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

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

***Abstract*—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.**

I. 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, 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) [1], [2], [3] 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 [4]. Other SRT applications including collaborative video conferencing, multimedia processing, real-time control, augmented reality platforms, have similar characteristics.

This research was supported by Huawei Technologies Co. Ltd.

The computing infrastructure, e.g. [5], 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 [6], the community

has established theoretical frameworks to study the scheduling of hard real-time applications where all tasks need to complete by deadlines without violation [7], [8]. The results typically assume worst case execution times/workloads and are too conservative for SRT applications.

There are different models for the QoS needs of SRT applications. The work in [9], [10] propose the notion of (m, k)-firm deadlines requiring at least m out of any k consecutive tasks complete by their deadlines. But it typically assumes worst case workloads to get analytical result for such tight requirements. The authors in [11], [12] 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. The work in [13] aims to guarantee bounded deadline tardiness for all users. These frameworks and QoS models are incomplete or not suitable for applications like CRAN and video conferencing which require that most tasks complete by the deadlines.

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 [14] where the authors propose a static allocation approach to meet the QoS requirements. We shall use this as an evaluation benchmark. More recently, the authors in [12], [15] 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 policies that use the idea underlying longest-queue-first policies, whose performance are studied in [16], [17], [18] but in different settings. Moreover, the scheduling problem we consider is more than just one of ordering users according to some policies like largest-deficit-first. We also need to design the task scheduler to allocate resources to tasks across a computing system's cores.

Additional related work includes the problem of stochastic scheduling [19], and those studying the mixing of real-time and non real-time traffic, see e.g., [20], [21], [22].

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 II introduces our system model and Section III describes a reservation-based approach and a general outer bound for the feasibility region. Section IV discusses two prioritization-based policies and studies their efficiency ratios. Simulation results are exhibited in Section V. Section VI discusses generalizations and Section VII concludes the paper.

II. System Model

We first introduce our user, system and QoS models.

A. 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 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 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. [23] 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.

As a QoS requirement, each user i requires 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.

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 the extended version of this paper [24], we discuss possible generalizations where users may generate tasks with different periods and where a task may further consists of sub-tasks.

B. Computing Infrastructure

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 VI 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.

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.

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 events or repeated experiments. We shall only consider non-clairvoyant resource allocation policies.

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 system could be a cloud-based cluster of machines or a centralized server with a collection of processors/cores. There are many possible non-clairvoyant resource allocation policies which may involve exploiting knowledge of workload distributions, exploiting history events, preempting tasks at appropriate times, dynamically prioritizing tasks, etc.

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

C. 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? 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[2]

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

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 $t+1$. The deficit vector $\mathbf{X}(t) = (X_1(t), X_2(t), \cdots, X_n(t))$ 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". 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[3]. 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:* 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.

III. Reservation-Based Static Sharing and Outer Bound for the System Feasibility Region

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.

A. 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

[2]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.

[3]All the results in this paper can be generalized to a broader range of non-clairvoyant policies and $\mathbf{q}$'s with irrational values, see [24].

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

$$\Pr(W_i \leq w_i(q_i)) = q_i,$$

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. Fig.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.

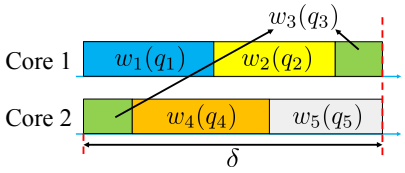

Fig. 1. An example of the reservation-based approach.

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

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

where $\mathbf{q} \preceq \mathbf{1}$ means $q_i \leq 1$ for all $i \in N$.

This approach was perhaps first proposed in [14] and is also loosely used in reservation based schemes adopted in modern cloud infrastructure, see e.g., [5]. 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.

B. Outer Bound for the System Feasibility Region F

Ideally we wish 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 $\text{int}(F_\eta) \subseteq F \subseteq \text{cl}(F_\eta)$, where $\text{int}(F_\eta)$ and $\text{cl}(F_\eta)$ is the interior and closure of F_η, and thus is equivalent to F for practical purposes.

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., [15]. 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** 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. See Appendix A for a detailed proof.

