are subject to hard deadlines, see e.g., (Carpenter et al. 2004; Davis and Burns 2011a; Leung 1989; Liu 2000). The results

typically assume worst case execution times/workloads and are too conservative for SRT applications.

Different models have been introduced for the QoS needs of Soft Real-Time (SRT) applications. The work in (Bernat and Burns 1997; Hamdaoui and Ramanathan 1995; Ramanathan 1999) proposes the notion of (m, k)-firm deadlines requiring at least m out of any k consecutive tasks complete by their deadlines. But many services do not need such tight requirements and the analytical results typically require deterministic workloads. The authors in (Hou and Kumar 2013; Liu et al. 1987) consider imprecise computation models where each task consists of a mandatory part, which needs to complete by the deadline, and an optional part which improves the computational results. This is a reasonable model for tasks like artificial intelligence computation since additional optional iterations improve the results. However, many real-time tasks do not contain optional part and some of these tasks can miss the deadlines up to some degree. The work in (Liu and Anderson 2009) aims to guarantee bounded maximum deadline tardiness for all users. However, these frameworks and QoS models are not suitable for applications like CRAN and video conferencing where it is useless to process a task after its deadline and it is better to simply drop the task if it misses the deadline.

This paper focuses on an SRT QoS model where a bound on the fraction of tasks completed on time is the QoS requirement. Such a model was first introduced in (Atlas and Bestavros 1998) where the authors propose a static allocation approach to meet such a QoS requirements. We shall use this as an evaluation benchmark. More recently, the authors in (Hou and Kumar 2012a, 2013) adopt this QoS model to study a wireless access point supporting users that periodically generate packets which need to be transmitted within that period, and propose simple "optimal" scheduling policies. However, their results are limited to the setting where only one user can transmit at a time and where packet transmissions can be viewed as tasks with geometrically distributed workloads.

In this paper we consider prioritization policies that use the idea underlying longest-queue-first policies, whose performance has been studied in (Dimakis and Walrand 2006; Joo et al. 2007a; Kang et al. 2013) but in different settings. Moreover, the scheduling problem we consider is more than just one of ordering users according to a policy such as largest-deficit-first. We also need to design the task scheduler to allocate resources to tasks across a computing system's cores.

Work on stochastic scheduling, e.g., (Ahmadizar et al. 2010; Allahverdi and Sotskov 2003; Blazewicz et al. 1986; Bruno et al. 1974; Lawler et al. 1993; Pinedo 2012) considers how to schedule a set of tasks with random workloads on multiple cores and aims to find a single schedule to minimize some objective function. Most of this type of work does not consider task completion deadlines and focuses on minimizing the expected completion time of the last task or the average expected completion time of all tasks. Moreover, such work typically assumes exponential workloads in order to get analytical results.

Additional related work include those studying the mixing of real-time and non real-time traffic, see e.g., (Jaramillo and Srikant 2011; Patil and de Veciana 2007; Shakkottai and Stolyar 2001), and those studying user/job management, see e.g., (Amir et al. 2000; Delimitrou and Kozyrakis 2014; Mars et al. 2011).

Our Contributions. In this paper, we consider a computing system consisting of multiple resources and study the scheduling of SRT users' random workloads subject to QoS constraints on timely task completions. To our knowledge, we are the first to give a theoretical characterization of the feasibility region for this general SRT framework and

to consider performance and near-optimality of simple efficient scheduling policies. The contributions of this paper are threefold.

First, we propose a general framework for SRT user scheduling on multiple resources, albeit we assume the workloads are New Better than Used in Expectation (NBUE) type. In this framework, we develop an outer bound for the set of feasible QoS requirements for all possible non-clairvoyant resource allocation policies.

Second, we study resource allocation policies which prioritize users based on Largest "Deficit" First (LDF) in each period and schedule tasks accordingly. We develop a general inner bound for the feasibility region for this class of policies. This enables us to study the efficiency of two policies: (1) LDF-based greedy task scheduling for users with variable workloads, and (2) LDF-based task selection and optimal scheduling for users with deterministic workloads. These simple policies are near-optimal when the deadlines are relatively large, and/or the number of resources is large.

Finally, we evaluate the performance of the proposed policies in terms of the required number of resources to fulfill a given set of users' QoS requirements. We exhibit substantial savings versus a traditional reservation-based approach in various system settings. We also discuss generalizations of our results when the resources have different processing speeds.

Paper Organization. The paper is organized as follows: Section 2 introduces our system model and Section 3 describes a reservation-based approach and a general outer bound for the feasibility region. Section 4 discusses two prioritization-based policies and studies their efficiency ratios. Simulation results are exhibited in Section 5. Section 6 discusses generalizations and Section 7 concludes the paper. Some of the proofs are provided in the Appendix.

2 SYSTEM MODEL

We first introduce our user, system and QoS models.

2.1 Soft Real-Time (SRT) User Model

We consider a computing system shared by a set of users $N = \{1, 2, \cdots, n\}$. The system operates over discrete periods $t = 1, 2, \cdots$. We denote by δ the length of a period. In each period each of the n users generates exactly one task. These tasks are available for processing at the beginning of the period, and need to complete by the end of the period. Tasks not completed on time are dropped, i.e., cannot be processed in subsequent periods. Here we assume a task is the unit of scheduling, i.e., a task cannot be processed in parallel.

The workload of a task will refer to its resource requirement or service time. If a task's workload is large it may not be possible to complete on time. A task's workload is modeled by a random variable whose distribution captures variability in its resource requirement and/or uncertainty in the computing system, e.g., caused by memory contention across the cores. We assume task workloads for a given user are independent and identically distributed (i.i.d.) across periods and workloads from different users are independent, possibly with different distributions. Let W_i be a random variable denoting the workload of a task from user i and let $\mu_i = \mathrm{E}[W_i]$. Next we introduce a further assumption on task workloads which seems reasonable for SRT users and will enable theoretical analysis.

Definition 2.1. A non-negative random variable W is said to satisfy **New Better than Used in Expectation (NBUE)** if for all $t > 0$,

$$\mathrm{E}[W - t | W > t] \leq \mathrm{E}[W]. \tag{1}$$

In this paper we shall assume all task workloads are NBUE.

The NBUE property characterizes many workload distributions of interest. (Müller and Stoyan 2002) provides a discussion of NBUE distributions which include, but are not limited to, exponential, gamma with shape parameter $k \geq 1$ and deterministic distributions. A common class of distributions that are not NBUE is the heavy-tailed one. However since tasks need to complete within a period[1], we are not likely to encounter tasks with such tails in the settings under consideration.

We shall assume that each user i has a QoS requirement given by a minimal long-term average number of tasks completed on time per period, denoted by q_i where $q_i \in [0, 1]$. We let $\mathbf{q} = (q_1, q_2, \cdots, q_n)$ and assume q_i's are rational[2].

Let us consider some examples. An SRT user might correspond to the processing associated with a set of co-located cellular antennas in the CRAN context or an end user in video conferencing. Accordingly, the period δ would correspond to a wireless subframe or the length of a group of video frames, respectively. For SRT users, it is generally useless to process a task after its deadline. For example, in video conferencing it is not desirable to display an out-of-date frame. This is why in this model tasks not completed on time are dropped. In Section 6, we discuss possible generalizations where users may generate tasks with different periods and where a task may further consists of sub-tasks.

2.2 Computing Infrastructure and Space of Policies

A computing system can be very complex consisting of diverse, heterogeneous resources. In this paper, for simplicity of explanation we start with a computing system comprising of m identical resources (cores)—a simple but relevant model. In Section 6 we discuss generalizations where cores have different processing speeds.

Given m identical cores, a task processed on any core requires the same processing time and each core can process only one task at a time. In each period, the computing system dynamically schedules tasks according to a given strategy. Given the resource limit and the randomness of workloads, some tasks complete on time and some may fail.

In this paper we only consider stationary[3] *non-clairvoyant* resource allocation policies. A resource allocation policy is said to be *non-clairvoyant* if it does not make use of information regarding future events, such as tasks' workload realizations, which are not generally known until the tasks complete. However, a non-clairvoyant resource allocation policy may still have knowledge of a user's task workload distribution, which can be obtained from the history of events or repeated experiments.

Unless otherwise specified we allow task preemption/migration, i.e., interrupting a task being processed and resuming later on the same/different core. We shall ignore the overheads of these operations. But in practice these operations involve context switching, and therefore, policies with minimal preemption and migration are desirable. There are many possible non-clairvoyant resource allocation policies which may involve exploiting knowledge of workload distributions, exploiting history of events, preempting tasks at appropriate times, dynamically prioritizing tasks, etc.

In our model a "core" represents the minimum unit of compute resource such as physical computing core, specialized hardware, or hyper-thread as appropriate. The computing

[1] In fact, we only require (1) to be true for $0 < t \leq \delta$.

[2] All the results in this paper can be generalized to $\mathbf{q}$'s with irrational values. For simplicity in the proof we do not consider that level of generality.

[3] A policy is stationary if the system is ergodic under this policy.

system could be a cloud-based cluster of machines or a centralized server with a collection of processors/cores.

2.3 SRT QoS Feasibility

Given a requirement vector $\mathbf{q}$, a computing system and a non-clairvoyant resource allocation policy, how do we verify if $\mathbf{q}$ is feasible? We shall say $\mathbf{q}$ is feasible if the requirement vector $\mathbf{q}$ is dominated by the "service rate" vector. More formally, to keep track of the deficit among users' QoS requirements and actually completed tasks, for each user $i \in N$ and period $t+1$, we define[4]

$$X_i(t+1) = [X_i(t) + q_i - Y_i(t+1)]^+, \tag{2}$$

where $[x]^+ = \max[x, 0]$ and $Y_i(t+1)$ is an indicator random variable which takes value 1 if user i's task completes in period $t+1$. We let $\mathbf{X}(t) = (X_1(t), X_2(t), \cdots, X_n(t))$ denote the deficit vector. $\mathbf{X}(t)$ is a summary of the history of events up to period t.

We shall say that the long-term QoS requirement q_i for user i is met if and only if $X_i(t)$ is "stable"[5] . Formally, in this paper we consider non-clairvoyant resource allocation policies under which the process $\{\mathbf{X}(t)\}_{t\geq 1}$ is a Markov chain[6]. We assume the initial state $\mathbf{X}(0)$, the QoS requirements $\mathbf{q}$ and the policy make $\{\mathbf{X}(t)\}_{t\geq 1}$ an irreducible Markov chain.

Definition 2.2. We say the QoS requirement vector $\mathbf{q}$ is **feasible** if there exists a non-clairvoyant resource allocation policy η under which the Markov chain $\{\mathbf{X}(t)\}_{t\geq 1}$ is positive recurrent, i.e., this policy fulfills $\mathbf{q}$. We denote by F_η the feasibility region of policy η, i.e., the set of QoS requirement vectors fulfilled by policy η. The union of F_η over all allowable policies gives the **system feasibility region** F.

We shall refer to this model as SRT-Multiple Identical Cores (SRT-MIC) with NBUE workloads and the aim is to devise non-clairvoyant resource allocation policies that fulfill $\mathbf{q}$.

In summary, the SRT-MIC model with NBUE workloads is an abstract system model which captures a family of systems supporting SRT users with random workloads. To summarize, the SRT-MIC model with NBUE workloads is parameterized by the number of cores m, number of users n, period length δ, QoS requirements $\mathbf{q}$, and the NBUE workload distributions.

3 RESERVATION-BASED STATIC SHARING AND OUTER BOUND FOR THE SYSTEM FEASIBILITY REGION

Clearly simple policies like Earliest Deadline First (EDF) do not apply in our setting. Indeed in our problem statement all users generate tasks which have the same deadline at the start of the scheduling interval. In fact in the sequel (see Section 6) we will see that even if users generate tasks with different deadlines EDF performs poorly because it does not take the soft QoS requirements $\mathbf{q}$ into account.

[4]We truncate the deficit at 0 via $[x]^+$ simply for the convenience of defining feasibility. Removing the truncation does not change the results in the paper.

[5] Our QoS model is essentially the same as those used in prior work (Atlas and Bestavros 1998; Dimakis and Walrand 2006; Hou and Kumar 2012a, 2013; Joo et al. 2007a).

[6] All the results in this paper can be generalized to a broader range of non-clairvoyant resource allocation policies under which some variation of $\mathbf{X}(t)$ is a Markov chain. For example, if a resource allocation policy depends on the deficit vectors in the past two periods, then $\{(\mathbf{X}(t), \mathbf{X}(t+1))\}_{t\geq 1}$ is a Markov chain. For simplicity of explanation, we assume $\{\mathbf{X}(t)\}_{t\geq 1}$ is a Markov chain.

