The Dissertation Committee for Yuhuan Du
certifies that this is the approved version of the following dissertation:

Analysis and Design of Resource Allocation Policies for Cloud-Based Computing Systems Supporting Soft Real-Time Applications

Committee:

Gustavo de Veciana, Supervisor

Sanjay Shakkottai

Robert W. Heath Jr.

François Baccelli

C. Greg Plaxton

Analysis and Design of Resource Allocation Policies for Cloud-Based Computing Systems Supporting Soft Real-Time Applications

by

Yuhuan Du, B.E.; M.S.E.

DISSERTATION

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

DOCTOR OF PHILOSOPHY

THE UNIVERSITY OF TEXAS AT AUSTIN

December 2015

Dedicated to my parents and my wife.

Acknowledgments

Foremost, I would like to express my sincerest gratitude to my PhD supervisor Dr. Gustavo de Veciana, for his continuous support and guidance during my PhD tenure at UT. I am fortune to have such a great supervisor and I have greatly benefited from his steadfast knowledge, scientific keenness, and extraordinary personality. Thanks to his patience and constant support, my PhD life has been a truly enjoyable journey and I could not have imagined a better supervisor, mentor, and lifetime friend.

I would also like to say thanks to Dr. Sanjay Shakkottai, Dr. Robert W. Heath Jr., Dr. François Baccelli and Dr. C. Greg Plaxton for serving on my dissertation committee, and for their valuable time and helpful comments on my research work. I thank Dr. Jeff Andrews, Dr. Constantine Caramanis, Dr. Sujay Sanghavi, Dr. Lorenzo Alvisi, Dr. Adnan Aziz, Dr. Craig Chase, Dr. Dewayne E. Perry, Dr. Evdokia Nikolova, and several other faculty members at UT who have influenced me and broadened my scientific horizons through their excellent courses.

I gratefully acknowledge Huawei and National Science Foundation for financially supporting my research. I would like to say thanks to Dr. Alan Gatherer for his support and suggestions through the CRAN project, Dr. Mattan Erez and Haishan Zhu for the collaboration and all the discussions.

I would also like to thank Dr. Carl Edlund and Mr. Donald Lau for serving as my summer intern mentors at Microsoft and Yelp during summer 2012 and 2014. These experiences helped relate my research to the industry and will remain beneficial for my future career.

My life at UT would never have been so memorable and fun without my brilliant friends. I thank Virag, Vinay, Arjun, Pranav and Pablo for being helpful group mates. I would like to say special thanks to my friends who share the most enjoyable time with me: Zheng, Yicong, Jiaxiao, Tianyang, Chao C., Hongbo, Qiaoyang, Jing, Yingxi, Ping, Yudong, Cong, Chao J., Xinyang, Yajun, Yingzhe, Tong, Jianhua, Nan, Qi, Xingqin and many others. I will remember the time we watched and played soccer/basketball, our routine dinners, our Happy Friday Nights, and all the wonderful moments we had fun together.

Finally, I will always be indebted to my dear family: mom, dad, and Yang, for their generous love and unconditional support.

Analysis and Design of Resource Allocation Policies for Cloud-Based Computing Systems Supporting Soft Real-Time Applications

Publication No. ____________________

Yuhuan Du, Ph.D.
The University of Texas at Austin, 2015

Supervisor: Gustavo de Veciana

Cloud-based computing infrastructure can provide an efficient means to support real-time applications with compute and/or communication deadlines, e.g., virtualized base station processing, and collaborative video conferencing. In many cases, such applications can tolerate occasional deadline violations without substantially impacting their Quality of Service (QoS). A fundamental problem in such systems is deciding how to allocate shared resources so as to meet applications' QoS requirements. A simple framework to address this problem is to, (1) dynamically prioritize users as a possibly complex function of their deficits (difference of achieved vs required QoS), and (2) allocate resources so as to expedite users assigned higher priority.

In the first part of this dissertation, we focus on a general class of systems using such priority-based resource allocation. In this setting we characterize the set of feasible QoS requirements. We then consider "simple" weighted

Largest Deficit First (**w**-LDF) prioritization policies, where users with higher weighted deficits are given higher priority. We give an inner bound for the feasible set of QoS requirements under **w**-LDF policies, and characterize its geometry under an additional monotonicity assumption. Additional insights on the optimality of LDF/hierarchical-LDF are also discussed.

In the second part of this dissertation, we consider a specific class of computing systems having multiple uniform resources. We develop a general outer bound on the feasible QoS region for non-clairvoyant resource allocation policies, and study the efficiency and near-optimality 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. Analysis and simulations show substantial resource savings for such policies over reservation-based designs. We also discuss user/stream management for computing systems supporting soft real-time users.

Overall, the main contribution of this dissertation is a theoretical study on the efficiency and optimality of simple deficit-based resource allocation policies for systems supporting periodically generated, but stochastic workloads requiring soft guarantees on completion times.

Table of Contents

List of Tables

List of Figures

Chapter 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 dissertation, we focus on resource allocation for 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) [6, 13, 26] being considered for next generation cellular deployments, see Figure 1.1. 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 [29].

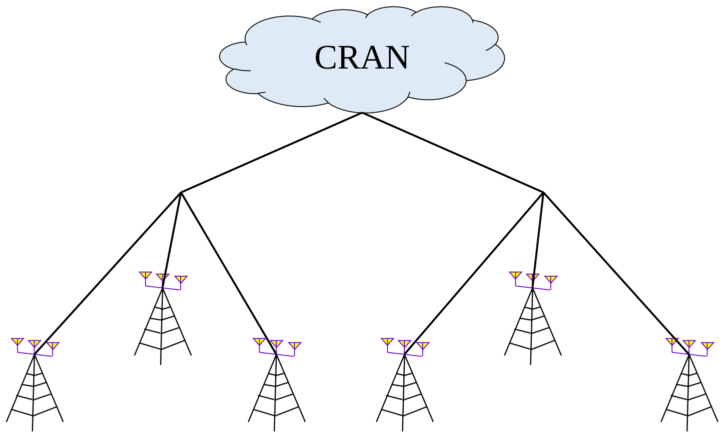

Figure 1.1: CRAN architecture.

Another example of an SRT application are services associated with distributed multi-party collaborative video conferencing or educational applications. In this setting multiple participants at various locations send their video and content to a processing center where it is combined, tailored, transcoded and sent back to distributed attendees with possibly different resolutions or points of view, etc. In interactive settings, one must ensure small end-to-end delays and thus tight processing delays. Still in many cases, it is acceptable that some video frames not be delivered on time without substantially impacting user perceived quality of experience.

Google glass and other augmented reality platforms share similar characteristics to the above examples. In such applications, a stream of local observations including video/sound could be sent to computing centers for processing, e.g., face and activity recognition, and results returned for display. To ensure “fluidity” the turnaround for such processing must be quite tight yet such applications may tolerate occasional failures in meeting processing deadlines if they are handled properly.

Other SRT applications include multimedia processing, and real-time control, transportation, and power networks.

The computing infrastructure, e.g. [68], 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. It typically requires a load balancer to distribute application traffic across a number of resources and to achieve hardware resource virtualization so users can focus on application development. In other words, the challenge is to orchestrate a complex collection of resources to efficiently meet applications’ SRT requirements. In this dissertation we mainly focus on a single computing system, e.g., managed server/center, shared by a set of users, corresponding to instances of SRT applications, and explore the fundamental problem of allocating the shared resources efficiently so as to meet users' QoS requirements.

In Chapter 2, we study an approach to resource allocation based on decomposition of concerns: (1) user priorities are dynamically set based on the history outcomes, and (2) resources are allocated so as to favor users with higher priority. This chapter focuses on a general class of systems using such priority-based resource allocation. We first characterize the set of feasible QoS requirements and show the optimality of max weight-like prioritization. We then consider simple weighted Largest Deficit First (**w**-LDF) prioritization policies, where users with higher weighted deficits are given higher priority. In this chapter we develop an inner bound for the feasible set under **w**-LDF policies, and, under an additional monotonicity assumption, characterize its geometry leading to a sufficient condition for optimality. Additional insights on the efficiency ratio of **w**-LDF policies, the optimality of hierarchical-LDF and characterization of clustering of failures are also discussed.

In Chapter 3, we consider a specific class of computing systems having multiple uniform resources and address the problem of resource allocation to support heterogeneous soft real-time applications subject to QoS constraints. 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, i.e., the prioritization schemes discussed in Chapter 2. 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. User/stream management for computing systems supporting SRT users are also discussed.

In Chapter 4, we conclude the dissertation and point to other open research challenges beyond scheduling in architecting cloud-based computing systems to support heterogeneous soft real-time applications.

Chapter 2

Efficiency and Optimality of Largest Deficit First Prioritization for General System Model

2.1 Introduction

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

This chapter is based on my work in [22] (submitted to INFOCOM 2016) and [23] (submitted to ACM Transactions on Modeling and Performance Evaluation of Computing Systems), which are co-authored by Dr. Gustavo de Veciana.

retransmitted. More generally real-time applications' long-term QoS may depend in a complex manner on what was accomplished on time, e.g., partial completion of a set of tasks, or notions of video quality.

Enabling efficient sharing of compute/communication resources is a challenging problem. On the one hand, even for a single resource, tying the sharing model, e.g., round robin or priority schemes, to QoS metrics is generally hard due to the uncertainty in applications' workloads and possible variations in processing speeds. On the other hand, today's applications leverage complex networks of heterogeneous compute/communication resources, e.g., multi-core computers, embedded network system, or combinations of computation on mobile devices and the cloud. Consider the example in Figure 2.1. User 1 periodically generates a task that needs to be processed sequentially on Resources A, B, C, D in each period while User 2 generates tasks to be processed on Resource B then C. How should one go about designing resource sharing policies across multiple heterogeneous resources, where parallelism, task preemption and migration are allowed? Furthermore, how can one address heterogeneous QoS requirements associated with real-time applications? For example, User 1's QoS may still benefit from partial completions while User 2 only benefits if all processing is completed. This general class of problems involving both heterogeneous resources and user QoS requirements is the focus of this chapter.

The design space of possible solutions to this problem is huge and has been explored in many research communities. In this chapter we study an

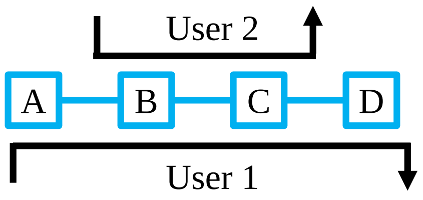

Figure 2.1: An example for a network of resources. A, B, C and D represent compute/communication resources. Tasks from User 1 need to be processed on A, B, C, D while tasks from User 2 require processing on B, C.

approach to resource allocation based on a decomposition of concerns:

1. user priorities are dynamically set based on the history outcomes;

2. and, resources are allocated so as to favor users with higher priority.

In such a framework there is quite a bit of latitude in choosing how priorities are set, and in turn how these affect the allocation of resources. For example, users' priorities could be set based on measured deficits, the "difference" of the required and achieved QoS, i.e., Largest Deficit First (LDF) prioritization. In turn, for a complex system, such as that in Figure 2.1, resources could be allocated greedily giving preemptive access to tasks associated with higher-priority users.

In general an optimal user prioritization strategy could leverage detailed information regarding how these priorities will impact the allocation of resources and the completion outcomes to achieve the best possible user QoS. Such strategies require excessive amounts of information regarding the underlying compute/communication resources and resource allocation mechanism, and thus are generally hard to implement. By contrast, LDF-based prioritization is quite intuitive. It requires only tracking of users' possibly heterogeneous QoS deficits, in this sense it is truly decoupling user prioritization from the underlying priority-based resource allocation. Unfortunately, it is known to be suboptimal in certain settings [19, 36, 38].

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

Related Work. There have been much work studying dynamic prioritization policies in the context of diverse resource, workload and/or QoS models.

The authors in [32, 33] propose a framework to model a wireless access point serving a set of clients that in each period generate packets which need to be transmitted by the end of the period. In their model only one client can transmit at a time and thus the access point can be viewed as a single resource. Each client transmits its packets over an unreliable channel which has a fixed probability of success, and thus, the time to successfully transmit a packet can be modeled as a geometric random variable. In this setting the authors show that the LDF policy is "optimal." However, the results are restricted to a single resource shared by users with geometric workloads. In this chapter we study the performance of LDF in a more general setting which includes this prior work as a special case. This initial set of papers motivated follow-up work in wireless context, see e.g., [34, 35, 54].

The performance of LDF and similar policies has also been studied in [19, 36, 38, 49]. The authors in [19] consider the generalized switch model and were the first to propose the notion of "local pooling" as a sufficient condition for the Longest Queue First (LQF) policy to be throughput optimal. Subsequently, the work in [36] considers a multi-hop wireless network under a node-exclusive interference model and shows that the efficiency ratio of the greedy maximal matching policy, which is essentially LQF, equals to the "local pooling" factor of the network graph. More recently, the authors in [38] consider real-time traffic in ad hoc wireless networks under a link-interference model and also characterize the efficiency ratio of the LDF policy.