This simple outer bound applies to any non-clairvoyant resource allocation policy in any specific SRT-MIC system with NBUE workload distributions. Note however the result does not necessarily hold for non-NBUE workloads. See the extended version of this paper [24] for an illustrative example.

IV. 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 Fig.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.

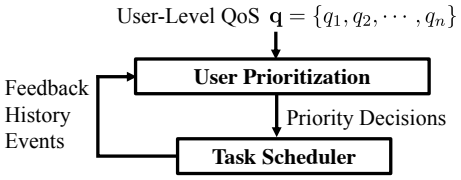

Fig. 2. The framework for prioritization-based resource allocation policies.

For user prioritization, we shall adopt 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** 3:* The **Largest Deficit First (LDF)** policy is such that, given the users' deficit vector $\mathbf{X}(t)$ at period $t+1$, it picks a priority decision $\mathbf{d}$ that satisfies

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

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.

A. 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 start with 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 subsets of users $S \subseteq N$, we let $|S|$ be the number of users in S and we define $D(S)$ to be the set of all priority decisions that assign the highest $|S|$ priorities to users in S. Now we have the following theorem.

***Theorem** 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 policy LDF+$\mathcal{X}$ is given by $\text{int}(R_{\text{IB}}) \subseteq F_{\text{LDF+}\mathcal{X}}$, 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}. For a detailed proof, see the extended version of this paper [24].

Next we explore specific task schedulers and use Theorem 2 to study their performance.

B. 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., [17]. The *efficiency ratio* of a non-clairvoyant resource allocation policy η is defined as

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

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

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

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 1, this result does not necessarily hold for non-NBUE workloads. A sketch of the proof of this result using Theorem 2 is included in Appendix B and detailed proof is provided in the extended version of this paper [24].

Theorem 3 provides a lower bound on the efficiency ratio of LDF+Greedy, denoted by $\gamma_{\text{LDF+Greedy}}$. The bound is tight in the sense that for any $\epsilon > 0$, there exists an SRT-MIC system with NBUE workloads such that $\gamma_{\text{LDF+Greedy}} < 1 - \frac{\max_{i \in N} \mu_i}{\delta} + \epsilon$. Such a system is also detailed in [24].

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, see [24], or prior work in [15]. 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.

C. 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 $i \in N$. Even for deterministic workloads, the scheduling of tasks in different periods to meet soft QoS requirements is not straightforward. 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, we are able to assess how many tasks we can complete prior to initiating processing. Indeed, it is intuitive, and established in [25], that one can complete all tasks in a user subset S in a period by some optimal scheduling if and only if $\sum_{i\in S}\mu_i \le m\delta$. We consider one such optimal algorithm: Largest Local Remaining Execution time First (LLREF) [25]. Let us briefly describe how LLREF would work in the SRT-MIC model and then introduce a task scheduler that combines the idea of task selection and this optimal scheduling.

To that end we introduce some terminology used in [25]. 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:* 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 LRE 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 [8].

***Definition** 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
$$j(\mathbf{d}) = \max\Big\{j|\sum_{i=1}^{j}\mu_{d_i} \le m\delta\Big\}. \tag{3}$$
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 [25], it follows that all tasks from $J(\mathbf{d})$ will complete.

Paralleling Theorem 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:* 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 3 and is provided in the extended version of this paper [24].

The efficiency ratio lower bound γ_2 in this theorem is better than γ_1 obtained in Theorem 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.

D. 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.

1) Resource Requirements for Reservation-Based Static Sharing:

Based on the definition of F_{RB} in III-A, the required number of cores to fulfill the users' QoS requirements $\mathbf{q}$ under reservation-based static sharing is given by
$$m_{\text{RB}} = \Big\lceil \frac{\sum_{i\in N} w_i(q_i)}{\delta} \Big\rceil, \tag{4}$$
where $\lceil x\rceil$ is the ceiling of x.

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 1, we know m_η must satisfy $m_\eta\delta \ge \sum_{i\in N} q_i\mu_i$, giving the following lower bound on the required number of cores:
$$\underline{m} \equiv \Big\lceil \frac{\sum_{i\in N} q_i\mu_i}{\delta} \Big\rceil. \tag{5}$$

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 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,

$$m^{\text{est}}_{\text{LDF+Greedy}} \equiv \left\lceil \frac{\sum_{i\in N} q_i\mu_i}{\delta - \max_{i\in N}\mu_i} \right\rceil. \tag{6}$$

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

One 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 [24]. We observe that the inequality holds true in the various simulation settings considered next.