In this section we introduce a reservation-based policy and a general outer bound for the system feasibility region F which applies to any non-clairvoyant resource allocation policy. These serve as benchmarks which enable us to evaluate the performance of the policies proposed in the sequel.

3.1 Reservation-Based Static Sharing Policies

A straightforward and commonly adopted approach to meet users' QoS requirements $\mathbf{q}$ is to allocate dedicated resources, i.e., core time, to each user. For user i, with task workload W_i and the requirement q_i, we let $w_i(q_i)$ represent the minimum core time reservation needed to ensure the requirement is met. Specifically, $w_i(q_i)$ is given by

$$w_i(q_i) = \min_w \{w | \Pr(W_i \leq w) \geq q_i\},$$

and thus, when q_i is close to 1, $w_i(q_i)$ will approach the worst-case workload for user i.

Reservation-based static sharing policies allocate core time $w_i(q_i)$ to each user i in each period and the tasks from users are only processed in the corresponding allocated time. Figure 1 exhibits an example with 2 cores. Note that in this example User 3's task first executes on Core 2 and later continues on Core 1. Therefore, a reservation-based static sharing policy, although seemingly simple, can be aggressive in requiring task preemption/migration and knowledge of workload distributions to compute $w_i(q_i)$ for all users.

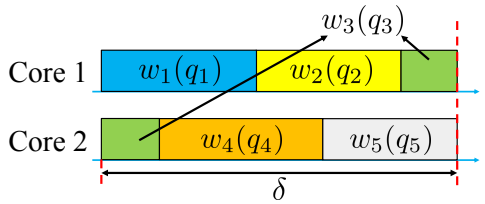

Fig. 1. An example of the reservation-based approach.

Note that since a task cannot be processed in parallel, if $w_i(q_i)$ exceeds the period length δ, the requirement for user i cannot be met. In this paper, we assume the task workloads and requirements $\mathbf{q}$ are such that $w_i(q_i)$ is bounded by δ.

For a system with m identical cores, the feasibility region F_{RB} of reservation-based static sharing is given by

$$F_{\text{RB}} = \{\mathbf{q} \in \mathbb{R}_+^n \mid \mathbf{q} \preceq \mathbf{1}, \sum_{i \in N} w_i(q_i) \leq m\delta, \\ w_i(q_i) \leq \delta, \forall i \in N\}, \tag{3}$$

where $\mathbf{q} \preceq \mathbf{1}$ means $q_i \leq 1$ for all $i \in N$. Clearly $\mathbf{q} \preceq \mathbf{1}$ comes from the fact that each user generates only one task in each period.

This approach was perhaps first proposed in (Atlas and Bestavros 1998) and is also loosely used in reservation based schemes adopted in modern cloud infrastructure, see e.g., (Verma et al. 2015). Cores are not used efficiently under such a policy. When the realization of a task workload is smaller than the allocated time, the remaining time is wasted and cannot be used to process other real-time tasks. Typically , e.g. (Verma et al. 2015), the resources are then used to support best effort traffic.

3.2 Outer Bound for the System Feasibility Region F

Ideally we aim to devise a policy that can fulfill all feasible QoS requirement vectors. More formally, a non-clairvoyant resource allocation policy η is said to be *feasibility optimal* if its feasibility region F_η is such that $F \subseteq \text{cl}(F_\eta)$, where $\text{cl}(F_\eta)$ is the closure of F_η. We know by definition of F that $\text{int}(F_\eta) \subseteq F$ where $\text{int}(F_\eta)$ is the interior of F_η, and thus F_η is for practical purposes equivalent to the system feasibility region F.

Given the heterogeneity and randomness of tasks' workloads and the large number of possible non-clairvoyant resource allocation policies, a feasibility optimal policy is unknown except for very specific resource and workload models, see e.g., (Hou and Kumar 2012b). To solve this and to provide a benchmark to evaluate other resource allocation policies, we develop a simple outer bound R_{OB} for the system feasibility region F. Formally, we have the following theorem.

THEOREM 3.1. *For the SRT-MIC model with NBUE workloads, the system feasibility region F is such that*

$$F \subseteq R_{\text{OB}} \equiv \{\mathbf{q} \in \mathbb{R}_+^n \mid \mathbf{q} \preceq \mathbf{1}, \sum_{i \in N} q_i \mu_i \leq m\delta\}.$$

Intuitively, if q_i tasks of user i are completed each period, the expected time spent on user i is roughly given by $q_i \mu_i$. To make $\mathbf{q}$ feasible, the total time spent on all users $\sum_{i \in N} q_i \mu_i$ cannot exceed the total available core time given by $m\delta$. This informal argument is perhaps deceptive. Note that in fact the expected time to complete the q_i tasks for user i in each period might be smaller than $q_i \mu_i$ since completed tasks might tend to have smaller workloads. This seems to imply that $m\delta$ could be smaller than $\sum_{i \in N} q_i \mu_i$ for some feasible $\mathbf{q}$. This is where the NBUE assumption on workloads is critical to the result.

Note this simple outer bound applies only to non-clairvoyant resource allocation policies for a specific SRT-MIC system with NBUE workload distributions. A formal proof of the theorem is given below.

PROOF. Given a feasible QoS requirement vector $\mathbf{q} \preceq \mathbf{1}$, the goal is to show $\sum_{i \in N} q_i \mu_i \leq m\delta$.

Suppose $\mathbf{q}$ is fulfilled by a non-clairvoyant resource allocation policy η, by definition $\{\mathbf{X}(t)\}_{t \geq 1}$ is positive recurrent and therefore, there exists a stationary distribution. We consider a typical period where the deficit vector $\mathbf{X}(t)$ follows the stationary distribution and introduce further notation associated with period $t+1$. To simplify notation, we will suppress the period index in this proof.

For each user i, we define Y_i to be the indicator random variable that the task from user i completes in a typical period. By the Ergodic Theorem, $\text{E}[Y_i]$ also represents the time-averaged number of task completions per period for user i. If we view $X_i(t)$ as a queue, the average arrival q_i should not exceed the average departure $\text{E}[Y_i]$. For each user subset $S \subseteq N$, we define U_S to be a random variable denoting the total core time spent on users in S in a typical period. Clearly, $\text{E}[U_S]$ cannot exceed the total available core time $m\delta$. To show $\sum_{i \in N} q_i \mu_i \leq m\delta$, it suffices to show that $\sum_{i \in N} \text{E}[Y_i]\mu_i \leq \text{E}[U_N]$. To that end we first develop an equation connecting $\sum_{i \in N} \text{E}[Y_i]\mu_i$ and $\text{E}[U_N]$, and then use the NBUE assumption to show the inequality.

We say a task is *unfinished* if it starts processing but does not complete in a given period. Let A_i be the indicator random variable that user i's task is unfinished in a typical period.

Now if $Y_i + A_i = 1$ it indicates that user i's task starts processing in the period though it may not have completed. For each user i, we further define $E_i = A_i(W_i - U_{\{i\}})$. Intuitively, E_i represents the "residual workloads for user i's unfinished tasks". Note that these random variables and their means depend on the policy η.

For each user subset $S \subseteq N$, the total time spent on users in S can be written as

$$U_S = \sum_{i \in S} (Y_i + A_i) W_i - \sum_{i \in S} E_i,$$

and by taking expectations, we get

$$\mathrm{E}[U_S] = \sum_{i \in S} \mathrm{E}[(Y_i + A_i) W_i] - \sum_{i \in S} \mathrm{E}[E_i]. \tag{4}$$

Clearly $Y_i + A_i$, which indicates that user i's task starts processing, is independent of W_i. Indeed this follows from the requirement that the resource allocation policy be non-clairvoyant, and the independence among users' task workloads. In a typical period under policy η, the event that user i's task starts may depend on the workloads of others' tasks, but not on W_i.

Note that although $Y_i + A_i$ is independent of W_i, in general Y_i which indicates user i's task completes may depend on W_i, i.e., $\mathrm{E}[Y_i W_i] \neq \mathrm{E}[Y_i]\mu_i$. To better understand this, consider an extreme example. If $W_i > \delta$, clearly the user i's task cannot complete implying that $Y_i = 0$. Thus, $\mathrm{E}[Y_i | W_i > \delta] = 0 \neq \mathrm{E}[Y_i]$. Similarly, we can argue A_i is not independent of W_i.

Still given the independence of $Y_i + A_i$ and W_i, we have that

$$\mathrm{E}[(Y_i + A_i) W_i] = \mathrm{E}[Y_i + A_i] \cdot \mathrm{E}[W_i] = (\mathrm{E}[Y_i] + \mathrm{E}[A_i])\mu_i.$$

So (4) becomes

$$\mathrm{E}[U_S] = \sum_{i \in S} \mathrm{E}[Y_i]\mu_i + \sum_{i \in S} \mathrm{E}[A_i]\mu_i - \sum_{i \in S} \mathrm{E}[E_i]. \tag{5}$$

This equation holds for all non-clairvoyant resource allocation policies and for all subsets of users $S \subseteq N$.

Now let $S = N$. To show $\sum_{i \in N} \mathrm{E}[Y_i]\mu_i \leq \mathrm{E}[U_N]$, by (5) it suffices to show $\mathrm{E}[A_i]\mu_i \geq \mathrm{E}[E_i]$ for all users $i \in N$. We will show this is true under the NBUE workload assumption in the discrete-time scenario and it is straightforward to generalize the proof to the continuous-time scenario.

Recall that we denote by δ the length of a period. Suppose each period contains δ discrete time units. For all i and for $c = 1, 2, \cdots, \delta$, we let $A_{i,c}$ denote the indicator random variable that user i's task is unfinished and is processed for c time units in a typical period. Clearly, $A_i = \sum_{c=1}^{\delta} A_{i,c}$ and $\mathrm{E}[A_{i,c}] = \Pr(A_{i,c} = 1)$. By the law of total probability, the expected residual workload $\mathrm{E}[E_i]$ for user i can be written as

$$\mathrm{E}[E_i] = \sum_{c=1}^{\delta} \mathrm{E}[E_i | A_{i,c} = 1] \Pr(A_{i,c} = 1) = \sum_{c=1}^{\delta} \mu_{i,c} \mathrm{E}[A_{i,c}], \tag{6}$$

where $\mu_{i,c} = \mathrm{E}[W_i - c | W_i > c]$. This is because under the non-clairvoyant design the event $A_{i,c} = 1$ tells nothing about W_i except that $W_i > c$.

By the NBUE workload assumption we know that $\mu_{i,c} \leq \mu_i$ for $c > 0$ and therefore, we get the following inequality,

$$\mathrm{E}[E_i] \leq \sum_{c=1}^{\delta} \mu_i \, \mathrm{E}[A_{i,c}] = \mu_i \, \mathrm{E}[A_i]. \tag{7}$$

Note that the equality holds if all users' task workloads follow geometric distributions (or exponential distributions in continuous-time scenario), possibly with different parameters.

To summarize, by (5) and (7) we know that given a feasible requirement vector $\mathbf{q}$, for all user subsets $S \subseteq N$,

$$\sum_{i \in S} q_i \mu_i \leq \sum_{i \in S} \mathrm{E}[Y_i] \mu_i \leq \mathrm{E}[U_S] \leq m\delta, \tag{8}$$

which by letting $S = N$ implies $\sum_{i \in N} q_i \mu_i \leq m\delta$, and thus, $F \subseteq R_{\mathrm{OB}}$. □

A key part of this argument is the inequality (8), stating that for a feasible $\mathbf{q}$ the "effective" workload $\sum_{i \in S} q_i \mu_i$ for any user subset S should not exceed the total time spent on users in S, which is bounded by $m\delta$. This holds under the NBUE workload assumption but may not be true if users have non-NBUE task workloads. Intuitively, for non-NBUE workloads it is possible to design policies that exploit the workload distributions, e.g., by stopping at different times to maximize the "payoffs", and to achieve a larger feasibility region. For example, suppose all users generate tasks with non-NBUE workloads as follows,

$$W_i = \begin{cases} 1 & \text{with probability } 0.5 \\ 9 & \text{with probability } 0.5. \end{cases}$$

Clearly, the mean workload is $\mu_i = 5$. Let us consider such a policy. In each period, the system processes each task for exactly 1 time unit and stops if the task does not complete because given its workload distribution we know this task will require 8 more time units to complete. Suppose m and δ is such that $m\delta = n$ and therefore, the system can process each task for 1 time unit per period. Under such a policy we know $q_i = 0.5$ for all user i and the total time spent per period is $U_N = n$. Therefore,

$$\sum_{i \in N} q_i \mu_i = 2.5n > n = \mathrm{E}[U_N] = m\delta,$$

which is not consistent with (8) and Theorem 3.1. Non-NBUE workloads are beyond the scope of this paper. Yet for real-time computing workloads we expect NBUE to be a good assumption.