The results in [19, 36, 38] depend on the constant service rate model and the specific interference model, i.e., where the set of links/queues that can be scheduled simultaneously is restricted. These models may be appropriate in some wireless/queueing networks but do not necessarily hold in our broader context, e.g., soft real-time applications with stochastic workloads. Also, [19] lacks a performance analysis of LQF when it is not optimal and the works in [36, 38] focus on the efficiency ratio of LDF-like policies but lack a characterization of the full capacity region of these policies. Moreover, when the system can deliver more than the requirements, either the QoS requirements for real-time traffic or the throughput requirements for queueing systems, there is no discussion of how to manage the allocation of the "excess capacity" across users.

The authors in [16, 50, 64–66] propose max weight scheduling policies for different types of queueing systems and show them to be throughput optimal via the approaches summarized in [20, 21, 52]. The authors in [55] and [67] further characterize the delay of the max weight policy, and study its inefficiency in spatial wireless networks, respectively. As we will see in the sequel we too discuss a max weight-like scheduling policy, but it suffers from the usual complexity problems when the decision space is large and it requires excessive amounts of information, motivating us to consider simpler policies.

Additional related work includes work on modeling and scheduling of real-time tasks, see e.g., [17, 41, 59, 61].

Attempts have been made to incorporate many of the important research papers into our bibliography. Unfortunately we are almost certain to have missed some relevant research articles. We apologize for overlooking papers which should have been included.

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

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

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

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

Organization of the Chapter. This chapter is organized as follows: Section 2.2 introduces our general model for systems supporting priority-based resource allocation. Section 2.3 develops theoretical results and characterizes the performance of weighted LDF policies while Section 2.4 presents some examples for the optimality of the weighted LDF/hierarchical-LDF policies. Section 2.5 discusses some practical issues while the impact of weights is evaluated via simulation in Section 2.6. Section 2.7 concludes this chapter and points to future work. Some of the proofs are provided in the Appendix.

2.2 System Model

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

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

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

2.2.1 General Model for Systems Supporting Priority-Based Resource Allocation

We consider an abstract system that serves n users indexed from 1 to n. Let $N = \{1, 2, \cdots, n\}$ be the user set. The system operates in discrete time, over periods $t = 1, 2, \cdots$. In each period, it picks a user *priority decision*

$\mathbf{d} = (d_1, d_2, \ldots, d_n)$ where d_m is the index of the user with m^{th} highest priority. We let D denote the set of all possible priority decisions and let $|D|$ represent the number of possible decisions, thus, $|D| = n!$

In each period, given the priority decision $\mathbf{d}$ passed to the underlying resources, since there are intrinsic uncertainties in users' workloads, each user i achieves a non-negative random QoS payoff, denoted by $V_i(\mathbf{d})$. We let $\mathbf{V}(\mathbf{d}) = (V_1(\mathbf{d}), V_2(\mathbf{d}), \cdots, V_n(\mathbf{d}))$. We assume the payoffs are independent across periods. The distribution of $\mathbf{V}(\mathbf{d})$ depends on the selected priority decision $\mathbf{d}$ and the expected payoff vector given $\mathbf{d} \in D$ is denoted by $\mathbf{p}(\mathbf{d}) = \mathrm{E}[\mathbf{V}(\mathbf{d})]$. We assume all possible payoff vectors form a finite rational set. Moreover, we naturally assume that for each user $i \in N$, there exists a decision $\mathbf{d}$ such that $p_i(\mathbf{d}) > 0$.

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

$$X_i(t+1) = [X_i(t) + q_i - V_i(\mathbf{d}(t+1))]^+, \tag{2.1}$$

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

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

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

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

Definition 2.2.1. A **user prioritization policy** is a stationary policy that picks a priority decision $\mathbf{d}(t+1) \in D$ at period $t+1$ based on the following:

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

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

Definition 2.2.2. A payoff requirement vector $\mathbf{q}$ is said to be **feasible** if there exists a user prioritization policy η under which the Markov chain $\{\mathbf{X}(t)\}_{t\geq 1}$ is positive recurrent. We also say this policy fulfills this requirement vector.

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

Note that this model is general in the sense that the "impact" of priority decisions $\mathbf{d} \in D$ on the QoS payoff vectors P is at this point general, whereas the specific resource and workload models in prior work, e.g., [19, 36, 38], implicitly impose properties on P and therefore restrict the results significantly.

2.2.2 Summary of System Model and Assumptions

To summarize, the problem is to develop user prioritization policies to meet users' long-term payoff requirements and here are the assumptions:

- The system serves n users.
- In each period, the system picks a priority decision $\mathbf{d}$.
- Given the selected decision $\mathbf{d}$, the random payoff vectors $\mathbf{V}(\mathbf{d})$ are independent across periods and the distribution depends on $\mathbf{d}$. We assume all possible payoff vectors form a finite rational set.
- We let $\mathbf{p}(\mathbf{d}) = \mathrm{E}[\mathbf{V}(\mathbf{d})]$. We assume for each user $i \in N$, there exists a decision $\mathbf{d}$ such that $p_i(\mathbf{d}) > 0$.
- $\mathbf{q}$ is the long-term QoS requirement vector. We assume q_i's are rational.
- User prioritization policies pick decisions based on $\mathbf{q}$, $P = \{\mathbf{p}(\mathbf{d}) | \mathbf{d} \in D\}$ and $\mathbf{X}(t)$.
- We further assume the initial state $\mathbf{X}(0)$, the requirements $\mathbf{q}$, the set of all possible payoff vectors and the user prioritization policy make $\{\mathbf{X}(t)\}_{t \geq 1}$ an irreducible Markov chain.

2.2.3 Example: Centralized Computing System Supporting Real-Time Applications

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

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

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

2.2.4 Example: Complex Networks and Flexible Modeling of Application Execution Payoffs

As indicated in the introduction, our model also applies to a complex network of heterogeneous compute and communication resources, as long as users periodically and synchronously generate tasks that require timely processing on diverse resources and moving around in the network, e.g., as shown in Figure 2.1.

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

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

2.3 Performance Analysis

In this section we shall develop theoretical results for such systems. Some of these results are similar to prior work but in the more general model while other results are completely new. For completeness we shall develop a self-contained theoretical framework.

2.3.1 System Feasibility Region and Feasibility Optimal Policy

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

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

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

$$C \equiv \{\mathbf{q} \in \mathbb{R}^n_+ \mid \exists \mathbf{x} \in \text{Conv}(P) \text{ such that } \mathbf{q} \preceq \mathbf{x}\}. \tag{2.2}$$

Figure 2.2 exhibits C for a two-user (left figure) and three-user (right figure) setting. In the two-user setting, the points labeled $\mathbf{p}(\mathbf{d}_1)$ and $\mathbf{p}(\mathbf{d}_2)$ are the expected payoff vectors of two priority decisions, i.e., where User 1 or User 2 has higher priority, respectively. The shadowed area represents C. In the three-user setting, the circles represent the expected payoff vectors associated with the 6 possible priority decisions for three users, and the region dominated by their convex hull is C. Note that in an n-user scenario where $n \geq 3$, as displayed the expected payoff vectors need not be on a hyperplane in the n-dimensional space. As we will see this is essentially the source of complexity in studying such systems.

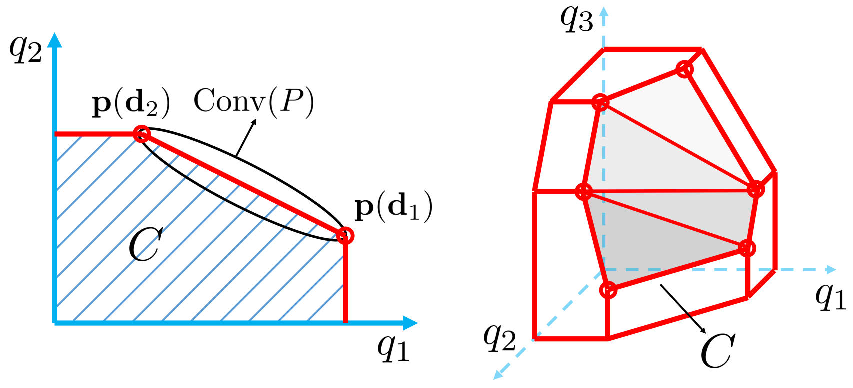

Figure 2.2: Examples of set C when $n = 2$ and $n = 3$.

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

Lemma 2.3.1. *The system feasibility region F is such that*

$$F \subseteq cl(C),$$

where $cl(C)$ is the closure of C.

Intuitively, if $\mathbf{q}$ is feasible, it is fulfilled by some user prioritization policy that in the long-term picks each priority decision some fraction of the time and thus, $\mathbf{q}$ is dominated by some point in the convex hull of P. This is similar to prior work, e.g., [65]. See Appendix 2.8.1 for a detailed proof. In other words, C is different from F by at most a boundary, and therefore, characterizes F for practical purposes. Thus, in the sequel we will also refer to C as the system feasibility region.

Ideally, it is desirable to devise an "optimal" policy that can fulfill all feasible requirements. More formally, a user prioritization policy η is said to be *feasibility optimal* if $\mathrm{int}(C) \subseteq F_\eta \subseteq \mathrm{cl}(C)$. Similar to prior work [65, 66], the following max weight-like policy is one such feasibility optimal policy.

Definition 2.3.1. The **deficit-based max weight (MW)** prioritization policy is such that, at period $t+1$, given the deficit vector $\mathbf{X}(t)$ computed by (2.1), it picks a priority decision $\mathbf{d}(t+1)$ that satisfies

$$\mathbf{d}(t+1) \in \arg\max_{\mathbf{d}\in D} \langle \mathbf{X}(t), \mathbf{p}(\mathbf{d})\rangle, \tag{2.3}$$

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

Theorem 2.3.2. *The system feasibility region F and the feasibility region of the MW policy F_{MW} are related to C as follows,*

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

and therefore, the MW policy is feasibility optimal.

See Appendix 2.8.2 for the proof.

However, the MW policy and time sharing policies require full knowledge of P which is challenging in complex practical systems. Moreover, these policies are hard to implement since they involve solving fairly complex optimization problems, i.e., Eq (2.3). Changes in the user set or payoff requirement vector $\mathbf{q}$ will also impact the realization of these policies. In summary, the requirements in terms of a-priori knowledge, the computational complexity and lack of flexibility to changes make them hard to use in practice. This motivates the policies considered in the next subsection.

For ease of reference, Table 2.1 provides a summary of the notation used to denote various regions identified in the rest of this chapter—some of these are introduced in the sequel.

Table 2.1: Notation of regions.

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

2.3.2 Weighted LDF Policies and Associated Feasibility Regions

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

Definition 2.3.2. Given a vector $\mathbf{w} = (w_1, w_2, \cdots, w_n) \succ \mathbf{0}$, the **weighted Largest Deficit First (w-LDF)** user prioritization policy is such that, at period $t+1$, given the deficit vector $\mathbf{X}(t)$, it picks a priority decision $\mathbf{d}$ that satisfies

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

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

Clearly, the $\mathbf{w}$-LDF prioritization policies do not require knowledge of the expected payoff vectors P. In terms of computational complexity, solving (2.3) is $O(n!)$ while sorting weighted deficits only requires $O(n \log n)$. It also allows us to further differentiate the performance across users by assigning different weights. The impact of weights is discussed in Section 2.6.

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

Theorem 2.3.3. *For any $\mathbf{w} \succ \mathbf{0}$, an inner bound for the feasibility region of the $\mathbf{w}$-LDF policy $F_{\mathbf{w}\text{-}LDF}$ is given by $int(R_{IB}) \subseteq F_{\mathbf{w}\text{-}LDF}$, where*

$$\begin{aligned} R_{IB} \equiv \{\mathbf{q} \in \mathbb{R}^n_+ \mid \exists \boldsymbol{\alpha} \succ \mathbf{0} \text{ such that } \forall S \subseteq N, \\ \sum_{i \in S} \alpha_i q_i \leq \min_{\mathbf{d} \in D(S)} \sum_{i \in S} \alpha_i p_i(\mathbf{d})\} \end{aligned} \tag{2.4}$$

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

In other words, if $\mathbf{q} \in R_{\text{IB}}$, it is feasible under all $\mathbf{w}$-LDF policies except perhaps boundary points. The underlying intuition for this bound is as follows. A vector $\mathbf{q}$ is in R_{IB} if there is a weight vector $\boldsymbol{\alpha} \succ \mathbf{0}$ such that for any subset of users S, and decisions giving users in S the highest priorities, the weighted sum of payoff requirement $\sum\limits_{i \in S} \alpha_i q_i$ will not exceed the least sum weighted payoff $\sum\limits_{i \in S} \alpha_i p_i(\mathbf{d})$. Based on $\boldsymbol{\alpha}$, we can construct an appropriate Lyapunov function to show feasibility for $\mathbf{q}$ and each $\mathbf{w}$. See Appendix 2.8.3 for the proof.