V. 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.

A. 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 Fig.3. We choose this NBUE workload distribution as a representative one.

In the bottom panel in Fig.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. Intuitively, in this homogeneous-user scenario, if we ignore the ceilings in (4) (5), the upper bound on savings becomes,

$$1 - \frac{\underline{m}}{m_{\text{RB}}} \simeq 1 - \frac{q\mu}{w(q)}, \tag{7}$$

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%.

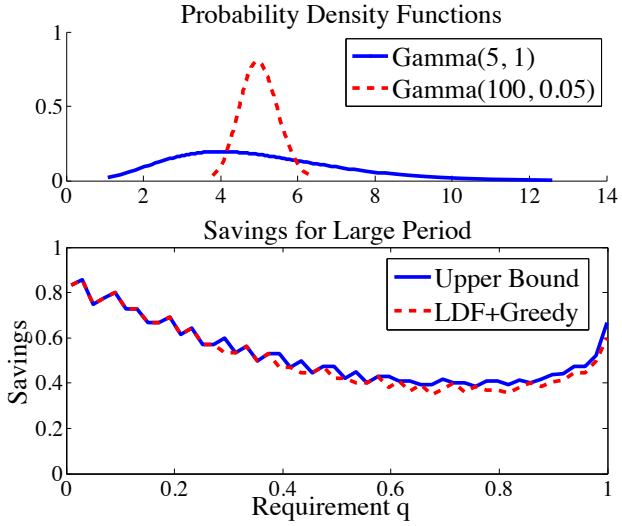

Fig. 3. Top: the probability density functions for Gamma$(5, 1)$ and Gamma$(100, 0.05)$. Bottom: the resource savings for large period.

B. 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 Fig.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 Fig.3. Intuitively, this is because for deterministic workloads, by (7) we know $w(q)$ equals to μ and thus we get

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

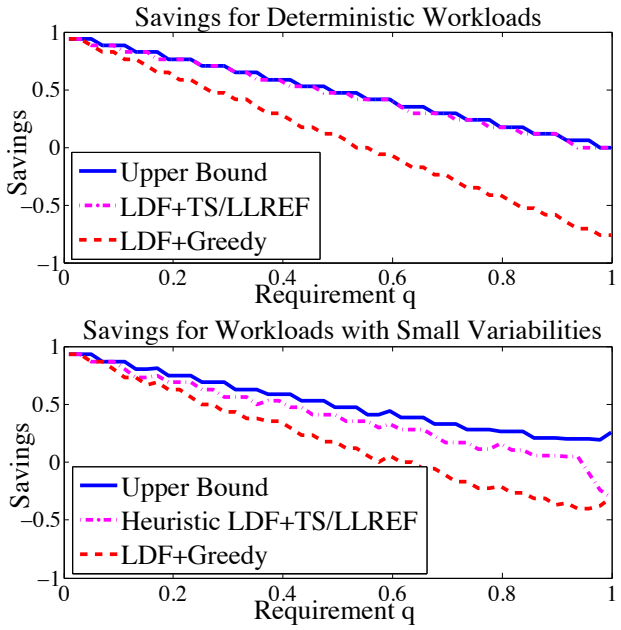

Fig. 4. Top: the resource savings under deterministic workloads. Bottom: the resource savings under random workloads with small variability.

C. LDF+TS/LLREF for Workloads with Small Variability

For workloads with small variability, we envisage that the heuristic LDF+TS/LLREF described in Section IV-C 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 Fig.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 IV-C. We conduct the same analysis for resource savings and exhibit the results in the bottom panel in Fig.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.

VI. Generalizations to Cores with Different Processing Speeds

In this section we summarize generalizations of our results to systems consisting of cores with 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., [8]. 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. In the model we have considered, $s_c = 1$ for all $c \in C$.