4 LARGEST DEFICIT FIRST (LDF) BASED POLICIES

Our aim is to devise a non-clairvoyant resource allocation policy that is easy to implement and whose feasibility region is near optimal. In this section we consider a specific class of policies, called *prioritization-based resource allocation* policies, which decompose resource allocation into two sub-problems, see Figure 2:

(1) User prioritization: in each period the system dynamically prioritizes users based on the history of events.
(2) Task scheduler: the system schedules users' tasks on cores based on their priorities.

There are still many options for each sub-problem. For example, task scheduling might be done greedily by simply scheduling the task with the highest priority, or using the priorities to first select a subset of tasks and then process that task subset via optimal scheduling policies.

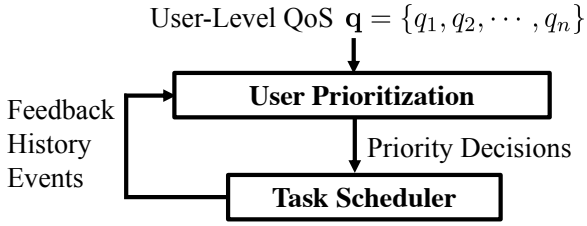

Fig. 2. The framework for prioritization-based resource allocation policies.

In this paper we shall prioritize users based on the Largest Deficit First (LDF) policy which is defined as follows.

We let $\mathbf{d} = (d_1, d_2, \cdots, d_n)$ denote a *priority decision* where d_k is the index of the user with k^{th} highest priority and D denote the set of all possible priority decisions.

Definition 4.1. The **Largest Deficit First (LDF)** policy is such that, given the users' deficit vector $\mathbf{X}(t)$, the priority decision $\mathbf{d}$ for period $t+1$ is such that

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

with ties broken arbitrarily (possibly randomly). In other words, it sorts the deficits and assigns priorities accordingly.

The LDF user prioritization can be combined with different approaches of task scheduling. In the sequel we will explore such combinations and characterize their performance.

4.1 Inner Bound for Feasibility Region of LDF+$\mathcal{X}$

Given a task scheduling policy $\mathcal{X}$, we let LDF+$\mathcal{X}$ refer to the resource allocation policy that combines LDF user prioritization and task scheduler $\mathcal{X}$. In this subsection, we provide an inner bound for its feasibility region $F_{\text{LDF}+\mathcal{X}}$.

We first introduce some further notation. Given a task scheduler, in each period, the task completions depend on the selected priority decision. We let $p_i(\mathbf{d})$ denote the expected number of tasks completed in a period for user i under priority decision $\mathbf{d}$ and let $\mathbf{p}(\mathbf{d}) = (p_1(\mathbf{d}), p_2(\mathbf{d}), \cdots, p_n(\mathbf{d}))$. Note that different task schedulers will correspond to different sets of vectors $P = \{\mathbf{p}(\mathbf{d}) | \mathbf{d} \in D\}$. We denote by $\mathbf{x} \succ \mathbf{0}$ a positive vector $\mathbf{x}$ with $x_i > 0$ for all $i \in N$. For all user subsets $S \subseteq N$, we let $|S|$ be the number of users in S and we let $D(S)$ denote the set of all priority decisions that assign the highest $|S|$ priorities to users in S. The following theorem gives an inner bound on $F_{\text{LDF}+\mathcal{X}}$.

THEOREM 4.2. *Given a task scheduler $\mathcal{X}$ and thus the $\mathcal{X}$ dependent expected completion vectors $P = \{\mathbf{p}(\mathbf{d}) | \mathbf{d} \in D\}$, an inner bound for the feasibility region of the resource allocation policy LDF+$\mathcal{X}$ is given by int$(R_{\text{IB}}) \subseteq F_{\text{LDF}+\mathcal{X}}$, where*

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

Intuitively, $\mathbf{q}$ is in R_{IB} and is feasible under the LDF+$\mathcal{X}$ policy if there is a weight vector $\boldsymbol{\alpha} \succ \mathbf{0}$ such that for any subset of users S, if the users in S are given the highest priorities, the weighted sum of the requirements $\sum_{i \in S} \alpha_i q_i$ does not exceed the least weighted sum of the "service rate" $\sum_{i \in S} \alpha_i p_i(\mathbf{d})$. Again, different task schedulers $\mathcal{X}$ will have different vectors P and thus different inner bounds R_{IB}. A proof is provided in Appendix 8.1. Note that Theorem 4.2 applies beyond the SRT-MIC model when the LDF policy is used but in a general setting where $\mathbf{p}(\mathbf{d})$ represent the expected payoffs under priority decision $\mathbf{d}$ and users require long-term time-averaged payoff $\mathbf{q}$ per period. The LDF policy can also be generalized to a class of weighted LDF policies. This general result is further developed in (Du and de Veciana 2016).

Next we explore specific task schedulers and use Theorem 4.2 to study their performance.

4.2 Performance Analysis of LDF+Greedy Scheduling

Given an LDF-based user priority decision in each period, a natural way to allocate resources is to greedily process tasks from highest to lowest priority. Specifically, to start by putting the m tasks with the highest priority on the m cores and, once one of these tasks completes, continue by processing the task with priority $m+1$ on the available core, etc.

We let *LDF+Greedy* refer to the resource allocation policy that combines LDF and such a greedy task scheduler. Note this is easy to implement and does not require any a-priori knowledge of the tasks' workloads. Also this policy does not use task preemption or migration.

Next we characterize the performance of LDF+Greedy. To that end, we introduce a metric called the efficiency ratio, see e.g., (Joo et al. 2007b). The *efficiency ratio* of a non-clairvoyant resource allocation policy η is defined as

$$\gamma_\eta = \sup_\gamma \{\gamma | \gamma F \subseteq F_\eta\}.$$

Clearly γ_η characterizes the performance gap between a policy η and the best possible way of orchestrating the scheduling of multiple tasks across multiple cores. Also γ_η equals to 1 if and only if policy η is feasibility optimal.

THEOREM 4.3. *For the SRT-MIC model with NBUE workloads, the efficiency ratio of LDF+Greedy exceeds γ_1 where*

$$\gamma_1 = 1 - \frac{\max_{i \in N} \mu_i}{\delta}.$$

Further, γ_1 is a tight lower bound in the sense that for any $\epsilon > 0$, there exists an SRT-MIC system with NBUE workloads such that the efficiency ratio of LDF+Greedy for this system is smaller than $\gamma_1 + \epsilon$.

The intuition underlying this result is as follows. We say a task is *unfinished* if it starts processing but does not complete in a period. The time spent on an unfinished task goes to waste since it does not contribute to a task completion. For LDF+Greedy, in one period, at most 1 task is unfinished per core and thus the wasted time on each core is expected to be less than $\max_{i \in N} \mu_i$. Given the period is of length δ, the gap between LDF+Greedy and optimality is bounded by $\frac{\max_{i \in N} \mu_i}{\delta}$. Note that again this argument is deceptively simplified since unfinished tasks might tend to have larger workloads. Also as for Theorem 3.1, this result does not necessarily hold for non-NBUE workloads. The formal proof is given below.

PROOF. First we show that γ_1 is a lower bound for the efficiency ratio of LDF+Greedy, denoted by $\gamma_{\text{LDF+Greedy}}$.

Given a requirement vector $\mathbf{q}$ fulfilled by resource allocation policy η, by (8) we know for all subsets of users $S \subseteq N$,

$$\sum_{i \in S} q_i \mu_i \leq \mathrm{E}[U_S],$$

where $\mathrm{E}[U_S]$ represents the time-averaged core time spent on users in S per period under policy η.

During each period, the total time U_S spent on users in S is bounded by the total task workload $\sum_{i \in S} W_i$ of users in S and the total available core time $m\delta$. We define $T_S = \min\left[\sum_{i \in S} W_i, m\delta\right]$ and therefore, for all user subsets S, we have that

$$\sum_{i \in S} q_i \mu_i \leq \mathrm{E}[U_S] \leq \mathrm{E}[T_S]. \tag{9}$$

Thus, for a vector $\mathbf{q}$ satisfying (9) the aim to show $\gamma_{\text{LDF+Greedy}} \geq \gamma_1$ which is equivalent to showing $\gamma_1 \mathbf{q} \in \mathrm{cl}(F_{\text{LDF+Greedy}})$. By Theorem 4.2, it suffices to show that $\gamma_1 \mathbf{q} \in R_{\text{IB}}$. In LDF+Greedy, the expected vector $\mathbf{p}(\mathbf{d})$ described in Section 4.1 represents the expected numbers of timely completions under greedy task scheduler under priority decision $\mathbf{d}$. Therefore, $\gamma_1 \mathbf{q} \in R_{\text{IB}}$ follows if one can find a vector $\boldsymbol{\alpha} \succ \mathbf{0}$ such that for all $S \subseteq N$,

$$\sum_{i \in S} \alpha_i \gamma_1 q_i \leq \min_{\mathbf{d} \in D(S)} \sum_{i \in S} \alpha_i p_i(\mathbf{d}).$$

We will show $\boldsymbol{\alpha} = (\mu_1, \mu_2, \cdots, \mu_n) \succ \mathbf{0}$ satisfies the above condition. Intuitively, this means for any subset of users S, if the users in S are given the highest priorities, then γ_1 times the "workload requirements" $\sum_{i \in S} \mu_i q_i$ does not exceed the least "workload weighted service rate" $\sum_{i \in S} \mu_i p_i(\mathbf{d})$.

By (9) it suffices to show for all S,

$$\gamma_1 \, \mathrm{E}[T_S] \leq \min_{\mathbf{d} \in D(S)} \sum_{i \in S} \mu_i p_i(\mathbf{d}),$$

which is equivalent to showing for any given user subset S and priority decision $\mathbf{d} \in D(S)$ that

$$\sum_{i \in S} \mu_i p_i(\mathbf{d}) \geq \gamma_1 \, \mathrm{E}[T_S] = \mathrm{E}[T_S] - \frac{\max_{i \in N} \mu_i}{\delta} \mathrm{E}[T_S]. \tag{10}$$

First we rewrite $\sum_{i \in S} \mu_i p_i(\mathbf{d})$ by similar approach used to obtain (5). Specifically, as in the proof of Theorem 3.1, for each subset of users $S \subseteq N$ and each user $i \in N$, we let $U_S(\mathbf{d})$, $A_i(\mathbf{d})$ and $E_i(\mathbf{d})$ denote the time spent on users in S, the indicator random variable that user i's task is unfinished and the residual workload of user i's unfinished tasks in a period under the greedy task scheduler with priority decision $\mathbf{d}$, respectively.

By (5), for the given S and $\mathbf{d}$, we have that

$$\mathrm{E}[U_S(\mathbf{d})] = \sum_{i \in S} p_i(\mathbf{d}) \mu_i + \sum_{i \in S} \mathrm{E}[A_i(\mathbf{d})] \mu_i - \sum_{i \in S} \mathrm{E}[E_i(\mathbf{d})].$$

Recall that intuitively this means for the any user subset S, the total time spent on S equals to "workloads of S's started tasks" minus the "residual workloads of S's unfinished tasks". Reordering the equations gives the following,

$$\sum_{i\in S} p_i(\mathbf{d})\mu_i = \mathrm{E}[U_S(\mathbf{d})] + \sum_{i\in S}\mathrm{E}[E_i(\mathbf{d})] - \sum_{i\in S}\mathrm{E}[A_i(\mathbf{d})]\mu_i.$$

Now (10) follows by showing that

$$\mathrm{E}[U_S(\mathbf{d})] + \sum_{i\in S}\mathrm{E}[E_i(\mathbf{d})] \geq \mathrm{E}[T_S] \tag{11}$$

and

$$\sum_{i\in S}\mathrm{E}[A_i(\mathbf{d})]\mu_i \leq \frac{\max_{i\in N}\mu_i}{\delta}\mathrm{E}[T_S], \tag{12}$$

respectively.