Understanding the geometry of R_{IB} enables us to characterize the performance gap between $\mathbf{w}$-LDF and feasibility optimal policies. Let us informally consider the geometry of R_{IB} for the two special cases in Figure 2.2. In the two-user case in Figure 2.2, R_{IB} is the same as C and thus, the $\mathbf{w}$-LDF policies are always feasibility optimal. However, in the three-user case in Figure 2.2, this need not be true. Indeed, in this setting, the region R_{IB} corresponds to C "minus" the convex hull of P, modulo some boundary points. This is exhibited in Figure 2.3. In the next subsection, we will formalize these observations and show under what conditions they hold true.

2.3.3 Geometry of R_{IB} under Monotonicity in Payoffs

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

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

Definition 2.3.3. The system with expected payoff vectors $P = \{\mathbf{p}(\mathbf{d}) | \mathbf{d} \in$

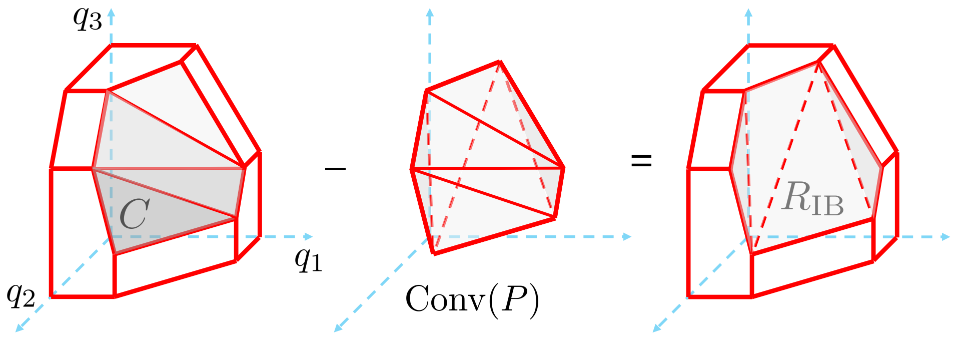

Figure 2.3: Visualizing R_{IB} for the three-user scenario in Figure 2.2. In this example, $R_{\mathrm{IB}} = \mathrm{cl}(C - \mathrm{Conv}(P))$.

$D\}$ is said to satisfy **monotonicity in individual expected payoffs** if, for any two priority decisions $\mathbf{d}_1$ and $\mathbf{d}_2$ and any user i such that $S_i(\mathbf{d}_1) \subseteq S_i(\mathbf{d}_2)$, we have that $p_i(\mathbf{d}_1) \geq p_i(\mathbf{d}_2)$. We call this **monotonicity in payoffs** for short.

In other words, a user i can expect to get a higher payoff if some users that have higher priority than it are re-assigned to have lower priorities. Note that we are not comparing the expected payoffs of different users since payoffs can be defined in different ways for different users and may not be comparable. This property characterizes in a broad sense how priorities impact the expected payoffs when the underlying system allocates resources. It is a natural condition but need not hold in general. For example, by the property of monotonicity in payoffs, if $S_i(\mathbf{d}_1) = S_i(\mathbf{d}_2)$, then $p_i(\mathbf{d}_1) = p_i(\mathbf{d}_2)$. In other words, re-ordering users with higher priority than user i does not impact its payoff, which may not hold in general.

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

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

We call B the *dominant of the convex hull*. Contrast this to the definition of C in (2.2).

For the special cases in Figure 2.2 and 2.3, $B \cap C$ equals to $\mathrm{Conv}(P)$, but in general it can be larger than $\mathrm{Conv}(P)$. Figure 2.4 shows a conceptual picture of what could happen. The three circles represent three possible expected payoff vectors. Here, the whole shadowed area $B \cap C$ is larger than the region $\mathrm{Conv}(P)$ which is the triangle formed by the three circles. Note that this is only a conceptual example to help visualize $B \cap C$ in higher dimensions. In reality for two dimensions, i.e., systems with two users, we know there are only 2 expected payoff vectors as shown in Figure 2.2.

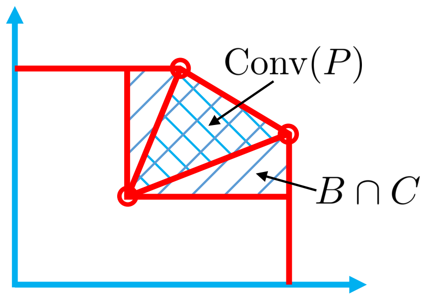

Figure 2.4: An example where $B \cap C$ is larger than $\mathrm{Conv}(P)$.

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

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

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

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

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

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

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

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

Definition 2.3.4. Let R be defined as follows:

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

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

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

Theorem 2.3.4. *If the system satisfies monotonicity in payoffs, then the inner bound region R_{IB} is such that*

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

See Appendix 2.8.4 for this somewhat intricate argument.

2.3.4 Sufficient Condition for w-LDF's Optimality

By Theorem 2.3.3 and Theorem 2.3.4, we immediately get

$$\text{int}(R) \subseteq \text{int}(R_{\text{IB}}) \subseteq F_{\mathbf{w}\text{-LDF}}. \tag{2.5}$$

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

Definition 2.3.5. The system with expected payoff vectors $P = \{\mathbf{p}(\mathbf{d}) | \mathbf{d} \in D\}$ is said to satisfy **subset payoff equivalence** if for each subset of users S

the vectors in $P^S = \{\mathbf{p}^S(\mathbf{d}) | \mathbf{d} \in D(S)\}$ lie on a hyperplane, i.e., there exists a nonzero $\boldsymbol{\alpha}^S \succeq \mathbf{0}$ such that for all $\mathbf{d}_1, \mathbf{d}_2 \in D(S)$,

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

Theorem 2.3.5. *If the system satisfies monotonicity in payoffs and subset payoff equivalence, then*

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

and therefore, the $\mathbf{w}$*-LDF policies are feasibility optimal.*

Please refer to Appendix 2.8.7 for detailed proof.

The conditions for this theorem are akin but not equivalent to the conditions introduced in [19] for the generalized switch model. Specifically, we require the system to satisfy monotonicity in payoffs and subset payoff equivalence. The work in [19] requires local pooling in the generalized switch model. In the model in [19], given a priority decision $\mathbf{d} = (d_1, d_2, \cdots, d_n)$ where d_i is the index of the queue with the the i^{th} highest priority, the queue service rate vector can be denoted by $\mathbf{m}(\mathbf{d})$, where $m_{d_i}(\mathbf{d})$ represents the units of work that can be removed from queue d_i in one time slot under priority decision $\mathbf{d}$. $\mathbf{m}(\mathbf{d})$ is akin to $\mathbf{p}(\mathbf{d})$ in our context. However, the generalized switch model in [19] implies properties on the service rate vectors $\mathbf{m}(\mathbf{d})$. For example, it implies that for all $\mathbf{d} = (d_1, d_2, \cdots, d_n)$, we have $m_{d_1}(\mathbf{d}) \geq m_{d_1}(\mathbf{d}')$ for all $\mathbf{d}'$, and $m_{d_2}(\mathbf{d}) \geq m_{d_2}(\mathbf{d}')$ for all $\mathbf{d}'$ satisfying $m_{d_1}(\mathbf{d}) = m_{d_1}(\mathbf{d}')$, etc.

These implicit requirements do not necessarily hold in systems which satisfy the conditions in Theorem 2.3.5.

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

Corollary 2.3.6. *If the system has two users and satisfies monotonicity in payoffs, then* **w***-LDF policies are feasibility optimal.*

Note that in a two-user scenario, the property of monotonicity in payoffs simply means a user gets higher payoff under the higher priority than its payoff under the lower priority. In Section 2.4.2 we will consider hierarchical systems serving two classes of exchangeable users and use this corollary to show the optimality of LDF-like policies.

2.3.5 Efficiency Ratio Analysis

When the optimality conditions in Theorem 2.3.5 do not hold, one can still study the efficiency ratio, see e.g., [36], to evaluate the performance of **w**-LDF policies.

Definition 2.3.6. The **efficiency ratio** of the **w**-LDF policy is defined as

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

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

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

Definition 2.3.7. Given a subset of users $S \subseteq N$, the **subset payoff ratio** σ_S for S is defined as

$$\sigma_S = \max_{\substack{\boldsymbol{\alpha}^S \succeq \mathbf{0} \\ \boldsymbol{\alpha}^S \neq \mathbf{0}}} \frac{\min_{\mathbf{d} \in D(S)} \langle \boldsymbol{\alpha}^S, \mathbf{p}^S(\mathbf{d}) \rangle}{\max_{\mathbf{d} \in D(S)} \langle \boldsymbol{\alpha}^S, \mathbf{p}^S(\mathbf{d}) \rangle}. \tag{2.6}$$

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

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

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

Theorem 2.3.7. *If the system satisfies monotonicity in payoffs, the efficiency ratio of the* $\mathbf{w}$*-LDF policy is such that*

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

See Appendix 2.8.8 for the proof. Intuitively, the bottleneck of the efficiency ratio is the subset S where σ_S is the smallest.

Note that by picking any $\boldsymbol{\alpha} \succ \mathbf{0}$, we can get lower bounds on σ_S for all subsets $S \subseteq N$ by placing its projection $\boldsymbol{\alpha}^S$ into (2.6). Thus, any $\boldsymbol{\alpha} \succ \mathbf{0}$ enables us to construct a lower bound on $\gamma_{\mathbf{w}\text{-LDF}}$. A trivial option is $\boldsymbol{\alpha} = \mathbf{1}$, where for each subset S the value of $\langle \mathbf{1}^S, \mathbf{p}^S(\mathbf{d})\rangle$ represents the sum payoff of users in S under decision $\mathbf{d}$.

We have shown that the efficiency and optimality of the $\mathbf{w}$-LDF policies is related to R_{IB}. Understanding and analyzing the geometry of R_{IB} can in principle enable us to provide feedback to the designers of priority-based resource allocation mechanisms regarding which specific priority decision or set of priority decisions are problematic and bottlenecks for the system so that the designers can focus on improving the resource allocation for these problematic decisions. For example, in the conceptual setting shown in Figure 2.4, the priority decision corresponding to the lower left circle is the “bottleneck” of the system and should be targeted to make the dominant of the convex hull as small as possible. This is of particular interest for some practical systems where it is possible to get explicit knowledge of P which reflect the underlying priority-based resource allocation, e.g., by collecting data over a long time.

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

2.4 Examples for w-LDF's Optimality

Theorem 2.3.5 gives a sufficient condition for **w**-LDF to be feasibility optimal. One example system that satisfies these conditions is the model considered in prior work [32] which, as mentioned in Section 2.1, can be viewed as a single-resource geometric-workload model. In this section we consider more system settings and show how our results provide useful and practical insights.

2.4.1 Exchangeable Expected Payoffs

We shall start by showing that for systems that are "symmetric", **w**-LDF policies are feasibility optimal.

Definition 2.4.1. A subset of users S is said to have **exchangeable expected payoffs** if, for all priority decisions $\mathbf{d} \in D$ and all $i, j \in S$, if we switch the priorities of user i and j and use $\mathbf{d}'$ to represent the resulting new priority decision, then

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

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

If the users in N have exchangeable expected payoffs, we can verify the property of subset payoff equivalence by picking $\boldsymbol{\alpha}^S = \mathbf{1}^S$ for each subset of users S. Therefore, by Theorem 2.3.5 we get the following corollary.

Corollary 2.4.1. *If the set of users N have exchangeable expected payoffs and the system satisfies monotonicity in payoffs, then the* $\mathbf{w}$*-LDF policies are feasibility optimal.*

See Appendix 2.8.9 for the proof.

2.4.2 Multiple Classes of Exchangeable Users and Hierarchical-LDF

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

In practice, depending on whether the priorities of classes can change dynamically, there are two types of class-based hierarchical prioritization:

Type 1 where the class priorities are fixed, and Type 2 where one can dynamically prioritize classes of users, and then users within each class.

The first type of hierarchical prioritization might correspond to a setting where the users/applications are separated into human-interactive/high-QoS and background-processing/low-QoS categories [29], and it is always desirable to first process high-QoS users. In this setting, the problem is reduced to a collection of independent user prioritization problems similar to the one considered in this chapter. By Corollary 2.4.1, $\mathbf{w}$-LDF is feasibility optimal to prioritize users in each class.

The second type of dynamic hierarchical prioritization might be of interest in systems where switching between processing different user classes involves overheads, and/or where it is inefficient to mix the processing of different user classes, probably because of resource contention or deadlocks.

In this setting, we propose a class-based *hierarchical-LDF* policy that in each period works in two steps by (1) prioritizing classes by LDF based on the aggregate deficits, i.e., the sum of deficits for users in the same class, and (2) prioritizing users in each class according to LDF based on individual users' deficits. The framework of hierarchical-LDF is exhibited in Figure 2.5. Note that here LDF can be replaced by $\mathbf{w}$-LDF for any $\mathbf{w} \succ \mathbf{0}$ and the following result would hold.