In this setting, the outer bound R_{OB} in Theorem 1 becomes

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

For LDF+Greedy, given that cores have different processing speeds, one may want to migrate tasks to faster cores if possible. Therefore, depending on whether task preemption/migration is allowed, there are two types of LDF+Greedy. In Type 1, the task scheduler greedily and preemptively schedules tasks with the highest priority on the fastest cores. In this setting, γ_1 in Theorem 3 becomes $\gamma_1 = 1 - \frac{\max_{i\in N}\mu_i}{\overline{s}\cdot\delta}$. In Type 2, the task scheduler greedily places tasks on available cores by priority but does not allow preemption/migration. Here γ_1 becomes $\gamma_1 = 1 - \frac{\max_{i\in N}\mu_i}{\min_{c\in C} s_c\cdot\delta}$.

We can also generalize LDF+TS/LLREF and Theorem 4 with additional natural assumptions. See the extended version of this paper [24] for more details and other generalizations.

VII. 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

[1] China Mobile, "C-RAN The Road Towards Green RAN," Oct 2011.
[2] C. J. Bernardos *et al.*, "An architecture for software defined wireless networking," *IEEE Wireless Communications*, vol. 21, no. 3, 2014.
[3] Y. Du and G. de Veciana, "Wireless Networks Without Edge: Dynamic Radio Resource Clustering and User Scheduling," in *INFOCOM 2014*.
[4] Personal communication with Alan Gatherer.
[5] A. Verma *et al.*, "Large-scale cluster management at Google with Borg," in *Proc. of EuroSys 2015*.
[6] C. L. Liu and J. W. Layland, "Scheduling Algorithms for Multiprogramming in a Hard Real-Time Environment," *J. of the ACM*, 1973.
[7] J. W. S. Liu, *Real-Time Systems*. Prentice Hall, April 2000.
[8] R. I. Davis and A. Burns, "A Survey of Hard Real-Time Scheduling for Multiprocessor Systems," *ACM Computing Surveys*, vol. 43, 2011.
[9] M. Hamdaoui *et al.*, "A Dynamic Priority Assignment Technique for Streams with (m, k)-Firm Deadlines," *IEEE Trans. Comput.*, 1995.
[10] P. Ramanathan, "Overload management in real-time control applications using (m, k)-firm guarantee," *IEEE Trans. Parallel Distrib. Syst.*, 1999.
[11] J. W. Liu, K.-J. Lin, and S. Natarajan, "Scheduling Real-time, Periodic Jobs Using Imprecise Results," in *Proc. of RTSS 1987*.
[12] I.-H. Hou and P. R. Kumar, *Packets with Deadlines: A Framework for Real-Time Wireless Networks*. Morgan & Claypool Publishers, 2013.
[13] C. Liu and J. H. Anderson, "Task Scheduling with Self-Suspensions in Soft Real-Time Multiprocessor Systems," in *Proc. of RTSS 2009*.
[14] A. Atlas and A. Bestavros, "Statistical Rate Monotonic Scheduling," in *Proc. of RTSS 1998*.
[15] I.-H. Hou and P. R. Kumar, "Queueing systems with hard delay constraints: a framework for real-time communication over unreliable wireless channels," *Queueing Systems*, 2012.

[16] A. Dimakis and J. Walrand, "Sufficient Conditions for Stability of Longest-Queue-First Scheduling: Second-Order Properties Using Fluid Limits," *Advances in Applied Probability*, vol. 38, no. 2, June 2006.
[17] C. Joo *et al.*, "Performance Limits of Greedy Maximal Matching in Multi-hop Wireless Networks," in *IEEE Conf Decis Control*, 2007.
[18] X. Kang *et al.*, "On the Performance of Largest-Deficit-First for Scheduling Real-Time Traffic in Wireless Networks," in *MobiHoc 2013*.
[19] M. L. Pinedo, *Scheduling: Theory, Algorithms, and Systems*, 2012.
[20] S. Shakkottai and A. L. Stolyar, "Scheduling algorithms for a mixture of real-time and non-real-time data in HDR," in *Proc. of ITC 2001*.
[21] J. J. Jaramillo and R. Srikant, "Optimal Scheduling for Fair Resource Allocation in Ad Hoc Networks With Elastic and Inelastic Traffic," *IEEE Trans. on Networking*, vol. 19, 2011.
[22] S. Patil and G. de Veciana, "Managing Resources and Quality of Service in Heterogeneous Wireless Systems Exploiting Opportunism," *IEEE Trans. on Networking*, 2007.
[23] A. Müller and D. Stoyan, *Comparison Methods for Stochastic Methods and Risks*. Wiley, March 2002.
[24] Extended version. [Online]. Available: http://arxiv.org/abs/1601.06333
[25] H. Cho, B. Ravindran, and E. D. Jensen, "An Optimal Real-Time Scheduling Algorithm for Multiprocessors," in *Proc. of RTSS 2006*.