To demonstrate (11), it suffices to show for each workload realization,

$$u_S(\mathbf{d}) + \sum_{i\in S} e_i(\mathbf{d}) \geq t_S,$$

where $u_S(\mathbf{d}), e_i(\mathbf{d}), t_S$ are realizations of $U_S(\mathbf{d}), E_i(\mathbf{d}), T_S$, respectively.

If $u_S(\mathbf{d}) = m\delta$, clearly $u_S(\mathbf{d}) + \sum_{i\in S} e_i(\mathbf{d}) \geq m\delta \geq t_S$. Otherwise, $u_S(\mathbf{d}) < m\delta$. Since $\mathbf{d} \in D(S)$ assigns the highest priorities to users in S, by greedy task scheduler $u_S(\mathbf{d}) < m\delta$ implies that at the end of the period no task from users in S is waiting to be scheduled, i.e., all tasks from users in S start processing and therefore, $u_S(\mathbf{d}) + \sum_{i\in S} e_i(\mathbf{d}) \geq \sum_{i\in S} w_i \geq t_S$, where w_i represents the realization of workload W_i. Therefore, (11) is verified.

Now it remains to show (12). Clearly we have that

$$\sum_{i\in S}\mathrm{E}[A_i(\mathbf{d})]\mu_i \leq \max_{i\in N}\mu_i \cdot \sum_{i\in S}\mathrm{E}[A_i(\mathbf{d})].$$

Thus, to demonstrate (12) it suffices to show that

$$\sum_{i\in S}\mathrm{E}[A_i(\mathbf{d})] \leq \frac{\mathrm{E}[T_S]}{\delta}. \tag{13}$$

We define $A_S(\mathbf{d}) = \sum_{i\in S} A_i(\mathbf{d})$ to be the number of unfinished tasks in a period from users in S under greedy task scheduler under priority decision $\mathbf{d}$. Since there are at most m unfinished tasks, we have $A_S(\mathbf{d}) \leq m$.

Under greedy task scheduling, for $\mathbf{d} \in D(S)$ we claim $A_S(\mathbf{d}) = k$ implies $T_S \geq k\delta$ for $k = 0, 1, \cdots, m$. This is true because $A_S(\mathbf{d}) = k$ means there are k unfinished tasks on k different cores, implying these k cores are busy processing tasks from users in S throughout the period. Therefore, $\sum_{i\in S} W_i \geq k\delta$ and thus $T_S \geq k\delta$.

By this claim, we can get that

$$\begin{aligned}
\mathrm{E}[T_S] &= \sum_{k=0}^{m} \mathrm{E}[T_S | A_S(\mathbf{d}) = k] \cdot \Pr(A_S(\mathbf{d}) = k) \\
&\geq \sum_{k=0}^{m} k\delta \cdot \Pr(A_S(\mathbf{d}) = k) \\
&= \delta \sum_{k=0}^{m} k \cdot \Pr(A_S(\mathbf{d}) = k) \\
&= \delta \, \mathrm{E}[A_S(\mathbf{d})] \\
&= \delta \sum_{i \in S} \mathrm{E}[A_i(\mathbf{d})].
\end{aligned}$$

This proves (13) which in turn shows (10) and therefore, $\gamma_1 \mathbf{q} \in R_{\mathrm{IB}} \subseteq \mathrm{cl}(F_{\mathrm{LDF+Greedy}})$.

The lower bound γ_1 is indeed tight in the sense that for any $\epsilon > 0$, there exists an SRT-MIC system with NBUE workloads such that $\gamma_{\mathrm{LDF+Greedy}} < 1 - \frac{\max_{i \in N} \mu_i}{\delta} + \epsilon$. Such a system is detailed in Appendix 8.2.

□

It follows that if $\delta \gg \max_{i \in N} \mu_i$, then γ_1 is close to 1, i.e., LDF+Greedy is close to optimal. This is true when the task workloads are small relative to the core processing speed.

However, when δ is comparable to $\max_{i \in N} \mu_i$, the efficiency ratio lower bound γ_1 is small, although in some scenarios LDF+Greedy may still be efficient. For example, LDF+Greedy is feasibility optimal if the task workloads of all users follow the same exponential (or geometric) distribution, or in systems studied in prior work (Hou and Kumar 2012a). This is due to the memoryless property of the exponential (or geometric) distribution. We omit the proof here. Still in some scenarios where we know more about the task workloads it is interesting to explore other simple policies that perform better than LDF+Greedy, especially when δ is comparable to the maximum mean workload. That motivates the discussion in the next subsection.

4.3 Performance Analysis of LDF+TS/LLREF Scheduling under Deterministic Workloads

In this subsection, we consider systems where users generate tasks with deterministic, but possibly different, workloads, i.e., $\Pr(W_i = \mu_i) = 1$ for all $i \in N$. For soft real-time users that can tolerate missing some deadlines, even if they generate tasks with deterministic workloads, one can still intentionally drop a fraction of tasks in each period while guaranteeing the users' long-term QoS requirements. Selecting a subset of tasks to be processed in each period is like a bin backing problem. And to fulfill the long-term soft QoS requirements, one need to dynamically change or rotate the selected task subset. Recall that in each period $t+1$, we let $X_i(t)$ be the defict for user i and $Y_i(t+1)$ is the indicator variable that user i's task completes in this period. It is known that the max-weight algorithm which maximizes $\sum_{i \in N} X_i(t) Y_i(t+1)$ in each period $t+1$ is optimal. But this involves solving this fairly complex optimization problem which is indeed NP-complete in each period and this policy is hard to implement in practical systems. We want to explore simple policies that perform well.

Note deterministic workloads satisfy the NBUE property. Also note that for deterministic workloads, non-clairvoyant policies have knowledge of workload realizations. We shall once again prioritize users using LDF prioritization. Intuitively, the greedy task scheduler wastes time on multiple cores if multiple tasks are unfinished at the end of a period, so we will devise a task scheduler that orchestrates across cores so as to "reduce" wasted core time to finish more tasks.

For deterministic workloads, one can assess how many tasks one can complete prior to initiating processing. Indeed, it is intuitive, and established in (Cho et al. 2006), that one can complete all tasks in a user subset S in a period by some optimal scheduling if and only if $\sum_{i\in S} \mu_i \leq m\delta$. We consider one such optimal algorithm: Largest Local Remaining Execution time First (LLREF) (Cho et al. 2006). Let us briefly describe how LLREF[7] would work in the SRT-MIC model and then introduce a task scheduler that combines the idea of task selection and LLREF scheduling.

To that end we introduce some terminology used in (Cho et al. 2006). Consider a period starting at time $t\delta$ and ending at time $(t+1)\delta$, at any time $\tau \in [t\delta, (t+1)\delta]$, the *Local Remaining Execution time (LRE)* of user i is defined as the remaining time needed to complete its task. The LRE decrements as the task is processed. Further, the *laxity* of user i is defined as the remaining time before the deadline of user i's task, i.e., $(t+1)\delta - \tau$, minus the current LRE of user i. Thus, if some user has zero laxity at some time, one needs to start processing the task immediately to complete it by its deadline.

Definition 4.4. For the SRT-MIC model with deterministic workloads, the **Largest Local Remaining Execution time First (LLREF)** policy is such that, given a selected user subset S for the period, it does the following:

(1) At the beginning of the period, m tasks associated with users in S are chosen to be processed according to largest LRE first.
(2) When a running task completes, or a non-running task reaches a state where it has zero laxity, again the m tasks in S with largest local remaining execution time are selected to be processed.

Note that the LLREF policy uses task preemption and possibly migration. A review of variants of LLREF aimed at reducing task preemptions is provided in (Davis and Burns 2011b).

Definition 4.5. The **Task Selection/LLREF (TS/LLREF)** task scheduler is such that, given the user priority decision $\mathbf{d}$ for a period, it does the following:

(1) Task selection: it greedily selects users based on $\mathbf{d}$ until the sum workload exceeds $m\delta$. More formally, it selects

$$j(\mathbf{d}) = \max\left\{j | \sum_{i=1}^{j} \mu_{d_i} \leq m\delta\right\}. \tag{14}$$

Let $J(\mathbf{d}) = \{d_1, d_2, \cdots, d_{j(\mathbf{d})}\}$ represent the selected user subset.
(2) LLREF for $J(\mathbf{d})$: the system uses LLREF scheduling for tasks in $J(\mathbf{d})$ in this period.

By (Cho, Ravindran, and Jensen Cho et al.), it follows that all tasks from $J(\mathbf{d})$ will complete.

[7]LLREF is defined to be applicable in more general settings where users might generate tasks with different period. We will discuss this in Section 6.

Paralleling Theorem 4.3, we have the following result for the LDF+TS/LLREF resource allocation, i.e., the combination of LDF user prioritization and TS/LLREF task scheduling.

THEOREM 4.6. *For the SRT-MIC model with deterministic workloads, the efficiency ratio of LDF+TS/LLREF exceeds* γ_2 *where*

$$\gamma_2 = 1 - \frac{\max_{i \in N} \mu_i}{m\delta}.$$

Intuitively, under TS/LLREF, the task selection rule guarantees that in any given period the wasted time $m\delta - \sum_{i \in J(\mathbf{d})} \mu_i$ is less than $\max_{i \in N} \mu_i$. Given the total available core time $m\delta$, the gap between LDF+TS/LLREF and optimality is again bounded by the fraction of wasted time, i.e., $\frac{\max_{i \in N} \mu_i}{m\delta}$. A formal proof of this result is similar to that of Theorem 4.3 and is provided in Appendix 8.3.

The efficiency ratio lower bound γ_2 in this theorem is better than γ_1 obtained in Theorem 4.3, specifically the dependence on m is much stronger. For a system with a large number of cores m, γ_2 is close to 1, i.e., LDF+TS/LLREF is close to feasibility optimal even if δ is comparable to $\max_{i \in N} \mu_i$.

Although LDF+TS/LLREF is designed for deterministic workloads, we envisage it will work well for workloads with small variability by using the expected workload, or some more sophisticated workload estimation w_i^{est}. Specifically, TS makes selections based on w_i^{est} and LLREF computes local remaining execution time and laxity by assuming $W_i = w_i^{\text{est}}$. Note that this heuristic LDF+TS/LLREF is still non-clairvoyant. This will be explored in the simulation section.

4.4 Resource Requirements

So far we have analytically characterized the efficiency ratios of two LDF-based resource allocation policies. Another metric of interest is the resource requirements in terms of the number of cores m needed to fulfill a set of users' QoS requirements. To that end in this subsection we shall explore the required m given n, δ, the random workload distributions and the requirement vector $\mathbf{q}$. A policy that requires a smaller m is better in that it saves compute resources and/or energy.

4.4.1 *Resource Requirements for Reservation-Based Static Sharing.*

Based on the definition of F_{RB} in 3.1, the required number of cores to fulfill the users' QoS requirements $\mathbf{q}$ under reservation-based static sharing is given by

$$m_{\text{RB}} = \left\lceil \frac{\sum_{i \in N} w_i(q_i)}{\delta} \right\rceil, \tag{15}$$

where $\lceil x \rceil$ is the ceiling of x.

4.4.2 *Lower Bound on Resource Requirements.*

For any non-clairvoyant resource allocation policy η, we let m_η denote the required number of cores to fulfill users' QoS requirements under policy η. By Theorem 3.1, we know m_η must satisfy $m_\eta \delta \geq \sum_{i \in N} q_i \mu_i$, giving the following lower bound on the required number of cores:

$$\underline{m} \equiv \left\lceil \frac{\sum_{i \in N} q_i \mu_i}{\delta} \right\rceil. \tag{16}$$

4.4.3 Resource Requirements Estimate for LDF+Greedy.

Ideally one would like a tight upper bound for the required resources $m_{\text{LDF+Greedy}}$ for LDF+Greedy. By Theorem 4.3 we know that LDF+Greedy may expect to waste up to $\max_{i\in N} \mu_i$ time on each core in a period because of unfinished tasks. Thus, to complete an "effective" workload $\sum_{i\in N} q_i\mu_i$, we propose an estimate for $m_{\text{LDF+Greedy}}$ as follows,

$$m^{\text{est}}_{\text{LDF+Greedy}} \equiv \left\lceil \frac{\sum_{i\in N} q_i\mu_i}{\delta - \max_{i\in N} \mu_i} \right\rceil. \tag{17}$$

If $\delta \gg \max_{i\in N} \mu_i$, this estimate is close to the lower bound $\underline{m}$.

We can analytically show that indeed $m^{\text{est}}_{\text{LDF+Greedy}} \geq m_{\text{LDF+Greedy}}$ when δ and n are large, see the extended version of this paper (EXT 2017). We observe that the inequality holds true in the various simulation settings considered next.