Theorem 2.4.2. *In a system with two classes of exchangeable users, if the property of monotonicity in payoffs is satisfied, the hierarchical-LDF policy*

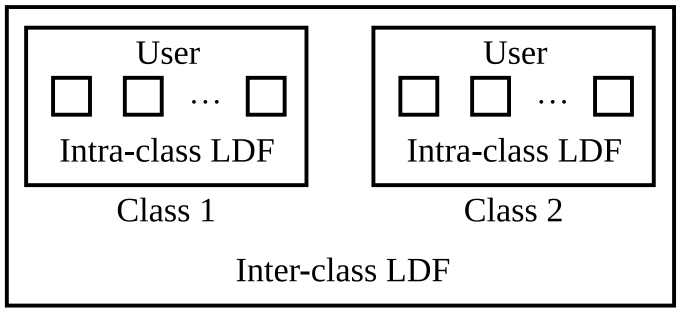

Figure 2.5: The framework for class-based hierarchical-LDF policy.

is feasibility optimal among all possible class-based hierarchical prioritization policies.

The proof follows directly from Corollary 2.3.6 and 2.4.1. By Corollary 2.3.6 we know the class-based LDF policy is optimal to set priorities amongst the two classes and by Corollary 2.4.1 we know the LDF-based user prioritization is also optimal for the exchangeable users in each class.

More generally, for systems serving multiple (more than two) classes of exchangeable users, one can view each class as a "super user", and define the aggregate payoff and QoS requirement for a super user to be the sum of payoffs and QoS requirements for users in that class, respectively. Then the dynamic prioritization of super users can be viewed as the problem considered in this chapter. Therefore, by Theorem 2.3.5, if the system with the super users' expected aggregate payoffs satisfies monotonicity in payoffs and subset payoff equivalence, the LDF policy is a feasibility optimal choice for prioritizing super users and thus, the hierarchical-LDF policy is feasibility optimal among all class-based hierarchical prioritization policies. Indeed, all the results we have introduced, e.g., Theorem 2.3.2-2.3.7, still hold for the prioritization of these super users.

2.5 Some Practical Issues

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

However, the non-negative definition of deficit (2.1) makes it hard to track excess payoffs. For example, consider a model with 2 users and suppose the payoff is always 1 for the high priority user and 0 for the low priority user. Suppose the payoff requirement vector is $\mathbf{q} = (0.1, 0.5)$. Since $1 > 0.1 + 0.5$, we know $\mathbf{q}$ is feasible and the system can deliver 0.4 excess payoff. Suppose we use the LDF policy, starting from $\mathbf{X}(0) = (0, 0)$ it is easy to verify[3] that the system will switch giving high priority to these two users, and thus the achieved average payoff vector is $\mathbf{p} = (0.5, 0.5)$. Clearly User 1 gets 0.4 excess payoff while User 2 gets nothing. This happens because $X_1(t)$ and $X_2(t)$

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

are frequently forced to 0 from different negative values, which causes the "unfairness" between these two users.

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

$$X'_i(t+1) = X'_i(t) + q_i - V_i(\mathbf{d}(t+1)), \tag{2.7}$$

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

Now for the simple example above, if we adopt LDF but based on the possibly negative deficits $\mathbf{X}'(t) = (X'_1(t), X'_2(t), \cdots, X'_n(t))$, we can get achieved average payoff vector $\mathbf{p} = (0.3, 0.7)$. We observe that the two users equally split the excess payoff.

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

Note that for completeness we will need to modify the feasibility definition since the process $\{\mathbf{X}'(t)\}_{t\geq 1}$ is no longer positive recurrent as it may keep decreasing or increasing. Now we call a payoff requirement vector $\mathbf{q}$ feasible if, under some user prioritization policy, for each user i the time-averaged payoff per period is at least q_i. Formally, recall that $Y_i(\mathbf{d}(t))$ is the random payoff for user i in period t. As the payoff requirement, each user i requires that

$$\liminf_{\tau\to\infty} \frac{1}{\tau}\sum_{t=1}^{\tau} Y_i(\mathbf{d}(t)) \geq q_i, \text{with probability } 1.$$

Note that this definition and Definition 2.2.2 are just two technical ways to define the feasibility. With the theorem in [8] we can actually show that for any user prioritization policy, the sets of feasible QoS requirements under these two different feasibility definitions differ by at most a boundary and thus, are equivalent for practical purposes. Therefore, all the results we discuss in this chapter hold under both feasibility definitions.

2.6 Simulations

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

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

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

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

2.6.1 Impact of Weights on Long-Term Completion Ratios

In Table 2.2 we consider a requirement vector $\mathbf{q}$ that is feasible under the $\mathbf{w}$-LDF policies and display the achieved $\mathbf{p}$ under two different weight vectors $\mathbf{w}$. For each weight vector $\mathbf{w}$, we verify that $w_i(p_i - q_i)$ is the same for all three users. Contrasting the two lines in Table 2.2, we can see that for a system which can deliver more than required, changing the weight vector reallocates the excess payoffs and gives more excess payoff to users with smaller weights.

Table 2.2: Achieved completion ratio vectors under two weight vectors.

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

2.6.2 Characterization of Clustering of Failures and Impact of Weights

If a user's task is not completed in a period, we call it a failure event. The requirement vector $\mathbf{q}$ focuses on long-term task completion ratio, but it would likely be undesirable for a user to experience consecutive or clustered failure events. Figure2.6 gives an example of failure events. In this subsection we consider the same $\mathbf{q} = (0.8, 0.6, 0.4)$ used above and explore the clustering of failures under two $\mathbf{w}$-LDF policies.

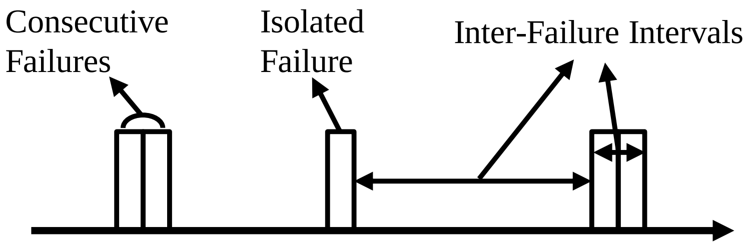

Figure 2.6: Characteristics of clustering of failures.

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

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

Under some **w**-LDF policy, we define *SD ratio* of user i to be the ratio of user i's IFI SD to the SD of the geometric random variable with parameter $1 - p_i$. Table 2.3 shows the SD ratios of three users under two different weight vectors $\mathbf{w}$. Under $\mathbf{w} = (1, 1, 1)$, the ratios are less than 1, indicating that the failures under the LDF policy are less clustered compared to the scheme where failure event happens i.i.d. in each period. The last two columns in Table 2.3 indicates that increasing the weight of user i reduces the degree of failure clustering for user i but at the price of other users' more clustered failures. Thus, the users' sensitivities to clustered failures is another factor to consider when one assigns weights to users.

Table 2.3: Characterization of clustering of failures.

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

2.7 Conclusion

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

2.8 Appendix

2.8.1 Proof of Lemma 2.3.1

Given $\mathbf{q} \in F$, since it is feasible there exists a user prioritization policy η that fulfills $\mathbf{q}$, i.e., the Markov chain $\{\mathbf{X}(t)\}_{t\geq 1}$ is positive recurrent, which implies there exists a stationary distribution over the state space. Since η is a stationary policy that picks $\mathbf{d}(t+1)$ based on $\mathbf{X}(t)$, by Ergodic Theorem each priority decision $\mathbf{d}$ is selected with some time fraction $\alpha_{\mathbf{d}}$ such that $\sum\limits_{\mathbf{d}\in D} \alpha_{\mathbf{d}} = 1$. If we consider $X_i(t)$ as a queue, the average arrival q_i should not be bigger than the average departure which is given by $\sum\limits_{\mathbf{d}\in D} \alpha_{\mathbf{d}} p_i(\mathbf{d})$ since otherwise $X_i(t)$ goes a.s. to infinity and the chain cannot be positive recurrent.

Therefore,

$$\mathbf{q} \preceq \sum_{\mathbf{d}\in D} \alpha_{\mathbf{d}} \mathbf{p}(\mathbf{d}) \in \mathrm{Conv}(P),$$

which implies $\mathbf{q} \in C \subseteq \mathrm{cl}(C)$.

2.8.2 Proof of Theorem 2.3.2

We start by introducing a lemma.

Lemma 2.8.1. *A payoff requirement vector* $\mathbf{q}$ *is in* C *if and only if, for any* *non-negative vector $\boldsymbol{\gamma} \succeq 0$, there exists a priority decision $\mathbf{d}$, such that*

$$\langle \boldsymbol{\gamma}, \mathbf{q} \rangle \leq \langle \boldsymbol{\gamma}, \mathbf{p}(\mathbf{d}) \rangle.$$

Equivalently, $\mathbf{q}$ is in C if and only if, for any $\boldsymbol{\gamma} \succeq \mathbf{0}$,

$$\langle \boldsymbol{\gamma}, \mathbf{q} \rangle \leq \max_{\mathbf{d} \in D} \langle \boldsymbol{\gamma}, \mathbf{p}(\mathbf{d}) \rangle.$$

To understand this lemma, since any vector in C is dominated by some $\mathbf{q} \in \mathrm{Conv}(P)$, we consider for simplicity a $\mathbf{q} \in \mathrm{Conv}(P)$. Such a vector can be expressed as a convex combination of expected payoff vectors, i.e., $\mathbf{q} = \sum_{\mathbf{d} \in D} \alpha_{\mathbf{d}} \mathbf{p}(\mathbf{d})$, where $\sum_{\mathbf{d} \in D} \alpha_{\mathbf{d}} = 1$ and $\alpha_{\mathbf{d}} \geq 0, \forall \mathbf{d} \in D$.

Now we have

$$\begin{aligned}
\langle \boldsymbol{\gamma}, \mathbf{q} \rangle &= \sum_{\mathbf{d} \in D} \alpha_{\mathbf{d}} \langle \boldsymbol{\gamma}, \mathbf{p}(\mathbf{d}) \rangle \\
&\leq \sum_{\mathbf{d} \in D} \alpha_{\mathbf{d}} \max_{\mathbf{d} \in D} \langle \boldsymbol{\gamma}, \mathbf{p}(\mathbf{d}) \rangle \\
&= \max_{\mathbf{d} \in D} \langle \boldsymbol{\gamma}, \mathbf{p}(\mathbf{d}) \rangle.
\end{aligned}$$

This actually proves the necessity of the condition. The formal proof is shown below.

Proof of Lemma 2.8.1. Given a payoff requirement vector $\mathbf{q}$, by definition of C, we know that $\mathbf{q}$ lying in set C is equivalent to the *feasibility* of the following set of linear equations and inequalities,

$$\begin{cases} \mathbf{q} \preceq \sum\limits_{\mathbf{d}\in D} \alpha_{\mathbf{d}} \mathbf{p}(\mathbf{d}) \\ \alpha_{\mathbf{d}} \geq 0, \forall \mathbf{d} \in D \\ \sum\limits_{\mathbf{d}\in D} \alpha_{\mathbf{d}} = 1. \end{cases} \tag{2.8}$$

The condition in Lemma 2.8.1 is equivalent to the *infeasibility* of

$$\begin{cases} \boldsymbol{\gamma} \succeq \mathbf{0} \\ \langle \boldsymbol{\gamma}, \mathbf{q} \rangle > \langle \boldsymbol{\gamma}, \mathbf{p}(\mathbf{d}) \rangle, \forall \mathbf{d} \in D. \end{cases} \tag{2.9}$$

By strong duality [10] it is easy to prove that set (2.8) being feasible is equivalent to set (2.9) being infeasible and this concludes the proof. □

Proof of Theorem 2.3.2. By Lemma 2.3.1 we know $F \subseteq \mathrm{cl}(C)$ and by definition we know $F_{\mathrm{MW}} \subseteq F$. To prove the theorem it suffices to show $\mathrm{int}(C) \subseteq F_{\mathrm{MW}}$. We show this by constructing a Lyapunov function and using Foster's theorem.

Given $\mathbf{q} \in \mathrm{int}(C)$, the goal is to show $\mathbf{q}$ can be fulfilled by the MW policy.