APPENDIX

A. Proof of Theorem 1

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

Suppose $\mathbf{q}$ is fulfilled by a non-clairvoyant resource allocation policy η, by definition $\{\mathbf{X}(t)\}_{t\ge 1}$ 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 user i's task completes in a typical period. By the Ergodic Theorem, $\mathrm{E}[Y_i]$ 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 $\mathrm{E}[Y_i]$. For each user subset $S \subseteq N$, we define U_S to be a random variable denoting the total time spent on users in S in a typical period. Clearly, $\mathrm{E}[U_S]$ cannot exceed the total available core time $m\delta$. To show $\sum_{i\in N} q_i\mu_i \le m\delta$, it suffices to show that $\sum_{i\in N} \mathrm{E}[Y_i]\mu_i \le \mathrm{E}[U_N]$. To that end we first develop an equation connecting $\sum_{i\in N} \mathrm{E}[Y_i]\mu_i$ and $\mathrm{E}[U_N]$, 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.

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

$$U_S = \sum_{i\in S}(Y_i + A_i)W_i - \sum_{i\in S} E_i. \tag{8}$$

Clearly $Y_i + A_i$, which indicates that user i's task starts processing, is independent of W_i. Indeed this follows from the non-clairvoyance of the policy η 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. 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, $\mathrm{E}[Y_i | W_i > \delta] = 0 \ne \mathrm{E}[Y_i]$.

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

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

By taking expectations on both sides of (8), we get

$$\mathrm{E}[U_S] = \sum_{i\in S}\mathrm{E}[Y_i]\mu_i + \sum_{i\in S}\mathrm{E}[A_i]\mu_i - \sum_{i\in S}\mathrm{E}[E_i]. \tag{9}$$

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 $\sum_{i\in N} \mathrm{E}[Y_i]\mu_i \le \mathrm{E}[U_N]$, by (9) it suffices to show $\mathrm{E}[A_i]\mu_i \ge \mathrm{E}[E_i]$ 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.

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

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

where $\mu_{i,c} = \mathrm{E}[W_i - c | W_i > c]$.

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

$$\mathrm{E}[E_i] \le \sum_{c=1}^{\delta}\mu_i\mathrm{E}[A_{i,c}] = \mu_i\mathrm{E}[A_i]. \tag{10}$$

To summarize, by (9) and (10) we know for all $S \subseteq N$,

$$\sum_{i\in S} q_i\mu_i \le \sum_{i\in S}\mathrm{E}[Y_i]\mu_i \le \mathrm{E}[U_S] \le m\delta, \tag{11}$$

which implies $\sum_{i\in N} q_i\mu_i \le m\delta$, and thus, $F \subseteq R_{\mathrm{OB}}$.

B. Sketch of Proof of Theorem 3

Given a feasible $\mathbf{q}$, the goal is to show that $\gamma_1\mathbf{q} \in \mathrm{cl}(F_{\text{LDF+Greedy}})$. By Theorem 2, it suffices to show $\gamma_1\mathbf{q} \in R_{\mathrm{IB}}$. We pick $\boldsymbol{\alpha} = (\mu_1, \mu_2, \cdots, \mu_n) \succ \mathbf{0}$ and it suffices to show for all user subsets S and priority decisions $\mathbf{d} \in D(S)$ that

$$\sum_{i\in S}\mu_i p_i(\mathbf{d}) \ge \gamma_1 \sum_{i\in S}\mu_i q_i. \tag{12}$$

This is shown by results similar to (9), (11).