5 SIMULATIONS

In this section we address through simulation some of the questions that are still open:

(1) What are possible resource savings of adopting LDF+Greedy versus reservation-based static sharing? Are they close to optimal when δ is large? How do they depend on the QoS requirements $\mathbf{q}$?
(2) Our theorems on the lower bounds on efficiency ratios imply that LDF+TS/LLREF is better than LDF+Greedy for small δ and deterministic workloads. Is it true that LDF+TS/LLREF is more efficient?
(3) For workloads with small variability, can one use LDF+TS/LLREF and get gains over LDF+Greedy?

Our simulation setup is as follows. We start with an initial deficit vector $\mathbf{X}(0) = (0, 0, \cdots, 0)$. In each period, we independently generate a task workload realization for each user and simulate the specified policy to evaluate if tasks complete. All simulations are run for 3000 periods. A QoS requirement vector $\mathbf{q}$ is feasible if for all users i the fraction of task completions over the 3000 periods exceeds q_i.

5.1 Near-Optimality of LDF+Greedy for Large δ

To evaluate the resource savings of LDF+Greedy for large period length δ, we consider an SRT-MIC system model with $n = 200$ and $\delta = 50$, serving homogeneous users that have the same QoS requirement q and generate tasks with Gamma$(5, 1)$ workloads, i.e., a sum of 5 independent exponential random variables with parameter 1. The probability density function is shown in the top panel in Figure 3. We choose this NBUE workload distribution as a representative one[8].

In the bottom panel in Figure 3, we show the simulated resource savings of LDF+Greedy versus the reservation-based static sharing, i.e., $1 - \frac{m_{\text{LDF+Greedy}}}{m_{\text{RB}}}$, and the computed upper bound on resource savings $1 - \frac{\underline{m}}{m_{\text{RB}}}$ as the QoS requirement q increases from 0 to 1. The lines are not smooth because we take ceilings when computing $\underline{m}$ and m_{RB}.

It can be seen that the savings under LDF+Greedy is close to the upper bound in this setting. The "U" shape of the exhibited results depends on the workload distribution.

[8]Gamma distributions with shape parameters $k \geq 1$ is an important class of NBUE distributions. We pick Gamma$(5, 1)$ as a representative NBUE distribution.

Intuitively, in this homogeneous-user scenario, if we ignore the ceilings in (15) (16), the upper bound on savings becomes,

$$1 - \frac{m}{m_{\text{RB}}} \simeq 1 - \frac{q\mu}{w(q)}, \tag{18}$$

where μ is the common mean workload and $w(q)$ is the common required static allocation. For high q, $w(q)$ is like a worst-case workload and this is an improvement from worst case to average which is as high as 60-70% for Gamma$(5, 1)$ distribution. For medium $q \sim 50\%$, $q\mu$ is around 0.5μ while $w(q)$ is roughly μ, giving a 50% resource savings. For low q, $q\mu$ is much smaller compared to $w(q)$ and the savings can be up to 80-90%.

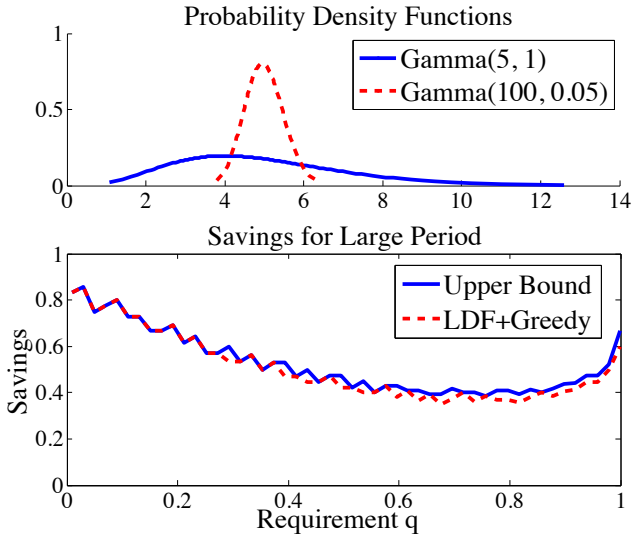

Fig. 3. Top: the probability density functions for Gamma$(5, 1)$ and Gamma$(100, 0.05)$. Bottom: the resource savings for large period.

5.2 LDF+Greedy vs. LDF+TS/LLREF for Deterministic Workloads and Small δ

To compare LDF+Greedy and LDF+TS/LLREF for short periods δ and deterministic workloads, we consider a system where $n = 30$ and $\delta = 9$ and where users are homogeneous and generate tasks with deterministic workloads $\mu = 5$. In the top panel in Figure 4, we exhibit the upper bound of resource savings and the resource savings under LDF+Greedy and LDF+TS/LLREF as the requirement q changes from 0 to 1.

As can be seen, LDF+TS/LLREF can achieve the upper bound on savings while LDF+Greedy does not perform as well. For high q, the savings for LDF+Greedy is even negative implying that LDF+Greedy is worse than the reservation-based approach. This is because we chose μ and δ such that LDF+Greedy wastes a significant amount of time on unfinished tasks. Observe that the savings are monotonically decreasing in q, which is different from the "U" shape exhibited in Figure 3. Intuitively, this is because for deterministic workloads, by (18) we know $w(q)$ equals to μ and thus we get

$$1 - \frac{m}{m_{\text{RB}}} \simeq 1 - q.$$

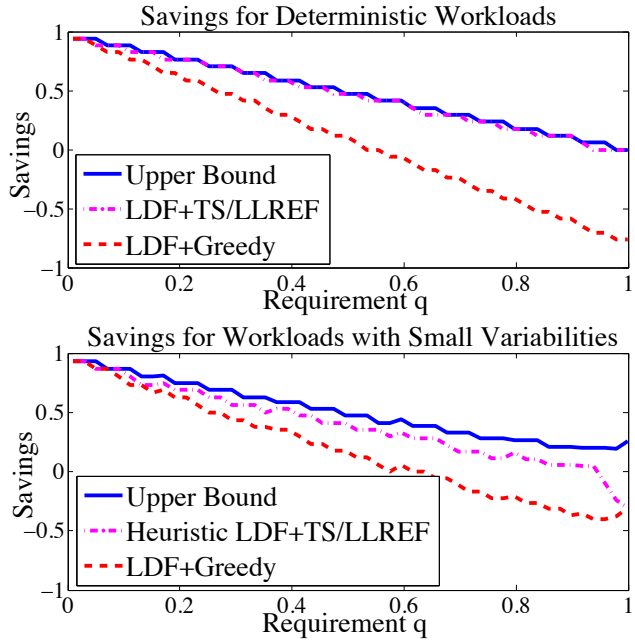

Fig. 4. Top: the resource savings under deterministic workloads. Bottom: the resource savings under random workloads with small variability.

5.3 LDF+TS/LLREF for Workloads with Small Variability

For workloads with small variability, we envisage that the heuristic LDF+TS/LLREF described in Section 4.3 is a good non-clairvoyant policy. Consider a SRT-MIC system with homogeneous users where $n = 30$ and $\delta = 9$ and where the task workload distributions are Gamma$(100, 0.05)$ exhibited on the top panel in Figure 3. Note that the distribution Gamma$(100, 0.05)$ has the same mean $\mu = 5$ but a small variance. In this setting, we shall estimate the workload to be $w^{\text{est}} = 1.1\mu$ and use our proposed heuristic LDF+TS/LLREF in Section 4.3. We conduct the same analysis for resource savings and exhibit the results in the bottom panel in Figure 4.

As can be seen, the heuristic LDF+TS/LLREF indeed performs better than LDF+Greedy. However, the performance of the heuristic LDF+TS/LLREF degrades for high q. This is due to the fact that some selected tasks fail to complete since their workloads are larger than w^{est}. One approach to solve this is to increase w^{est} as q becomes bigger.

Although we only considered homogeneous users, the above observations were found to be robust for heterogeneous users.

6 POSSIBLE GENERALIZATIONS

In this section we discuss the following generalizations of the SRT-MIC NBUE-workload model and associated results:

(1) Cores with different processing speeds.
(2) Users generating tasks at different periods.
(3) Tasks which further consist of sub-tasks that need to be processed in order.

Table 1. Results for different generalizations.

Model	Reservation-Based F_{RB}	Outer Bound R_{OB}	γ_1 (NBUE workloads)		γ_2 (deterministic workloads)
			preemptive	non-preemptive	
SRT-MIC	$\{\mathbf{q} \in \mathbb{R}_+^n \mid \mathbf{q} \preceq \mathbf{1}, \sum_{i \in N} w_i(q_i) \le m\delta, w_i(q_i) \le \delta, \forall i \in N\}$	$\{\mathbf{q} \in \mathbb{R}_+^n \mid \mathbf{q} \preceq \mathbf{1}, \sum_{i \in N} q_i \mu_i \le m\delta\}$	$1 - \frac{\max_{i \in N} \mu_i}{\delta}$		$1 - \frac{\max_{i \in N} \mu_i}{m\delta}$
Different speeds s_c	$\{\mathbf{q} \in \mathbb{R}_+^n \mid \mathbf{q} \preceq \mathbf{1}, B_n(\mathbf{q}) \le S_m \cdot \delta, B_k(\mathbf{q}) \le S_k \cdot \delta, 1 \le k \le m\}$	$\{\mathbf{q} \in \mathbb{R}_+^n \mid \mathbf{q} \preceq \mathbf{1}, \sum_{i \in N} q_i \mu_i \le S_m \cdot \delta\}$	$1 - \frac{\max_{i \in N} \mu_i}{\overline{s} \cdot \delta}$	$1 - \frac{\max_{i \in N} \mu_i}{\min_{c \in C} s_c \cdot \delta}$	$1 - \frac{\max_{i \in N} \mu_i}{S_m \cdot \delta}$
Different periods δ_i	$\{\mathbf{q} \in \mathbb{R}_+^n \mid \mathbf{q} \preceq \mathbf{1}, \sum_{i \in N} \frac{w_i(q_i)}{\delta_i} \le m, w_i(q_i) \le \delta_i, \forall i \in N\}$	$\{\mathbf{q} \in \mathbb{R}_+^n \mid \mathbf{q} \preceq \mathbf{1}, \sum_{i \in N} \frac{q_i \mu_i}{\delta_i} \le m\}$	N/A		$1 - \frac{\max_{i \in N} \frac{\mu_i}{\delta_i}}{m}$
Chains of subtasks $k(i)$	$\{\mathbf{q} \in \mathbb{R}_+^n \mid \mathbf{q} \preceq \mathbf{1}, \sum_{i \in N} w_i(q_i) \le m\delta, w_i(q_i) \le \delta, \forall i \in N\}$	$\{\mathbf{q} \in \mathbb{R}_+^n \mid \mathbf{q} \preceq \mathbf{1}, \sum_{i \in N} q_i \mu_i \le m\delta\}$	$1 - \frac{\max_{i \in N} \mu_i}{\delta}$		$1 - \frac{\max_{i \in N} \mu_i}{m\delta}$

We discuss these three generalizations in the following three subsections, respectively.

For ease of reference, Table 1 provides a summary of various generalizations—the necessary notation is introduced in the sequel.

6.1 Cores with Different Processing Speeds

We first consider generalizations where the cores may have different processing speeds. Let $C = \{1, 2, \cdots, m\}$ denote the set of cores. Suppose all cores are of the same type and each core $c \in C$ has processing speed s_c, i.e., cores are "uniform", see the taxonomy in, e.g., (Davis and Burns 2011a). In other words, if a task runs on a core with speed s for t time units, then $s \times t$ units of work are performed. In this context, the workload of a task refers to the required units of *work* to fully complete the task. Therefore, a task with workload w processed on core c has a processing time $\frac{w}{s_c}$. Let $\overline{s} = \frac{\sum_{c \in C} s_c}{m}$ be the average processing speed. Clearly, in the SRT-MIC model we have previously considered, $s_c = 1$ for each $c \in C$.

We assume $n \ge m$ since otherwise one only needs the n fastest cores. Next we discuss generalizations of our results.

6.1.1 Reservation-Based Static Sharing Policies. In reservation-based static sharing, given the computed $w_i(q_i)$ for all users $i \in N$, the question is whether it is feasible to find a static allocation guaranteeing that $w_i(q_i)$ units of work can be performed for each user i in each period.