By definition of interior there exists $\epsilon > 0$ such that $\mathbf{q}' = \mathbf{q} + \epsilon\mathbf{1} \in C$ where $\mathbf{1} = (1, 1, \cdots, 1)$. We define a Lyapunov function as

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

In period $t+1$, we have

$$\begin{aligned}
&\mathrm{E}\left[L(\mathbf{X}(t+1)) - L(\mathbf{X}(t)) | \mathbf{X}(t) = \mathbf{x}\right] \\
&= \mathrm{E}\left[\sum_{i=1}^{n} X_i(t+1)^2 - X_i(t)^2 | \mathbf{X}(t) = \mathbf{x}\right] \\
&\leq \mathrm{E}\left[\sum_{i=1}^{n} (X_i(t) + q_i - V_i(\mathbf{d}(t+1)))^2 \right. \\
&\qquad \left. - X_i(t)^2 | \mathbf{X}(t) = \mathbf{x}\right] \\
&= \mathrm{E}\left[\sum_{i=1}^{n} (q_i - V_i(\mathbf{d}(t+1)))^2 \right. \\
&\qquad \left. + 2\langle \mathbf{X}(t), \mathbf{q} - \mathbf{V}(\mathbf{d}(t+1))\rangle | \mathbf{X}(t) = \mathbf{x}\right] \\
&\leq \mathrm{E}\left[\sum_{i=1}^{n} (q_i^2 + V_i(\mathbf{d}(t+1))^2) \right. \\
&\qquad \left. + 2\langle \mathbf{X}(t), \mathbf{q} - \mathbf{V}(\mathbf{d}(t+1))\rangle | \mathbf{X}(t) = \mathbf{x}\right]
\end{aligned} \tag{2.10}$$

Under the MW policy, by (2.3) we know that

$$\begin{aligned}
&\mathrm{E}[\langle \mathbf{X}(t), \mathbf{V}(\mathbf{d}(t+1))\rangle | \mathbf{X}(t) = \mathbf{x}] \\
&= \mathrm{E}[\langle \mathbf{X}(t), \mathbf{p}(\mathbf{d}(t+1))\rangle | \mathbf{X}(t) = \mathbf{x}] \\
&= \max_{\mathbf{d} \in D} \langle \mathbf{x}, \mathbf{p}(\mathbf{d})\rangle.
\end{aligned}$$

Since $\mathbf{q}' = \mathbf{q} + \epsilon\mathbf{1} \in C$, we get that

$$\begin{aligned}
&\mathrm{E}[\langle \mathbf{X}(t), \mathbf{q} - \mathbf{V}(\mathbf{d}(t+1))\rangle | \mathbf{X}(t) = \mathbf{x}] \\
&\quad = \mathrm{E}[\langle \mathbf{X}(t), \mathbf{q}' - \mathbf{V}(\mathbf{d}(t+1))\rangle | \mathbf{X}(t) = \mathbf{x}] - \epsilon\langle \mathbf{x}, \mathbf{1}\rangle \\
&\quad = \langle \mathbf{x}, \mathbf{q}'\rangle - \max_{\mathbf{d}\in D}\langle \mathbf{x}, \mathbf{p}(\mathbf{d})\rangle - \epsilon\langle \mathbf{x}, \mathbf{1}\rangle \\
&\quad \leq -\epsilon\langle \mathbf{x}, \mathbf{1}\rangle,
\end{aligned}$$

where the last step is true by Lemma 2.8.1.

Also since there are finite payoff vectors, we use b_1 to represent an upper bound on all possible payoff values and payoff requirements. Therefore, by (2.10) we get that

$$\begin{aligned}
\mathrm{E}[L(\mathbf{X}(t+1)) - L(\mathbf{X}(t)) | \mathbf{X}(t) = \mathbf{x}] &\leq 2nb_1^2 - 2\epsilon\langle \mathbf{x}, \mathbf{1}\rangle \\
&\leq -1
\end{aligned}$$

for $\mathbf{x}$ satisfying $\langle \mathbf{x}, \mathbf{1}\rangle \geq \frac{nb_1^2}{\epsilon} + \frac{1}{2\epsilon}$.

It is not hard to show[4] there are finite states $\mathbf{x}$ with $\langle \mathbf{x}, \mathbf{1}\rangle < \frac{nb_1^2}{\epsilon} + \frac{1}{2\epsilon}$. Therefore, by Foster's theorem, $\{\mathbf{X}(t)\}_{t\geq 1}$ is positive recurrent and $\mathbf{q}$ is fulfilled by the MW policy. Thus, this shows that $\mathrm{int}(C) \subseteq F_{\mathrm{MW}}$. □

2.8.3 Proof of Theorem 2.3.3

We first introduce some further 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 $\boldsymbol{a}\circ\mathbf{b} = (a_1b_1, a_2b_2, \cdots, a_nb_n)$

[4]This is true because given our assumption that requirement $\mathbf{q}$ and the payoff vectors are rational valued and have finite options, the state space of process $\{\mathbf{X}(t)\}_{t\geq 1}$ is in a lattice, see e.g., [15].

the entrywise product.

For any $\mathbf{w} \succ \mathbf{0}$ and $\mathbf{q} \in \text{int}(R_{\text{IB}})$, the goal is to show that $\mathbf{q}$ can be fulfilled by the $\mathbf{w}$-LDF policy.

Let $\boldsymbol{\beta} = (\frac{1}{w_1}, \frac{1}{w_2}, \cdots, \frac{1}{w_n})$. By definition of interior there exists an $\epsilon > 0$ such that $\mathbf{q}' = \mathbf{q} + \epsilon\boldsymbol{\beta} \in R_{\text{IB}}$. By definition of R_{IB}, there exists a vector $\boldsymbol{\alpha} \succ \mathbf{0}$ such that $\mathbf{q}'$ and $\boldsymbol{\alpha}$ satisfy the conditions (2.4).

Consider the following candidate Lyapunov function:

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

Note that we consider a process $\{\mathbf{X}(t)\}_{t\geq 1}$ that is driven by the $\mathbf{w}$-LDF policy. In period $t+1$, by similar analysis as in (2.10) we have that

$$\begin{aligned}
&\text{E}\left[L(\mathbf{X}(t+1)) - L(\mathbf{X}(t)) | \mathbf{X}(t) = \mathbf{x}\right] \\
&\leq \text{E}\Bigg[\sum_{i=1}^{n} \alpha_i w_i (q_i^2 + V_i(\mathbf{d}(t+1))^2) \\
&\quad +2\langle \boldsymbol{\alpha} \circ \mathbf{w} \circ \mathbf{X}(t), \mathbf{q} - \mathbf{V}(\mathbf{d}(t+1))\rangle | \mathbf{X}(t) = \mathbf{x}\Bigg] \\
&= \text{E}\Bigg[\sum_{i=1}^{n} \alpha_i w_i (q_i^2 + V_i(\mathbf{d}(t+1))^2) \\
&\quad +2\langle \boldsymbol{\alpha} \circ \mathbf{w} \circ \mathbf{X}(t), \mathbf{q}' - \mathbf{V}(\mathbf{d}(t+1))\rangle | \mathbf{X}(t) = \mathbf{x}\Bigg] \\
&\quad -2\epsilon\langle \mathbf{x}, \boldsymbol{\alpha}\rangle
\end{aligned} \tag{2.11}$$

Let $\mathbf{d}$ denote the priority decision selected according to $\mathbf{w}$-LDF policy.

Thus, We have that

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

By reordering users according to priorities, we get

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

By **w**-LDF policy we know $w_{d_i} x_{d_i} \geq w_{d_{i+1}} x_{d_{i+1}}$. By (2.4) 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,

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

Suppose b_2 is an upper bound on all α_i and w_i, by (2.11), we get that

$$\begin{aligned}\mathrm{E}[L(\mathbf{X}(t+1)) - L(\mathbf{X}(t)) | \mathbf{X}(t) = \mathbf{x}] &\leq 2nb_2^2 b_1^2 - 2\epsilon \langle \mathbf{x}, \boldsymbol{\alpha}\rangle \\ &\leq -1\end{aligned}$$

for $\mathbf{x}$ satisfying $\langle \mathbf{x}, \boldsymbol{\alpha}\rangle \geq \frac{nb_2^2 b_1^2}{\epsilon} + \frac{1}{2\epsilon}$.

Again, since there are finite states $\mathbf{x}$ with $\langle \mathbf{x}, \boldsymbol{\alpha}\rangle < \frac{nb_2^2 b_1^2}{\epsilon} + \frac{1}{2\epsilon}$, by Foster's Theorem $\{\mathbf{X}(t)\}_{t\geq 1}$ is positive recurrent and $\mathbf{q}$ is fulfilled by the **w**-LDF policy.

Therefore, for any $\mathbf{w} \succ \mathbf{0}$, we have that

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

2.8.4 Proof of Theorem 2.3.4

The proof of Theorem 2.3.4 is complicated and we apologize for that. Part of the complication comes from understanding how the property of monotonicity in payoffs characterizes the geometry of the region R_{IB}. Further, a feasible payoff requirement implies feasibilities for all user subsets, and thus we need to look at the projections in all subspaces.

Figure 2.7 gives the high-level outline for the proof of Theorem 2.3.4. There are two parts which involve the technical results Lemma 2.8.2 and 2.8.3, which we will state in the proof. In order to allow the reader follow the proof, we defer their own proof to later.

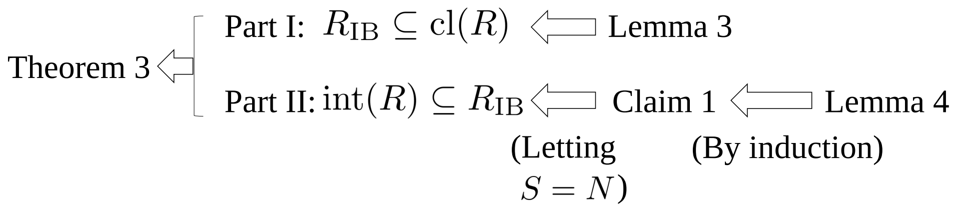
Theorem 3
Part I: $R_{\mathrm{IB}} \subseteq \mathrm{cl}(R)$
Lemma 3
Part II: $\mathrm{int}(R) \subseteq R_{\mathrm{IB}}$
Claim 1
Lemma 4
(Letting $S = N$)
(By induction)

Figure 2.7: Outline for the proof of Theorem 2.3.4.

Lemma 2.8.2. *If a system satisfies monotonicity in payoffs, then for all* $\mathbf{q} \in C$ *and all subsets of users* $S \subseteq N$, $\mathbf{q}^S \in C^S$.

We have argued in Section 2.3.3 that C^S does not necessarily equal to the projection of region C on the subspace of S in general, but the statement is true if the system satisfies monotonicity in payoffs. Please refer to Appendix 2.8.5 for detailed proof of this lemma.

Given a subset of users $S \subseteq N$, for two vectors $\mathbf{p}^S$ and $\mathbf{q}^S$, we say $\mathbf{p}^S \succ_S \mathbf{q}^S$ if $p_i^S > q_i^S$ for any $i \in S$. We define the subspace of S as $\mathbb{R}_+^S = \{\mathbf{q} \in \mathbb{R}_+^n | q_i = 0, \forall i \notin S\}$.

For a region $X \subseteq \mathbb{R}_+^n$ which lies in the subspace of S, we denote by $\mathrm{int}^S(X)$ the relative interior of X, i.e., the interior of X, relative to the subspace of S. Similarly we use $\mathrm{cl}^S(X)$, $\mathrm{bd}^S(X)$ to represent the relative closure and boundary of X in the subspace of S, respectively.

Following the definition of region R in Definition 2.3.4, we can express $\mathrm{int}(R)$ and $\mathrm{cl}(R)$ as below,

$$\mathrm{int}(R) = \{\mathbf{q} | \forall S \subseteq \{1, 2, \cdots, n\}, \mathbf{q}^S \in \mathrm{int}^S(C^S \setminus B^S)\}, \tag{2.12}$$

$$\mathrm{cl}(R) = \{\mathbf{q} | \forall S \subseteq \{1, 2, \cdots, n\}, \mathbf{q}^S \in \mathrm{cl}^S(C^S \setminus B^S)\}. \tag{2.13}$$

By definition, we know C^S and B^S are closed sets. We can also show[5]

$$\mathrm{int}^S(C^S \setminus B^S) = \mathrm{int}^S(C^S) \setminus B^S,$$
$$\mathrm{cl}^S(C^S \setminus B^S) = C^S \setminus \mathrm{int}^S(B^S).$$

Next we prove Theorem 2.3.4 in two parts: $R_{\mathrm{IB}} \subseteq \mathrm{cl}(R)$ and $\mathrm{int}(R) \subseteq R_{\mathrm{IB}}$.

Part I of the proof:

[5]This is clear from the definition of C^S and B^S. We omit the proof to save space.

We start with the easy part and first show $R_{\mathrm{IB}} \subseteq \mathrm{cl}(R)$.

Given $\mathbf{q} \in R_{\mathrm{IB}}$, by definition, there exists $\boldsymbol{\alpha} \succ \mathbf{0}$ such that for all subsets of users S,

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

i.e.,

$$\langle \boldsymbol{\alpha}^S, \mathbf{q}^S \rangle \leq \min_{\mathbf{d} \in D(S)} \langle \boldsymbol{\alpha}^S, \mathbf{p}^S(\mathbf{d}) \rangle. \tag{2.14}$$

To show $\mathbf{q} \in \mathrm{cl}(R)$, by (2.13) we need to show for all subsets of users S that $\mathbf{q}^S \in C^S \setminus \mathrm{int}^S(B^S)$.

Since $\mathbf{q} \in R_{\mathrm{IB}} \subseteq C$, by Lemma 2.8.2 we have $\mathbf{q}^S \in C^S$.

Now suppose for some subset of users S, $\mathbf{q}^S \in \mathrm{int}^S(B^S)$. By definition of B^S and interior, there exists $\mathbf{x}^S \in \mathrm{Conv}(P^S)$ such that $\mathbf{q}^S \succeq \mathbf{x}^S$ and $\mathbf{q}^S \neq \mathbf{x}^S$, which implies that

$$\langle \boldsymbol{\alpha}^S, \mathbf{q}^S \rangle > \langle \boldsymbol{\alpha}^S, \mathbf{x}^S \rangle.$$

Since $\mathbf{x}^S \in \mathrm{Conv}(P^S)$, there exists $\{c_{\mathbf{d}} | \mathbf{d} \in D(S)\}$ such that $\mathbf{x}^S = \sum_{\mathbf{d} \in D(S)} c_{\mathbf{d}} \mathbf{p}^S(\mathbf{d})$ and $c_{\mathbf{d}} \geq 0$, $\sum_{\mathbf{d} \in D(S)} c_{\mathbf{d}} = 1$. Therefore,

$$\begin{aligned} \langle \boldsymbol{\alpha}^S, \mathbf{q}^S \rangle > \langle \boldsymbol{\alpha}^S, \mathbf{x}^S \rangle &= \sum_{\mathbf{d} \in D(S)} c_{\mathbf{d}} \langle \boldsymbol{\alpha}^S, \mathbf{p}^S(\mathbf{d}) \rangle \\ &\geq \min_{\mathbf{d} \in D(S)} \langle \boldsymbol{\alpha}^S, \mathbf{p}^S(\mathbf{d}) \rangle, \end{aligned} \tag{2.15}$$

which is contradicted to (2.14). Therefore, $\mathbf{q}^S \notin \mathrm{int}^S(B^S)$ and thus $\mathbf{q} \in \mathrm{cl}(R)$, implying that $R_{\mathrm{IB}} \subseteq \mathrm{cl}(R)$.

Part II of the proof:

For the second part we show $\mathrm{int}(R) \subseteq R_{\mathrm{IB}}$.

Given $\mathbf{q} \in \mathrm{int}(R)$, by (2.12), we know for all S, $\mathbf{q}^S \in \mathrm{int}^S(C^S) \setminus B^S$. The goal is to show $\mathbf{q} \in R_{\mathrm{IB}}$, i.e., to find an $\boldsymbol{\alpha} \succ \mathbf{0}$ such that for all subsets of users S,

$$\langle \boldsymbol{\alpha}^S, \mathbf{q}^S \rangle \leq \min_{\mathbf{d} \in D(S)} \langle \boldsymbol{\alpha}^S, \mathbf{p}^S(\mathbf{d}) \rangle. \tag{2.16}$$

We start with the following lemma which is proved in Appendix 2.8.6 in the sequel.

Lemma 2.8.3. *If a system satisfies monotonicity in payoffs, given $\mathbf{q} \in$ int(R), for each subset of users S, there exists nonzero $\boldsymbol{\beta}^S \succeq \mathbf{0}$, such that for all $S' \subseteq S$ where $\boldsymbol{\beta}^{S'} \neq \mathbf{0}$,*

$$\langle \boldsymbol{\beta}^{S'}, \mathbf{q}^{S'} \rangle < \min_{\mathbf{d} \in D(S')} \langle \boldsymbol{\beta}^{S'}, \mathbf{p}^{S'}(\mathbf{d}) \rangle. \tag{2.17}$$

Given Lemma 2.8.3, by letting $S = N$ we find $\boldsymbol{\beta}$ that is very similar to the $\boldsymbol{\alpha}$ we are looking for, except for two differences: (1) $\boldsymbol{\beta}$ may not be strictly positive, and (2) it is strictly "less than" in (2.17). The idea is to add a small perturbation to $\boldsymbol{\beta}$ to construct a strictly positive vector.

Formally, given Lemma 2.8.3, to show $\mathbf{q} \in R_{\mathrm{IB}}$ we shall prove the following even stronger statement by induction.

Claim 1: If a system satisfies monotonicity in payoffs, given $\mathbf{q} \in \mathrm{int}(R)$, for each subset of users S, there exists $\boldsymbol{\alpha}^S \succ_S \mathbf{0}$ such that for all

$S' \subseteq S$,

$$\langle \boldsymbol{\alpha}^{S'}, \mathbf{q}^{S'} \rangle \leq \min_{\mathbf{d} \in D(S')} \langle \boldsymbol{\alpha}^{S'}, \mathbf{p}^{S'}(\mathbf{d}) \rangle. \tag{2.18}$$

By Claim 1, let $S = N$, we can find $\boldsymbol{\alpha} \succ \mathbf{0}$ satisfying (2.16) which implies $\mathbf{q} \in R_{\mathrm{IB}}$. Therefore, it suffices to prove Claim 1. We prove this by induction on the cardinality $|S|$ of user set S.

If $|S| = 0$, clearly Claim 1 is correct.

Suppose Claim 1 is correct for all S with $|S| \leq k-1$ where $k \geq 1$. Given an S with $|S| = k$, by Lemma 2.8.3, we can find nonzero $\boldsymbol{\beta}^S \succeq \mathbf{0}$ satisfying the conditions in Lemma 2.8.3. We separate the set S into two sets S_1 and S_2 where

$$\beta_i^S > 0, i \in S_1,$$
$$\beta_i^S = 0, i \in S_2.$$

Since $\boldsymbol{\beta}^S \neq \mathbf{0}$, $|S_2| \leq |S| - 1 = k - 1$. By induction of Claim 1 on S_2, there exists $\boldsymbol{\gamma}^{S_2} \succ_{S_2} \mathbf{0}$ such that for any $S' \subseteq S_2$,

$$\langle \boldsymbol{\gamma}^{S'}, \mathbf{q}^{S'} \rangle \leq \min_{\mathbf{d} \in D(S')} \langle \boldsymbol{\gamma}^{S'}, \mathbf{p}^{S'}(\mathbf{d}) \rangle. \tag{2.19}$$

We claim that for small enough $\delta > 0$,

$$\boldsymbol{\alpha}^S = \boldsymbol{\beta}^S + \delta \boldsymbol{\gamma}^{S_2} \succ_S \mathbf{0}$$

satisfies condition (2.18) for all $S' \subseteq S$.

Any $S' \subseteq S$ falls into one of the following two cases: $S' \subseteq S_2$ and $S' \not\subseteq S_2$. It suffices to show (2.18) in each case.

If $S' \subseteq S_2$, then $\boldsymbol{\alpha}^{S'} = \boldsymbol{\gamma}^{S'}$. By (2.19), we know (2.18) is correct.

If $S' \not\subseteq S_2$, then $\boldsymbol{\beta}^{S'} \neq \mathbf{0}$. Let $\boldsymbol{\gamma}^{S'} = (\boldsymbol{\gamma}^{S_2})^{S'}$. We know

$$\langle \boldsymbol{\alpha}^{S'}, \mathbf{q}^{S'} \rangle = \langle \boldsymbol{\beta}^{S'}, \mathbf{q}^{S'} \rangle + \delta \langle \boldsymbol{\gamma}^{S'}, \mathbf{q}^{S'} \rangle.$$

By Lemma 2.8.3, $\langle \boldsymbol{\beta}^{S'}, \mathbf{q}^{S'} \rangle < \min_{\mathbf{d} \in D(S')} \langle \boldsymbol{\beta}^{S'}, \mathbf{p}^{S'}(\mathbf{d}) \rangle$. Since there are finite subsets S', for small enough δ,

$$\begin{aligned} \langle \boldsymbol{\alpha}^{S'}, \mathbf{q}^{S'} \rangle &\leq \min_{\mathbf{d} \in D(S')} \langle \boldsymbol{\beta}^{S'}, \mathbf{p}^{S'}(\mathbf{d}) \rangle \\ &\leq \min_{\mathbf{d} \in D(S')} \langle \boldsymbol{\alpha}^{S'}, \mathbf{p}^{S'}(\mathbf{d}) \rangle, \end{aligned}$$

i.e., (2.18) holds true.

In summary, this proves Claim 1 and thus $\mathbf{q} \in R_{\mathrm{IB}}$. Therefore, $\mathrm{int}(R) \subseteq R_{\mathrm{IB}}$.

2.8.5 Proof of Lemma 2.8.2

First we introduce a further notation. Given a decision $\mathbf{d}$ and a user set S, we let $m(\mathbf{d}, S)$ represent the decision that satisfies

- $m(\mathbf{d}, S) \in D(S)$.
- For users $i, j \in S$ or $i, j \notin S$, if i has higher priority than j in $\mathbf{d}$, then i also has higher priority than j in decision $m(\mathbf{d}, S)$.

In other words, $m(\mathbf{d}, S)$ is the priority decision obtained by modifying decision $\mathbf{d}$ to assign highest priorities to users in S without changing the relative orders in and out of S, respectively.

Given that the system satisfies monotonicity in payoffs, for all $i \in S$, we have that

$$p_i(m(\mathbf{d}, S)) \geq p_i(\mathbf{d}). \tag{2.20}$$

Given $\mathbf{q} \in C$, the goal is to show $\mathbf{q}^S \in C^S$ for all subsets of users S. By definition of C, there exists a convex combination of vectors in P that dominates $\mathbf{q}$, i.e., there exists $\{\alpha_{\mathbf{d}} | \mathbf{d} \in D\}$ such that

$$\mathbf{q} \preceq \sum_{\mathbf{d} \in D} \alpha_{\mathbf{d}} \mathbf{p}(\mathbf{d}),$$

and $\alpha_{\mathbf{d}} \geq 0$, $\sum\limits_{\mathbf{d} \in D} \alpha_{\mathbf{d}} = 1$.

Therefore, for any subset of users S,

$$\mathbf{q}^S \preceq \sum_{\mathbf{d} \in D} \alpha_{\mathbf{d}} \mathbf{p}^S(\mathbf{d}),$$

which by (2.20) gives

$$\mathbf{q}^S \preceq \sum_{\mathbf{d} \in D} \alpha_{\mathbf{d}} \mathbf{p}^S(m(\mathbf{d}, S)).$$

We let $\mathbf{x}^S = \sum\limits_{\mathbf{d} \in D} \alpha_{\mathbf{d}} \mathbf{p}^S(m(\mathbf{d}, S))$. Since $m(\mathbf{d}, S) \in D(S)$, we know $\mathbf{x}^S \in \mathrm{Conv}(P^S)$, and by definition of C^S,

$$\mathbf{q}^S \in C^S.$$

2.8.6 Proof of Lemma 2.8.3

Given $\mathbf{q} \in \mathrm{int}(R)$, by (2.12) we know that for all subsets of users S, $\mathbf{q}^S \in \mathrm{int}^S(C^S) \setminus B^S$. Given a subset of users S, the goal is to find $\boldsymbol{\beta}^S$ which satisfies the requirements in Lemma 2.8.3. In this proof, we focus on the subspace of S.

Since $\mathbf{q}^S \in \mathrm{int}^S(C^S)$, there exists $\mathbf{x}^S \in \mathrm{Conv}(P^S)$ such that $\mathbf{q}^S \prec_S \mathbf{x}^S$. Since $\mathbf{q}^S \notin B^S$ and $\mathbf{x}^S \in \mathrm{Conv}(P^S) \subseteq B^S$, we know that connecting $\mathbf{q}^S$ and $\mathbf{x}^S$ intersects $\mathrm{bd}^S(B^S)$ at some point denoted by $\mathbf{v}^S$. By the closure property of B^S, $\mathbf{v}^S \in B^S$ and thus, $\mathbf{v}^S \succ_S \mathbf{q}^S$. Since $\mathbf{v}^S \in \mathrm{bd}^S(B^S)$, we get that $\mathbf{v}^S$ lies on a supporting hyperplane [10] of B^S, and by definition of B^S, there exists nonzero normal vector $\boldsymbol{\beta}^S \succeq \mathbf{0}$ of this supporting hyperplane such that

$$\langle \boldsymbol{\beta}^S, \mathbf{v}^S \rangle = \min_{\mathbf{d} \in D(S)} \langle \boldsymbol{\beta}^S, \mathbf{p}^S(\mathbf{d}) \rangle. \tag{2.21}$$

Figure 2.8 conceptually shows the process of constructing $\boldsymbol{\beta}^S$. The circles represent the expected payoff vectors. For simplicity we suppress the superscript S in the figure. We shall show this $\boldsymbol{\beta}^S$ satisfies the requirements in Lemma 2.8.3.