To answer this question, we first introduce some notation. Given a set Z of non-negative numbers and a positive integer k which satisfies $1 \le k \le |Z|$, we let $a(Z, k)$ be the sum of the largest k numbers in Z. We let $S_k = a(\{s_c | c \in C\}, k)$. Given a QoS requirement vector $\mathbf{q}$, for $1 \le k \le n$, we let $B_k(\mathbf{q}) = a(\{w_i(q_i) | i \in N\}, k)$ be the sum of the k largest core time reservations. By (Funk et al. 2001; Funk and Meka 2009), we know that a static allocation

is feasible if and only if the following conditions hold:

$$B_n(\mathbf{q}) \leq S_m \cdot \delta, \tag{19}$$

$$B_k(\mathbf{q}) \leq S_k \cdot \delta \text{ for } 1 \leq k \leq m. \tag{20}$$

Intuitively, (19) implies that the sum of required reservations does not exceed the total units of work that can be performed in a period. And (20) implies that the k largest reservation requirements can be satisfied by the k fastest cores.

Such a static allocation can be obtained according to prior work, see e.g., (Funk et al. 2001; Funk and Meka 2009). Therefore, the feasibility region of reservation-based static sharing F_{RB} is given by

$$F_{\text{RB}} = \{\mathbf{q} \in \mathbb{R}_+^n \mid \mathbf{q} \preceq \mathbf{1}, B_n(\mathbf{q}) \leq S_m \cdot \delta, \\ B_k(\mathbf{q}) \leq S_k \cdot \delta \text{ for } 1 \leq k \leq m\}.$$

This is consistent with our analysis when $s_c = 1$ for all $c \in C$, see Eq (3).

6.1.2 Outer Bound R_{OB} for the System Feasibility Region. For a system with different core processing speeds, the outer bound R_{OB} in Theorem 3.1 needs to be modified to

$$R_{\text{OB}} \equiv \{\mathbf{q} \in \mathbb{R}_+^n \mid \mathbf{q} \preceq \mathbf{1}, \sum_{i \in N} q_i \mu_i \leq S_m \cdot \delta\},$$

i.e., the "effective" workload $\sum_{i \in N} q_i \mu_i$ cannot exceed the maximum units of work $S_m \cdot \delta$ that can be performed in a period.

A proof of this result requires a slight modification of that of Theorem 3.1: we replace $m\delta$ by $S_m \cdot \delta$; we redefine U_S to be the total units of work performed for users in S in a typical period; and we redefine $A_{i,c}$ to be the indicator random variable that user i's task is unfinished and c units of work are performed for user i's task in a typical period.

6.1.3 LDF+Greedy Scheduling. For LDF+Greedy, if all cores have the same speed, there is no benefit of moving a running task from one core to another. However, if cores have different speeds, one may want to migrate tasks to faster cores if they become available. Therefore, depending on whether task preemption/migration is allowed, there are two types of greedy task schedulers: preemptive and non-preemptive greedy task scheduler.

Preemptive Greedy Task Scheduler: In the preemptive case, the task scheduler greedily and preemptively schedules tasks with the highest priority on the fastest cores. Specifically, at all times the task scheduler guarantees that the available[9] task with the highest priority is placed on the fastest core, the available task with the second highest priority is on the second fastest core, etc. In this setting, similarly to Theorem 4.3 we get the following corollary.

Corollary 6.1. *For the generalization of SRT-MIC model to cores with different processing speeds, the efficiency ratio of the preemptive LDF+Greedy exceeds γ_1 where*

$$\gamma_1 = 1 - \frac{\max_{i \in N} \mu_i}{\overline{s} \cdot \delta}.$$

Note that in the denominator we have an average processing speed $\overline{s}$, which equals to 1 in the SRT-MIC model we considered previously. Intuitively, this is because under the preemptive greedy task scheduler the unfinished tasks are always on the fastest cores. And

[9] A task is available if it is not completed yet.

the average processing speed of the k fastest cores is at least $\overline{s}$ for $1 \leq k \leq m$. We omit the proof to save space. For a detailed proof, see the extended version of this paper (EXT 2017).

Non-Preemptive Greedy Task Scheduler: The non-preemptive greedy task scheduler starts by putting the task with the highest priority on the fastest core, the task with the second highest priority on the second fastest core, etc. Once one of these tasks completes, it continues by processing the task with priority $m+1$ on the available core, etc. In this setting, we get the following corollary.

COROLLARY 6.2. *For the generalization of SRT-MIC model with different processing speeds, the efficiency ratio of the non-preemptive LDF+Greedy exceeds* γ_1 *where*

$$\gamma_1 = 1 - \frac{\max\limits_{i \in N} \mu_i}{\min\limits_{c \in C} s_c \cdot \delta}.$$

See the extended version of this paper (EXT 2017) for the proof.

Note that γ_1 under the preemptive LDF+Greedy is larger than that under the non-preemptive LDF+Greedy. This captures the benefit of task preemption/migration although these operations involve overheads in practice.

6.1.4 LDF+TS/LLREF Scheduling. For deterministic workloads, we shall generalize our proposed LDF+TS/LLREF scheduling. We first introduce a further assumption.

Assumption 1. *We suppose the n users' deterministic workloads are such that for all $1 \leq k \leq m$,*

$$M_k \leq S_k \cdot \delta,$$

where $M_k = a(\{\mu_i | i \in N\}, k)$ represents the sum of the k largest workloads.

Intuitively, this guarantees that for all $1 \leq k \leq m$, the k tasks with largest workloads can complete on the k fastest processors in a period.

Under Assumption 1, and by (Funk et al. 2001; Funk and Meka 2009), we can complete all tasks in a user subset S in a period by some optimal scheduling if and only if $\sum\limits_{i \in S} \mu_i \leq S_m \cdot \delta$. Such optimal scheduling algorithms include U-LLREF (Funk and Meka 2009), a variant of LLREF for cores with different speeds, and Proportionate Fair (Pfair) (Baruah et al. 1996).

Similar to the TS/LLREF task scheduler in Definition 4.5, we propose TS/U-LLREF or TS/Pfair where the task selection rule (14) naturally becomes

$$j(\mathbf{d}) = \max\left\{ j | \sum_{i=1}^{j} \mu_{d_i} \leq S_m \cdot \delta \right\}, \tag{21}$$

and the selected subset of users are scheduled via U-LLREF or Pfair algorithms.

Under Assumption 1, and similarly to Theorem 4.6, we can show that the efficiency ratio of LDF+TS/U-LLREF or LDF+TS/Pfair exceeds γ_2 where

$$\gamma_2 = 1 - \frac{\max\limits_{i \in N} \mu_i}{S_m \cdot \delta}.$$

The proof of this result follows that of Theorem 4.6 by simply replacing $m\delta$ with $S_m \cdot \delta$.

6.2 Users Generating Tasks at Different Periods

In this subsection, we consider possible generalizations of the SRT-MIC NBUE-workload model where users generate tasks at different periods, and discuss results that cannot be generalized and/or associated difficulties.

Specifically, suppose starting from time 0 each user i generates a task at the beginning of each period of length δ_i. We assume there exists a minimum common multiple Δ of δ_i for all i. We shall refer to Δ as a *super period*.

Again, each user requires the long-term time-averaged number of tasks completed on time per period $q_i \in [0, 1]$. To be consistent with the SRT-MIC model, we define the feasibility in terms of the positive recurrence of a Markov chain. Given $\mathbf{q} = (q_1, q_2, \cdots, q_n)$, we keep track of the deficits of users across super periods. For each user $i \in N$ and super period $t + 1$, we shall define deficit updates as follows,

$$X_i(t+1) = [X_i(t) + q_i \cdot \frac{\Delta}{\delta_i} - Y_i(t+1)]^+, \tag{22}$$

where $Y_i(t + 1)$ is a random variable representing the number of tasks completed on time for user i in super period $t + 1$. Let $\mathbf{X}(t) = (X_1(t), X_2(t), \cdots, X_n(t))$. We only consider non-clairvoyant resource allocation policies such that the process $\{\mathbf{X}(t)\}_{t\geq 1}$ is a Markov chain. A QoS requirement vector $\mathbf{q}$ is feasible if the Markov chain $\{\mathbf{X}(t)\}_{t\geq 1}$ is positive recurrent under some non-clairvoyant resource allocation policy.

6.2.1 Reservation-Based Static Sharing Policies. We first generalize the performance characterization of reservation-based static sharing policies. Similarly to the setting in 3.1, we can compute the required core time reservation per period $w_i(q_i)$ for all users i. Now $\frac{w_i(q_i)}{\delta_i}$ represents the required core utilization for user i if we want to allocate $w_i(q_i)$ core time to user i per period. Clearly, if $\sum_{i\in N} \frac{w_i(q_i)}{\delta_i} > m$ we cannot meet the core time reservations $w_i(q_i)$ for all users. Indeed, by prior work, see e.g., (Cho et al. 2006; Davis and Burns 2011a), we can characterize the feasibility region F_{RB} of reservation-based static sharing policies as follows,

$$F_{\text{RB}} = \{\mathbf{q} \in \mathbb{R}^n_+ \mid \mathbf{q} \preceq \mathbf{1}, \sum_{i\in N} \frac{w_i(q_i)}{\delta_i} \leq m, \\ w_i(q_i) \leq \delta_i, \forall i \in N\}.$$

Note that this is consistent with our analysis when all users have the same period, see Eq (3).

Given that $\sum_{i\in N} \frac{w_i(q_i)}{\delta_i} \leq m$ and $w_i(q_i) \leq \delta_i$ for all $i \in N$, since users have different periods, the remaining problem is how to allocate $w_i(q_i)$ to each user i in each period. One solution is to use the LLREF scheduling policy. We omit the details to save space. Refer to the extended version of this paper (EXT 2017) for detailed discussion.

6.2.2 Outer Bound R_{OB} for the System Feasibility Region. When users generate tasks with different periods, the outer bound R_{OB} for the system feasibility region can be generalized as follows,

$$R_{\text{OB}} = \{\mathbf{q} \in \mathbb{R}^n_+ \mid \mathbf{q} \preceq \mathbf{1}, \sum_{i\in N} \frac{q_i\mu_i}{\delta_i} \leq m\}.$$

Intuitively, $\sum_{i \in N} \frac{q_i \mu_i}{\delta_i}$ represents the sum of core utilizations to fulfill QoS requirement $\mathbf{q}$, which cannot exceed the maximum degree of parallelism m. The proof is similar to that of Theorem 3.1—refer to the extended version of this paper (EXT 2017) for details.

6.2.3 LDF-Based Policies Over Super Periods. A heuristic way to generalize our proposed LDF-based resource allocation policies to different-period scenarios is to adopt the LDF policy to pick a priority decision for each super period. Specifically, at the beginning of super period $t+1$, the system orders the deficit vector $\mathbf{X}(t)$ and assigns priorities from largest to smallest. These priorities are interpreted by the task scheduler to schedule tasks in this super period.

LDF+Greedy: When users generate tasks with different periods, the greedy task scheduler can be preemptive or non-preemptive depending on whether preemption/migration is allowed. In the preemptive version, at all times the task scheduler processes the m available tasks with the highest priority on the m cores. In the non-preemptive version, the task scheduler starts with m tasks with the highest priority. When a running task completes or reaches its deadline[10], the available non-running task with the highest priority is selected to be processed on the available core.

Unfortunately, for this generalized LDF+Greedy policy we cannot get a similar performance characterization as Theorem 4.3. Intuitively, this is because the greedy task scheduler can potentially waste a lot of time on unfinished tasks in different-period scenarios. For example, under the preemptive greedy task scheduler, we may start processing a task right before its deadline and fail to complete it, or we may process a task only for a short time before we have to switch to process another task with higher priority leaving the original task unfinished. These scenarios degrade the performance of the LDF+Greedy policy.

LDF+TS/LLREF under Deterministic Workloads: If the users generate tasks with different periods but with deterministic workloads, we can generalize the LDF+TS/LLREF policy and also Theorem 4.6. Naturally we assume $\mu_i \leq \delta_i$ for all $i \in N$. Otherwise, the tasks from user i cannot complete on time.

Under LDF+TS/LLREF, in each super period, a priority decision $\mathbf{d}$ is selected according to the LDF policy. Similarly to (14), the system selects the user subset $J(\mathbf{d}) = \{d_1, d_2, \cdots, d_{j(\mathbf{d})}\}$ where $j(\mathbf{d})$ is computed as follows,

$$j(\mathbf{d}) = \max\left\{j | \sum_{i=1}^{j} \frac{\mu_{d_i}}{\delta_{d_i}} \leq m\right\}. \tag{23}$$

We shall consider the case where the system adopts the LLREF policy to process and complete all tasks from $J(\mathbf{d})$ in this super period.