Since $\mathbf{v}^S \in \mathrm{bd}^S(B^S) \subseteq B^S$, there exists $\mathbf{u}^S \in \mathrm{Conv}(P^S)$ such that $\mathbf{v}^S \succeq \mathbf{u}^S$, and by (2.21), we get that

$$\langle \boldsymbol{\beta}^S, \mathbf{u}^S \rangle \leq \min_{\mathbf{d} \in D(S)} \langle \boldsymbol{\beta}^S, \mathbf{p}^S(\mathbf{d}) \rangle.$$

The vector $\mathbf{u}^S$ is also shown in Figure 2.8.

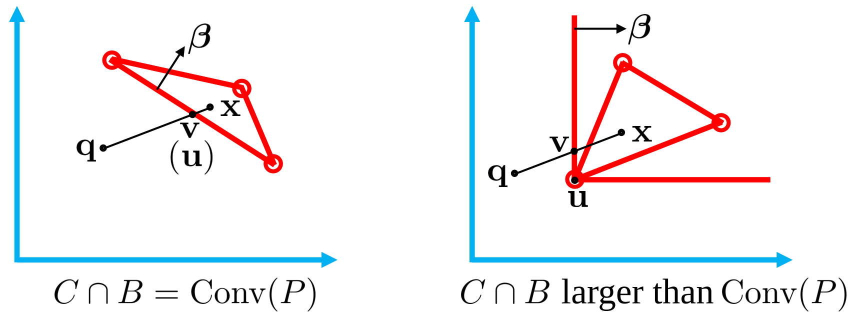

Figure 2.8: The process of constructing $\boldsymbol{\beta}$ when $C \cap B$ equals to $\mathrm{Conv}(P)$ (left figure), and when $C \cap B$ is larger than $\mathrm{Conv}(P)$ (right figure).

On the other hand, since $\mathbf{u}^S \in \mathrm{Conv}(P^S)$, by similar analysis as in (2.15) we know that

$$\langle \boldsymbol{\beta}^S, \mathbf{u}^S \rangle \geq \min_{\mathbf{d} \in D(S)} \langle \boldsymbol{\beta}^S, \mathbf{p}^S(\mathbf{d}) \rangle.$$

Thus,

$$\langle \boldsymbol{\beta}^S, \mathbf{u}^S \rangle = \langle \boldsymbol{\beta}^S, \mathbf{v}^S \rangle = \min_{\mathbf{d} \in D(S)} \langle \boldsymbol{\beta}^S, \mathbf{p}^S(\mathbf{d}) \rangle,$$

which implies $u_i^S = v_i^S$ if $\beta_i^S \neq 0$. Since $\mathbf{v}^S \succ_S \mathbf{q}^S$, for all subset $S' \subseteq S$ where $\boldsymbol{\beta}^{S'} \neq \mathbf{0}$, we have that

$$\langle \boldsymbol{\beta}^{S'}, \mathbf{q}^{S'} \rangle < \langle \boldsymbol{\beta}^{S'}, \mathbf{u}^{S'} \rangle,$$

Therefore, to show (2.17) it suffices to show for all $S' \subseteq S$ where $\boldsymbol{\beta}^{S'} \neq \mathbf{0}$ that

$$\langle \boldsymbol{\beta}^{S'}, \mathbf{u}^{S'} \rangle \leq \min_{\mathbf{d} \in D(S')} \langle \boldsymbol{\beta}^{S'}, \mathbf{p}^{S'}(\mathbf{d}) \rangle. \tag{2.22}$$

Given $\mathbf{u}^S \in \mathrm{Conv}(P^S)$, we write $\mathbf{u}^S = \sum_{\mathbf{d} \in D(S)} c_{\mathbf{d}} \mathbf{p}^S(\mathbf{d})$ where $c_{\mathbf{d}} \geq 0$ and $\sum_{\mathbf{d} \in D(S)} c_{\mathbf{d}} = 1$.

For all $S' \subseteq S$ where $\boldsymbol{\beta}^{S'} \neq \mathbf{0}$, we can rewrite $\langle \boldsymbol{\beta}^{S'}, \mathbf{u}^{S'} \rangle$ as follows,

$$\begin{aligned} & \langle \boldsymbol{\beta}^{S'}, \mathbf{u}^{S'} \rangle \\ & = \langle \boldsymbol{\beta}^{S}, \mathbf{u}^{S} \rangle - \langle \boldsymbol{\beta}^{S \setminus S'}, \mathbf{u}^{S \setminus S'} \rangle \\ & = \min_{\mathbf{d} \in D(S)} \langle \boldsymbol{\beta}^{S}, \mathbf{p}^{S}(\mathbf{d}) \rangle - \sum_{\mathbf{d} \in D(S)} c_{\mathbf{d}} \langle \boldsymbol{\beta}^{S \setminus S'}, \mathbf{p}^{S \setminus S'}(\mathbf{d}) \rangle \\ & \leq \min_{\mathbf{d} \in D(S)} \langle \boldsymbol{\beta}^{S}, \mathbf{p}^{S}(\mathbf{d}) \rangle - \min_{\mathbf{d} \in D(S)} \langle \boldsymbol{\beta}^{S \setminus S'}, \mathbf{p}^{S \setminus S'}(\mathbf{d}) \rangle. \end{aligned}$$

To show (2.22), it suffices to show that

$$\begin{aligned} & \min_{\mathbf{d} \in D(S)} \langle \boldsymbol{\beta}^{S}, \mathbf{p}^{S}(\mathbf{d}) \rangle \\ & \leq \min_{\mathbf{d} \in D(S)} \langle \boldsymbol{\beta}^{S \setminus S'}, \mathbf{p}^{S \setminus S'}(\mathbf{d}) \rangle \qquad (2.23) \\ & \quad + \min_{\mathbf{d} \in D(S')} \langle \boldsymbol{\beta}^{S'}, \mathbf{p}^{S'}(\mathbf{d}) \rangle. \qquad (2.24) \end{aligned}$$

Suppose $\mathbf{d}_1 \in D(S)$ and $\mathbf{d}_2 \in D(S')$ are the optimal solutions for (2.23) and (2.24), respectively. Formally,

$$\langle \boldsymbol{\beta}^{S \setminus S'}, \mathbf{p}^{S \setminus S'}(\mathbf{d}_1) \rangle = \min_{\mathbf{d} \in D(S)} \langle \boldsymbol{\beta}^{S \setminus S'}, \mathbf{p}^{S \setminus S'}(\mathbf{d}) \rangle,$$

$$\langle \boldsymbol{\beta}^{S'}, \mathbf{p}^{S'}(\mathbf{d}_2) \rangle = \min_{\mathbf{d} \in D(S')} \langle \boldsymbol{\beta}^{S'}, \mathbf{p}^{S'}(\mathbf{d}) \rangle.$$

We consider the unique decision $\mathbf{d}_3$ that satisfies the following: First, $\mathbf{d}_3 \in D(S')$, i.e., $\mathbf{d}_3$ assigns highest priority to users in S'. Second, the priority ordering for user subset S' in $\mathbf{d}_3$ are the same as those in $\mathbf{d}_2$. Third, the priority ordering for user subset $N \setminus S'$ in $\mathbf{d}_3$ are the same as those in $\mathbf{d}_1$.

Since $\mathbf{d}_1 \in D(S)$, we know $\mathbf{d}_3 \in D(S)$ and therefore,

$$\min_{\mathbf{d} \in D(S)} \langle \boldsymbol{\beta}^{S}, \mathbf{p}^{S}(\mathbf{d}) \rangle \leq \langle \boldsymbol{\beta}^{S}, \mathbf{p}^{S}(\mathbf{d}_3) \rangle.$$

Now it suffices to show that

$$\langle \boldsymbol{\beta}^S, \mathbf{p}^S(\mathbf{d}_3)\rangle \le \langle \boldsymbol{\beta}^{S\setminus S'}, \mathbf{p}^{S\setminus S'}(\mathbf{d}_1)\rangle + \langle \boldsymbol{\beta}^{S'}, \mathbf{p}^{S'}(\mathbf{d}_2)\rangle.$$

This is true because given that the system satisfies monotonicity in payoffs, we can get that

$$p_i^{S'}(\mathbf{d}_3) \le p_i^{S'}(\mathbf{d}_2) \text{ for } i \in S',$$

and

$$p_i^{S\setminus S'}(\mathbf{d}_3) \le p_i^{S\setminus S'}(\mathbf{d}_1) \text{ for } i \in S \setminus S'.$$

In summary, this proves (2.22) and thus $\boldsymbol{\beta}^S$ satisfies the conditions in Lemma 2.8.3.

2.8.7 Proof of Theorem 2.3.5

Clearly $F_{\mathbf{w}\text{-LDF}} \subseteq F \subseteq \mathrm{cl}(C)$. To show $\mathrm{int}(C) \subseteq F_{\mathbf{w}\text{-LDF}}$, by (2.5) it suffices to show $\mathrm{int}(C) \subseteq \mathrm{int}(R)$.

Given $\mathbf{q} \in \mathrm{int}(C)$, the goal is to show $\mathbf{q} \in \mathrm{int}(R)$, i.e., for all user subsets S,

$$\mathbf{q}^S \in \mathrm{int}^S(C^S) \setminus B^S.$$

Given a user subset S, by Lemma 2.8.2 we know $\mathbf{q}^S \in C^S$. Further we can show $\mathbf{q}^S \in \mathrm{int}^S(C^S)$ since otherwise $\mathbf{q} \in \mathrm{bd}(C)$. By definition of interior and C^S, there exists $\mathbf{x}^S \in \mathrm{Conv}(P^S)$ such that $\mathbf{q}^S \prec_S \mathbf{x}^S$.

Suppose $\mathbf{q}^S \in B^S$, by definition there exists $\mathbf{y}^S \in \mathrm{Conv}(P^S)$ such that $\mathbf{q}^S \succeq \mathbf{y}^S$. Now we get two vectors $\mathbf{x}^S, \mathbf{y}^S \in \mathrm{Conv}(P^S)$ and $\mathbf{x}^S \succ_S \mathbf{y}^S$. Since vectors in P^S lie on a hyperplane and $\mathbf{x}^S, \mathbf{y}^S \in \mathrm{Conv}(P^S)$, there exists nonzero $\boldsymbol{\alpha}^S \succeq \mathbf{0}$ such that

$$\langle \boldsymbol{\alpha}^S, \mathbf{x}^S \rangle = \langle \boldsymbol{\alpha}^S, \mathbf{y}^S \rangle,$$

which contradicts with $\mathbf{x}^S \succ_S \mathbf{y}^S$.

Therefore, $\mathbf{q}^S \notin B^S$ and thus $\mathrm{int}(C) \subseteq \mathrm{int}(R)$.

2.8.8 Proof of Theorem 2.3.7

Given monotonicity in payoffs, by (2.5) we know $\mathrm{int}(R) \subseteq F_{\mathbf{w}\text{-LDF}}$ and F differs from C by at most a boundary. By the definition of the efficiency ratio, we know

$$\gamma_{\mathbf{w}\text{-LDF}} = \sup\{\gamma | \gamma F \subseteq F_{\mathbf{w}\text{-LDF}}\} \geq \sup\{\gamma | \gamma C \subseteq \mathrm{int}(R)\} = \sup\{\gamma | \gamma C \subseteq \mathrm{cl}(R)\}.$$

To show $\gamma_{\mathbf{w}\text{-LDF}} \geq \min_{S \subseteq N} \sigma_S$, it suffices to show $\sup\{\gamma | \gamma C \subseteq \mathrm{cl}(R)\} \geq \min_{S \subseteq N} \sigma_S$, i.e., for each $\mathbf{q} \in C$, we have $\min_{S \subseteq N} \sigma_S \cdot \mathbf{q} \in \mathrm{cl}(R)$, which is equivalent to showing that for each $\mathbf{q} \in C$, there exists a subset of users $S \subseteq N$, such that $\sigma_S \cdot \mathbf{q} \in \mathrm{cl}(R)$.

Given a $\mathbf{q} \in C$, we define $\lambda(\mathbf{q}, R) = \sup\{\lambda | \lambda \mathbf{q} \in R\}$ which represents how far the vector $\mathbf{q}$ can extend before it goes beyond the region R and let $\mathbf{q}_{(R)} = \lambda(\mathbf{q}, R) \cdot \mathbf{q}$. We claim there exists a user subset S such that $\mathbf{q}^S_{(R)} \in \mathrm{bd}^S(B^S)$ since otherwise we can increase $\lambda(\mathbf{q}, R)$ while guaranteeing $\mathbf{q}_{(R)}$ is still in R. Next we shall show $\sigma_S \cdot \mathbf{q} \in \mathrm{cl}(R)$, i.e., $\lambda(\mathbf{q}, R) \geq \sigma_S$.