To characterize the efficiency ratio, we proved the following corollary which is similar to Theorem 4.6.

Corollary 6.3. *For the SRT-MIC system model with different periods and deterministic workloads, the efficiency ratio of LDF+TS/LLREF that operates over super periods exceeds* γ_2*, where*

$$\gamma_2 = 1 - \frac{\max_{i \in N} \frac{\mu_i}{\delta_i}}{m}.$$

[10]This implies that another task from the same user is released. That new task is also considered to be a non-running task.

Intuitively, under the task selection rule (23), for the selected user subset $J(\mathbf{d})$ we know that $m - \sum_{i \in J(\mathbf{d})} \frac{\mu_i}{\delta_i}$ is less than $\max_{i \in N} \frac{\mu_i}{\delta_i}$, and therefore, the performance gap is bounded by $\frac{\max_{i \in N} \frac{\mu_i}{\delta_i}}{m}$. The formal proof is straightforward generalization of the proof of Theorem 4.6 and we shall omit it.

Again, this result is consistent with our analysis when all users have the same period, see Theorem 4.6.

6.2.4 Fine-Grained LDF-Based System Designs. A problem for the LDF-based resource allocation policies over super periods is that the task completions of users vary a lot from super period to super period. For example, a user with high priority in one super period may complete a large number of tasks in this super period and then be assigned a low priority in the next super period, completing only a small number of tasks. Such bursty completions would likely be undesirable for users especially when the super period Δ is large.

To mitigate this problem, we could consider a fine-grained LDF policy to change the priority decisions more frequently. We divide the timeline into intervals associated with times where tasks become available for processing and deadlines. At the beginning of each interval, we compute the deficit between the QoS requirement and the actual number of completed tasks up to that time for each user i, sort the deficits from largest to smallest and assign priorities accordingly.

Given the priority decision in each interval, we can adopt a greedy task scheduler. If task preemption/migration is allowed, naturally we start by putting the m tasks with highest priority on the m cores, and once one of these tasks completes, we continue by putting the task with priority $m+1$ on the available core, etc. If preemption/migration is not allowed, at the beginning of this interval, we continue processing the tasks running at the end of the previous interval, and once one of these tasks completes or reaches the deadline, we put the non-running task with the highest priority on the available core, etc.

It would be of interest to characterize the performance of such resource allocation policies and to generalize LDF+TS/LLREF in future work.

6.3 Tasks Consisting of Sub-Tasks

We continue our discussion of possible generalizations of our SRT-MIC NBUE-workload model to the case where each task consists of several sub-tasks that need to be processed in order and all of which need to be completed by the end of the corresponding period. We assume all sub-tasks can be processed on all cores.

Specifically, suppose in each period each user $i \in N$ generates a task consisting of $k(i)$ sub-tasks, which have to be processed in order and cannot be processed in parallel. But sub-tasks of different tasks can be processed simultaneously. A task in a period is said to be completed on time if and only if all its sub-tasks complete by the end of the period. Each user i requires time-averaged task completions per period q_i. For a given user, we assume the sub-task workloads with the same sub-task index are i.i.d. across periods and the sub-task workloads with different indices are independent. For each user $i \in N$ and each sub-task index $k = 1, 2, \cdots, k(i)$, we denote by $W_i^{(k)}$ the workload of the k^{th} sub-task from user i and let $\mu_i^{(k)} = \mathrm{E}[W_i^{(k)}]$ be the mean sub-task workload. Clearly $W_i = \sum_{k=1}^{k(i)} W_i^{(k)}$ and $\mu_i = \sum_{k=1}^{k(i)} \mu_i^{(k)}$. We further assume each sub-task has an NBUE workload distribution.

Clearly our original SRT-MIC system model is a special case of this generalized model where $k(i) = 1$ for all users i. It turns out that our proposed approaches and performance characterization still hold under this generalized task model although some of the proofs need modification. We omit the details to save space. Refer to the extended version of this paper (EXT 2017) for detailed discussion.

In this section we have introduced three possible generalizations in parallel. Given these results, the combinations of multiple generalizations, e.g., scenarios where the processors have different processing speeds and users generate tasks with different periods, are straightforward and we omit the discussion here.

7 CONCLUSION

We have considered a computing system with multiple resources supporting soft real-time applications and established analytically and through simulation that simple resource allocation policies like LDF+Greedy are near-optimal and achieve substantial resource savings, except when the real-time constraints are tight, i.e., the period length is similar to the service time for a user's task. In this case, LDF+Greedy may not work well and it is worth exploring other policies. For workloads with small variability, we have proposed the LDF+TS/LLREF policy which indeed outperforms LDF+Greedy. For future work, a more detailed exploration of systems consisting of possibly different types of resources is of interest.

REFERENCES

2017. Extended version. http://arxiv.org/abs/1601.06333. (2017).

Fardin Ahmadizar, Mehdi Ghazanfari, and Seyyed Mohammad Taghi Fatemi Ghomi. 2010. Group shops scheduling with makespan criterion subject to random release dates and processing times. *Computers and Operations Research* 37 (2010), 152–162. Issue 1.

Ali Allahverdi and Yuri Sotskov. 2003. Two-machine flowshop minimum-length scheduling problem with random and bounded processing times. *International Transactions in Operational Research* 10 (2003), 65–76. Issue 1.

Yair Amir, Baruch Awerbuch, Amnon Barak, R. Sean Borgstrom, and Arie Keren. 2000. An Opportunity Cost Approach for Job Assignment in a Scalable Computing Cluster. *IEEE Transactions on Parallel and Distributed Systems* 11 (July 2000), 760–768. Issue 7.

Alia Atlas and Azer Bestavros. 1998. Statistical Rate Monotonic Scheduling. In *Proceedings of RTSS 1998*. 123–132.

S. K. Baruah, N. K. Cohen, C. G. Plaxton, and D. A. Varvel. 1996. Proportionate Progress: A Notion of Fairness in Resource Allocation. *Algorithmica* 15 (June 1996), 600–625. Issue 6.

Carlos J. Bernardos and others. 2014. An architecture for software defined wireless networking. *IEEE Wireless Communications* 21, 3 (2014).

Guillem Bernat and Alan Burns. 1997. Combining (^n_m)-Hard deadlines and Dual Priority Scheduling. In *Proceedings of RTSS 1997*. 46–57.

J. Blazewicz, M. Drabowski, and J. Weglarz. 1986. Scheduling Multiprocessor Tasks to Minimize Schedule Length. *IEEE Trans. Comput.* C-35 (1986), 389–393. Issue 5.

J. Bruno, E. G. Coffman Jr., and R. Sethi. 1974. Scheduling Independent Tasks To Reduce Mean Finishing Time. *Commun. ACM* 17 (1974), 382–387. Issue 7.

John Carpenter, Shelby Funk, Philip Holman, Anand Srinivasan, James Anderson, and Sanjoy Baruah. 2004. A Categorization of Real-time Multiprocessor Scheduling Problems and Algorithms. *Handbook of Scheduling: Algorithms, Models, and Performance Analysis* (2004).

China Mobile. 2011. C-RAN The Road Towards Green RAN. (Oct 2011).

Hyeonjoong Cho, Binoy Ravindran, and E. Douglas Jensen. An Optimal Real-Time Scheduling Algorithm for Multiprocessors. In *Proc. of RTSS 2006*.

Hyeonjoong Cho, Binoy Ravindran, and E. Douglas Jensen. 2006. An Optimal Real-Time Scheduling Algorithm for Multiprocessors. In *Proceedings of RTSS 2006*. 101–110.

J.H. Conway and N.J.A. Sloane. 2013. *Sphere Packings, Lattices and Groups*. Springer.

Robert I. Davis and Alan Burns. 2011a. A Survey of Hard Real-Time Scheduling for Multiprocessor Systems. *Comput. Surveys* 43 (October 2011). Issue 4.

Robert I. Davis and Alan Burns. 2011b. A Survey of Hard Real-Time Scheduling for Multiprocessor Systems. *Comput. Surveys* 43 (2011). Issue 4.

Christina Delimitrou and Christos Kozyrakis. 2014. Quasar: Resource-Efficient and QoS-Aware Cluster Management. In *Proceedings of the 19th international conference on Architectural support for programming languages and operating systems*. 127–144.

Antonis Dimakis and Jean Walrand. 2006. Sufficient Conditions for Stability of Longest-Queue-First Scheduling: Second-Order Properties Using Fluid Limits. *Advances in Applied Probability* 38, 2 (June 2006).

Yuhuan Du and Gustavo de Veciana. 2014. Wireless Networks Without Edge: Dynamic Radio Resource Clustering and User Scheduling. In *Proceedings of INFOCOM 2014*. 1321–1329.

Yuhuan Du and Gustavo de Veciana. 2016. Efficiency and Optimality of Largest Deficit First Prioritization: Resource Allocation for Real-Time Applications. *INFOCOM 2016* (April 2016).

Shelby Funk, Joël Goossens, and Sanjoy Baruah. 2001. On-line Scheduling on Uniform Multiprocessors. In *Proceedings of RTSS 2001*. 183–192.

Shelby Funk and Archana Meka. 2009. U-LLREF: An Optimal Scheduling Algorithm for Uniform Multiprocessors. In *Workshop on Models and Algorithms for Planning and Scheduling Problems*.

Alan Gatherer. 2015. Personal communication. (February 2015).

Moncef Hamdaoui and Parameswaran Ramanathan. 1995. A Dynamic Priority Assignment Technique for Streams with (m, k)-Firm Deadlines. *IEEE Trans. Comput.* 44 (December 1995), 1443–1451. Issue 12.

I-Hong Hou and P. R. Kumar. 2012a. Queueing systems with hard delay constraints: a framework for real-time communication over unreliable wireless channels. *Queueing Systems* 71 (March 2012), 151–177. Issue 1-2.

I-Hong Hou and P. R. Kumar. 2012b. Queueing systems with hard delay constraints: a framework for real-time communication over unreliable wireless channels. *Queueing Systems* 71 (2012). Issue 1-2.

I-Hong Hou and P. R. Kumar. 2013. *Packets with Deadlines: A Framework for Real-Time Wireless Networks*. Morgan & Claypool Publishers.

Juan Jose Jaramillo and R. Srikant. 2011. Optimal Scheduling for Fair Resource Allocation in Ad Hoc Networks With Elastic and Inelastic Traffic. *IEEE Transactions on Networking* 19 (August 2011), 1125–1136. Issue 4.

Changhee Joo, Xiaojun Lin, and Ness B. Shroff. 2007a. Performance Limits of Greedy Maximal Matching in Multi-hop Wireless Networks. In *IEEE Conference on Decision and Control*. 1128–1133.

Changhee Joo, Xiaojun Lin, and Ness B. Shroff. 2007b. Performance Limits of Greedy Maximal Matching in Multi-hop Wireless Networks. In *IEEE Conference on Decision and Control*.

Xiaohan Kang, Weina Wang, Juan Jose Jaramillo, and Lei Ying. 2013. On the Performance of Largest-Deficit-First for Scheduling Real-Time Traffic in Wireless Networks. In *Proceedings of MobiHoc*. 99–108.

Eugene L. Lawler, Jan karel Lenstra, Alexander H.G. Rinnooy Kan, and David B. Shmoys. 1993. Sequencing and Scheduling: Algorithms and Complexity. *Logistics of Production and Inventory* (1993), 445–522.

Joseph Y.-T. Leung. 1989. A New Algorithm for Scheduling Periodic, Real-Time Tasks. *Algorithmica* 4 (June 1989), 209–219. Issue 1.

Cong Liu and James H. Anderson. 2009. Task Scheduling with Self-Suspensions in Soft Real-Time Multiprocessor Systems. In *Proceedings of RTSS 2009*. 425–436.

C. L. Liu and James W. Layland. 1973. Scheduling Algorithms for Multiprogramming in a Hard Real-Time Environment. *J. ACM* 20, 1 (January 1973), 46–61.

Jane W.S. Liu, Kwei-Jay Lin, and Swaminathan Natarajan. 1987. Scheduling Real-time, Periodic Jobs Using Imprecise Results. In *Proceedings of RTSS 1987*. 252–260.

Jane W. S. Liu. 2000. *Real-Time Systems*. Prentice Hall.

Jason Mars, Lingjia Tang, Robert Hundt, Kevin Skadron, and Mary Lou Soffa. 2011. Bubble-Up: Increasing Utilization in Modern Warehouse Scale Computers via Sensible Co-locations. In *Proceedings of the 44th Annual IEEE/ACM International Symposium on Microarchitecture*.

A. Müller and D. Stoyan. 2002. *Comparison Methods for Stochastic Methods and Risks*. Wiley.

Shailesh Patil and Gustavo de Veciana. 2007. Managing Resources and Quality of Service in Heterogeneous Wireless Systems Exploiting Opportunism. *IEEE/ACM Transactions on Networking* 15 (October 2007), 1046–1058. Issue 5.

Michael L. Pinedo. 2012. *Scheduling: Theory, Algorithms, and Systems*. Springer.
Parameswaran Ramanathan. 1999. Overload management in real-time control applications using (m, k)-firm guarantee . *IEEE Transactions on Parallel and Distributed Systems* 10 (June 1999), 549–559. Issue 6.
Sanjay Shakkottai and Alexander L. Stolyar. 2001. Scheduling algorithms for a mixture of real-time and non-real-time data in HDR. In *Proceedings of the International Teletraffic Congress*. 793–804.
Abhishek Verma, Luis Pedrosa, Madhukar Korupolu, David Oppenheimer, Eric Tune, and John Wilkes. 2015. Large-scale cluster management at Google with Borg. In *Proceedings of EuroSys 2015*.

8 APPENDIX

8.1 Proof of Theorem 4.2

We first introduce some additional notation. Given two vectors $\boldsymbol{a} = (a_1, a_2, \cdots, a_n)$ and $\mathbf{b} = (b_1, b_2, \cdots, b_n)$, we denote by $\boldsymbol{a} \circ \mathbf{b} = (a_1 b_1, a_2 b_2, \cdots, a_n b_n)$ the entrywise product.

Given $\mathbf{q} \in \text{int}(R_{\text{IB}})$, we need only show $\mathbf{q}$ can be fulfilled by the LDF+$\mathcal{X}$ policy.

By definition of interior there exists an $\epsilon > 0$ such that $\mathbf{q}' = \mathbf{q} + \epsilon \mathbf{1} \in R_{\text{IB}}$. By definition of R_{IB}, there exists a vector $\boldsymbol{\alpha} \succ \mathbf{0}$ such that for all $S \subseteq N$,

$$\sum_{i \in S} \alpha_i q_i' \leq \min_{\mathbf{d} \in D(S)} \sum_{i \in S} \alpha_i p_i(\mathbf{d}). \tag{24}$$

Consider the following candidate Lyapunov function:

$$L(\mathbf{X}(t)) = \sum_{i=1}^{n} \alpha_i X_i(t)^2.$$

Note that the process $\{\mathbf{X}(t)\}_{t \geq 1}$ is now driven by LDF, and let $\mathbf{Y}(t) = (Y_1(t), Y_2(t), \cdots, Y_n(t))$ be the vector of indicator variables for users' task completions under LDF. At period $t+1$, we have that

$$\begin{aligned}
&\mathrm{E}\left[L(\mathbf{X}(t+1)) - L(\mathbf{X}(t)) | \mathbf{X}(t) = \mathbf{x}\right] \\
&= \mathrm{E}\left[\sum_{i=1}^{n} \alpha_i (X_i(t+1)^2 - X_i(t)^2) | \mathbf{X}(t) = \mathbf{x}\right] \\
&\leq \mathrm{E}\left[\sum_{i=1}^{n} \alpha_i ((X_i(t) + q_i - Y_i(t+1))^2 - X_i(t)^2) | \mathbf{X}(t) = \mathbf{x}\right] \\
&= \mathrm{E}\left[\sum_{i=1}^{n} \alpha_i (q_i - Y_i(t+1))^2 + 2\langle \boldsymbol{\alpha} \circ \mathbf{X}(t), \mathbf{q} - \mathbf{Y}(t+1)\rangle | \mathbf{X}(t) = \mathbf{x}\right] \\
&\leq \mathrm{E}\left[\sum_{i=1}^{n} \alpha_i (q_i^2 + Y_i(t+1)^2) + 2\langle \boldsymbol{\alpha} \circ \mathbf{X}(t), \mathbf{q} - \mathbf{Y}(t+1)\rangle | \mathbf{X}(t) = \mathbf{x}\right] \\
&= \mathrm{E}\left[\sum_{i=1}^{n} \alpha_i (q_i^2 + Y_i(t+1)^2) + 2\langle \boldsymbol{\alpha} \circ \mathbf{X}(t), \mathbf{q}' - \mathbf{Y}(t+1)\rangle | \mathbf{X}(t) = \mathbf{x}\right] - 2\epsilon \langle \mathbf{x}, \boldsymbol{\alpha}\rangle \quad (25)
\end{aligned}$$

For simplicity, let $\mathbf{d}$ denote the priority decision selected by LDF at period $t+1$. We have

$$\mathrm{E}\left[\langle \boldsymbol{\alpha} \circ \mathbf{X}(t), \mathbf{q}' - \mathbf{Y}(t+1)\rangle | \mathbf{X}(t) = \mathbf{x}\right] = \langle \boldsymbol{\alpha} \circ \mathbf{x}, \mathbf{q}' - \mathbf{p}(\mathbf{d})\rangle.$$

By reordering users according to priorities, we get

$$\begin{aligned}
&\langle \boldsymbol{\alpha} \circ \mathbf{x}, \mathbf{q}' - \mathbf{p}(\mathbf{d}) \rangle \\
&= \sum_{i=1}^{n} x_{d_i} [\alpha_{d_i} q'_{d_i} - \alpha_{d_i} p_{d_i}(\mathbf{d})] \\
&= \sum_{i=1}^{n-1} [x_{d_i} - x_{d_{i+1}}] [\sum_{j=1}^{i} \alpha_{d_j} q'_{d_j} - \sum_{j=1}^{i} \alpha_{d_j} p_{d_j}(\mathbf{d})] + x_{d_n} [\sum_{j=1}^{n} \alpha_{d_j} q'_{d_j} - \sum_{j=1}^{n} \alpha_{d_j} p_{d_j}(\mathbf{d})].
\end{aligned}$$

By the LDF policy we know $x_{d_i} \geq x_{d_{i+1}}$. By (24) we have $\sum_{j=1}^{i} \alpha_{d_j} q'_{d_j} \leq \sum_{j=1}^{i} \alpha_{d_j} p_{d_j}(\mathbf{d})$ for $1 \leq i \leq n$. Therefore,

$$\mathrm{E}\left[\langle \boldsymbol{\alpha} \circ \mathbf{X}(t), \mathbf{q}' - \mathbf{Y}(t+1) \rangle | \mathbf{X}(t) = \mathbf{x}\right] \leq 0.$$

Suppose b is an upper bound for all α_i, q_i and possible $Y_i(t+1)$, by (25)

$$\mathrm{E}[L(\mathbf{X}(t+1)) - L(\mathbf{X}(t)) | \mathbf{X}(t) = \mathbf{x}] \leq 2nb^3 - 2\epsilon \langle \mathbf{x}, \boldsymbol{\alpha} \rangle \leq -1$$

for $\mathbf{x}$ satisfying $\langle \mathbf{x}, \boldsymbol{\alpha} \rangle \geq \frac{nb^3}{\epsilon} + \frac{1}{2\epsilon}$.

It is not hard to show[11] there are finite states $\mathbf{x}$ satisfying $\langle \mathbf{x}, \boldsymbol{\alpha} \rangle < \frac{nb^3}{\epsilon} + \frac{1}{2\epsilon}$. Therefore, by Foster's Theorem $\{\mathbf{X}(t)\}_{t \geq 1}$ is positive recurrent and $\mathbf{q}$ is fulfilled by the LDF policy.

8.2 Lower Bound in Theorem 4.3 is Tight

Given $\epsilon > 0$, consider a SRT-MIC system model that has $m = \lceil \frac{\frac{1}{\epsilon}+1}{2} \rceil$ identical cores serving $2m$ users generating tasks with deterministic workload w in each period of length $\delta = 2w - \frac{w}{m}$. Suppose all users have the same QoS requirement q.

In this setting, since $w \leq \delta \leq 2w$, by using LDF+Greedy one can complete m tasks per period. However, by using LDF+TS/LLREF policy introduced in Section 4.3 we can complete $\lceil \frac{m\delta}{w} \rceil = 2m-1$ tasks per period, which is a lower bound on the number of completed tasks per period under a feasibility optimal policy.

Given that all users have the same QoS requirement, the efficiency ratio of LDF+Greedy equals to ratio of the number of tasks completed per period under LDF+Greedy to that under a feasibility optimal policy, and thus

$$\gamma_{\text{LDF+Greedy}} \leq \frac{m}{2m-1}.$$

Since $m = \lceil \frac{\frac{1}{\epsilon}+1}{2} \rceil \geq \frac{\frac{1}{\epsilon}+1}{2}$, we know $\epsilon \geq \frac{1}{2m-1}$. Further since $\delta = 2w - \frac{w}{m}$, we get that

$$1 - \frac{w}{\delta} + \epsilon \geq 1 - \frac{1}{2 - \frac{1}{m}} + \frac{1}{2m-1} = \frac{m}{2m-1}.$$

Thus, in this setting, we have that

$$\gamma_{\text{LDF+Greedy}} \leq 1 - \frac{w}{\delta} + \epsilon = 1 - \frac{\max_{i \in N} \mu_i}{\delta} + \epsilon.$$

[11] This is true because given our assumption that requirement $\mathbf{q}$ are rational valued, the state space of process $\{\mathbf{X}(t)\}_{t \geq 1}$ is in a lattice (Conway and Sloane 2013).

8.3 Proof of Theorem 4.6

Suppose we are given a QoS requirement vector $\mathbf{q}$. Under deterministic workloads, to fulfill $\mathbf{q}$ the average core processing time $\sum_{i\in N} q_i\mu_i$ per period should not exceed $m\delta$. Therefore, a feasible requirement vector $\mathbf{q}$ implies

$$\sum_{i\in N} q_i\mu_i \le m\delta,$$

and clearly $\mathbf{q} \preceq \mathbf{1}$.

The goal is to show $\gamma_2\mathbf{q} \in \text{cl}(F_{\text{LDF+TS/LLREF}})$. Recall that in this setting the vector $\mathbf{p}(\mathbf{d})$ represents the expected numbers of task completions per period for TS/LLREF task scheduling under priority decision $\mathbf{d}$. Given deterministic workloads and any decision $\mathbf{d}$, under LDF+TS/LLREF, $p_i(\mathbf{d})$ equals to 1 if user i's task is selected and thus completes, and equals to 0 otherwise. By Theorem 4.2 it suffices to show $\gamma_2\mathbf{q} \in R_{\text{IB}}$ and by letting $\boldsymbol{\alpha} = (\mu_1, \mu_2, \cdots, \mu_n)$, it suffices to show for any given user subset $S \subseteq N$ and priority decision $\mathbf{d} \in D(S)$,

$$\sum_{i\in S} \mu_i p_i(\mathbf{d}) \ge \gamma_2 \sum_{i\in S} \mu_i q_i. \tag{26}$$

We show this in the following two cases.

If $\sum_{i\in S} \mu_i \le m\delta$, the task selection rule (14) will assure that all users in S are selected and thus, $p_i(\mathbf{d}) = 1$ for all $i \in S$. Since $\mathbf{q} \preceq \mathbf{1}$ and $\gamma_2 \le 1$, we have $\sum_{i\in S} \mu_i p_i(\mathbf{d}) = \sum_{i\in S} \mu_i \ge \gamma_2 \sum_{i\in S} \mu_i q_i$.

Otherwise, $\sum_{i\in S} \mu_i > m\delta$ and then not all users in S are selected. The task selection rule (14) will ensure

$$\sum_{i=1}^{j(\mathbf{d})} \mu_{d_i} \le m\delta < \sum_{i=1}^{j(\mathbf{d})+1} \mu_{d_i}$$

and therefore,

$$\sum_{i\in S} \mu_i p_i(\mathbf{d}) = \sum_{i=1}^{j(\mathbf{d})} \mu_{d_i} > m\delta - \max_{i\in N} \mu_i = m\delta(1 - \frac{\max_{i\in N} \mu_i}{m\delta}) = \gamma_2 m\delta \ge \gamma_2 \sum_{i\in N} \mu_i q_i \ge \gamma_2 \sum_{i\in S} \mu_i q_i.$$

This proves (26) and therefore,

$$\gamma_2\mathbf{q} \in R_{\text{IB}} \subseteq \text{cl}(F_{\text{LDF+TS/LLREF}}).$$