Since $\mathbf{q}^S_{(R)} \in \mathrm{bd}^S(B^S) \subseteq B^S$, there exists $\mathbf{x}^S \in \mathrm{Conv}(P^S)$ such that

$\mathbf{q}^S_{(R)} \succeq \mathbf{x}^S$. Since $\mathbf{q} \in C$, by Lemma 2.8.2 we know $\mathbf{q}^S \in C^S$ and thus, there exists $\mathbf{y}^S \in \mathrm{Conv}(P^S)$ such that $\mathbf{q}^S \preceq \mathbf{y}^S$.

Given that $\mathbf{q}_{(R)} = \lambda(\mathbf{q}, R) \cdot \mathbf{q}$, for any nonzero $\boldsymbol{\alpha}^S \succeq \mathbf{0}$, we have that

$$\langle \mathbf{q}^S_{(R)}, \boldsymbol{\alpha}^S \rangle = \lambda(\mathbf{q}, R) \cdot \langle \mathbf{q}^S, \boldsymbol{\alpha}^S \rangle.$$

By $\mathbf{q}^S_{(R)} \succeq \mathbf{x}^S$ and $\mathbf{q}^S \preceq \mathbf{y}^S$, we get that

$$\langle \mathbf{x}^S, \boldsymbol{\alpha}^S \rangle \leq \lambda(\mathbf{q}, R) \cdot \langle \mathbf{y}^S, \boldsymbol{\alpha}^S \rangle.$$

Since $\mathbf{x}^S \in \mathrm{Conv}(P^S)$, by similar analysis as in (2.15) we know $\langle \mathbf{x}^S, \boldsymbol{\alpha}^S \rangle \geq \min\limits_{\mathbf{d} \in D(S)} \langle \boldsymbol{\alpha}^S, \mathbf{p}^S(\mathbf{d}) \rangle$. Similarly we can show $\langle \mathbf{y}^S, \boldsymbol{\alpha}^S \rangle \leq \max\limits_{\mathbf{d} \in D(S)} \langle \boldsymbol{\alpha}^S, \mathbf{p}^S(\mathbf{d}) \rangle$. Thus,

$$\min_{\mathbf{d} \in D(S)} \langle \boldsymbol{\alpha}^S, \mathbf{p}^S(\mathbf{d}) \rangle \leq \lambda(\mathbf{q}, R) \cdot \max_{\mathbf{d} \in D(S)} \langle \boldsymbol{\alpha}^S, \mathbf{p}^S(\mathbf{d}) \rangle. \tag{2.25}$$

Clearly, $\max\limits_{\mathbf{d} \in D(S)} \langle \boldsymbol{\alpha}^S, \mathbf{p}^S(\mathbf{d}) \rangle > 0$ for any nonzero $\boldsymbol{\alpha}^S \succeq \mathbf{0}$.

Since (2.25) is true for any nonzero $\boldsymbol{\alpha}^S \succeq \mathbf{0}$, we get that

$$\lambda(\mathbf{q}, R) \geq \max_{\substack{\boldsymbol{\alpha}^S \succeq \mathbf{0} \\ \boldsymbol{\alpha}^S \neq \mathbf{0}}} \frac{\min\limits_{\mathbf{d} \in D(S)} \langle \boldsymbol{\alpha}^S, \mathbf{p}^S(\mathbf{d}) \rangle}{\max\limits_{\mathbf{d} \in D(S)} \langle \boldsymbol{\alpha}^S, \mathbf{p}^S(\mathbf{d}) \rangle} = \sigma_S.$$

Therefore, for each $\mathbf{q} \in C$, there exists a subset of users $S \subseteq N$ such that $\sigma_S \cdot \mathbf{q} \in \mathrm{cl}(R)$, and thus, $\gamma_{\mathbf{w}\text{-LDF}} \geq \min\limits_{S \subseteq N} \sigma_S$.

2.8.9 Proof of Corollary 2.4.1

By Theorem 2.3.5, to show **w**-LDF policies are feasibility optimal, it suffices to show the system satisfies subset payoff equivalence. To show this, it suffices to show for all user subsets $S \subseteq N$ and all priority decisions $\mathbf{d}_1, \mathbf{d}_2 \in D(S)$ that

$$\langle \mathbf{1}^S, \mathbf{p}^S(\mathbf{d}_1) \rangle = \langle \mathbf{1}^S, \mathbf{p}^S(\mathbf{d}_2) \rangle.$$

This is true because we can convert $\mathbf{d}_1$ to $\mathbf{d}_2$ by repeatedly switching a pair of users in $\mathbf{d}_1$ at each step such that at step k both decisions assign the highest k priorities to the same users, respectively. By the definition of exchangeable expected payoffs, the sum of the expected payoffs for users in S remains the same at each step.

Chapter 3

LDF Prioritization and Scheduling for Computing Systems with Uniform Resources

3.1 Introduction

In Chapter 2 we have analyzed a general class of systems using priority-based resource allocation schemes. In this chapter we focus on a specific model for 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

This chapter is based on my work in [24] (submitted to INFOCOM 2016) and [25] (submitted to ACM Transactions on Modeling and Performance Evaluation of Computing Systems), which are co-authored by Dr. Gustavo de Veciana.

memory. In this chapter 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 [42], the community has established theoretical frameworks to study the scheduling of real-time applications where tasks are subject to hard deadlines, see e.g., [12, 17, 40, 44]. 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 [7, 30] 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 there is no analysis of approaches to meet such requirements. The authors in [33, 45] 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 [43] 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 chapter 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 [4] 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 [32, 33] 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 chapter we consider prioritization policies that use the idea underlying longest-queue-first policies, whose performance has been studied in [19, 36, 38] 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., [1, 2, 9, 11, 39, 58] 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., [35, 57, 62], and those studying user/job management, see e.g., [3, 18, 48].

Our Contributions. In this chapter, 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 chapter 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 lax, 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, e.g., to resources with different processing speeds, and user management for computing systems.

Organization of the Chapter. This chapter is organized as follows: Section 3.2 introduces our system model and Section 3.3 describes a reservation-based approach and a general outer bound for the feasibility region. Section 3.4 discusses two prioritization-based policies and studies their efficiency ratios. Simulation results are exhibited in Section 3.5. Section 3.6 discusses generalizations and Section 3.7 discusses user management for computing systems. Section 3.8 concludes the chapter and some of the proofs are provided in the Appendix.

3.2 System Model

We first introduce our user, system and QoS models.

3.2.1 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 it 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 3.2.1. A non-negative random variable W is said to satisfy **New Better than Used in Expectation (NBUE)** if for all $t > 0$,

$$\mathrm{E}[W - t | W > t] \leq \mathrm{E}[W]. \tag{3.1}$$

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

The exponential and geometric distributions are special NBUE distributions since they result in equality in the above for all $t > 0$ and for all integers $t > 0$, respectively. The notion of NBUE, see e.g., [53, 60], captures workload/lifetime distributions where the expected residual workload/lifetime of a task/device of age t is no more than that of a new task/device. By [60] the set of NBUE distributions is closed under convolution. Formally one can show the following results:

Proposition 3.2.1. *The NBUE property satisfies the following,*

- *If two independent random variables W, V are NBUE, then $W + V$ is NBUE.*

- *If a random variable W is NBUE, then for all real numbers $c > 0$, cW is NBUE.*

The NBUE property characterizes many workload distributions of interest, e.g., [53] provides a discussion of NBUE distributions including 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

[1]In fact, we only require (3.1) to be true for $0 < t \leq \delta$.

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

dropped. In Section 3.6, we discuss possible generalizations where users may generate tasks at different periods and where a task may further consist of sub-tasks.

3.2.2 Computing Infrastructure

A computing system can be very complex consisting of diverse, heterogeneous resources. In this chapter, for simplicity, we start by considering a computing system comprised of m identical resources (cores)—a simple but relevant model. In Section 3.6 we discuss generalizations where cores have different processing speeds.

Given m identical cores, a task processed on any core has the same processing time distribution 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 limited resources 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 they 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.

3.2.3 SRT QoS Feasibility

Given a QoS 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[3]

$$X_i(t+1) = [X_i(t) + q_i - Y_i(t+1)]^+, \tag{3.2}$$

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 $\mathbf{X}(t) =$

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

$(X_1(t), X_2(t), \cdots, X_n(t))$ 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". Formally, in this chapter we consider non-clairvoyant resource allocation policies under which the process $\{\mathbf{X}(t)\}_{t\geq 1}$ is a Markov chain[4]. 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 3.2.2. 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

[4]All the results in this chapter 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.

random workloads. It is parameterized by the number of cores m, number of users n, period length δ, QoS requirements $\mathbf{q}$, and the NBUE workload distributions.

3.2.4 Summary of System Model and Assumptions

To summarize, the problem is to devise non-clairvoyant resource allocation policies to meet users' long-term QoS requirements and here are the assumptions:

- The system consists of m identical cores and n users.
- The system operates over periods of length δ.
- In each period, each user generate one task that is available for processing at the beginning of the period and has to be processed by the end of the period. Tasks not completed on time are dropped.
- The task workloads for a given user are i.i.d. across periods and workloads from different users are independent, possibly with different distributions. We assume all task workloads are NBUE.
- q_i is the required percentage of tasks completed on time for user i. We assume all q_i's are rational.
- We consider non-clairvoyant resource allocation policies and we ignore the overheads of task preemption and migration.

- We further assume the initial state $\mathbf{X}(0)$, the requirements $\mathbf{q}$ and the policy make $\{\mathbf{X}(t)\}_{t\geq 1}$ an irreducible Markov chain.

3.3 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.

3.3.1 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 $w_i(q_i)$ be 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. Figure 3.1 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.

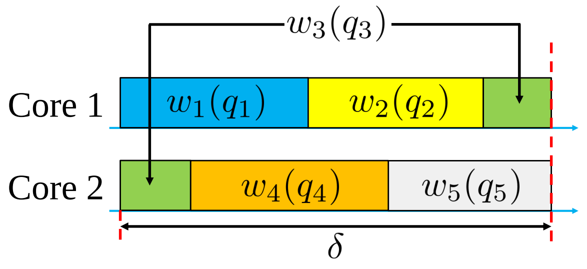

Figure 3.1: An example of the reservation-based approach.

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 chapter, we assume the task workloads and requirements $\mathbf{q}$ are such that for all i, $w_i(q_i)$ are bounded by δ.

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

$$F_{\text{RB}} = \{\mathbf{q} \in \mathbb{R}^n_+ \mid \mathbf{q} \preceq \mathbf{1}, \sum_{i \in N} w_i(q_i) \leq m\delta, \\ w_i(q_i) \leq \delta, \forall i \in N\}, \tag{3.3}$$

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 [4] and is also loosely used in reservation based schemes adopted in modern cloud infrastructure, see e.g.,

[68]. 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, see e.g., [68], the resources are then used to support other, e.g., best effort, traffic.

3.3.2 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 $\text{int}(F_\eta) \subseteq F \subseteq \text{cl}(F_\eta)$, where $\text{int}(F_\eta)$ and $\text{cl}(F_\eta)$ is the interior and closure of F_η, and thus is 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., [32]. To evaluate possible resource allocation policies, we provide a benchmark based on a simple outer bound R_{OB} for F given in the following theorem.

Theorem 3.3.1. *For the SRT-MIC model with NBUE workloads, the system feasibility region F is such that*

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

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 $\sum\limits_{i\in N} q_i\mu_i$ 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 $\sum\limits_{i\in N} q_i\mu_i$ 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 $\sum\limits_{i\in N} q_i\mu_i \le m\delta$.

Suppose $\mathbf{q}$ is fulfilled by a non-clairvoyant resource allocation policy η, by Definition 3.2.2, it follows that under η the process $\{\mathbf{X}(t)\}_{t\ge 1}$ is positive recurrent and therefore, there exists a stationary distribution. 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 let Y_i be the indicator random variable that the task from user i completes in a typical period. By the Ergodic Theorem, $\mathrm{E}[Y_i]$ 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 core 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 complete. For each user i, we further define $E_i = A_i(W_i - U_{\{i\}})$. Intuitively, E_i represents the "residual workload 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

$$\mathrm{E}[U_S] = \sum_{i\in S} \mathrm{E}[(Y_i + A_i)W_i] - \sum_{i\in S} \mathrm{E}[E_i]. \tag{3.4}$$

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 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 (3.4) becomes

$$\mathrm{E}[U_S] = \sum_{i \in S} \mathrm{E}[Y_i]\mu_i + \sum_{i \in S} \mathrm{E}[A_i]\mu_i - \sum_{i \in S} \mathrm{E}[E_i]. \tag{3.5}$$

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 (3.5) 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.

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

$$\begin{aligned}\mathrm{E}[E_i] &= \sum_{c=1}^{\delta} \mathrm{E}[E_i | A_{i,c} = 1] \Pr(A_{i,c} = 1) \\ &= \sum_{c=1}^{\delta} \mu_{i,c} \, \mathrm{E}[A_{i,c}], \qquad (3.6)\end{aligned}$$

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,

$$\mathrm{E}[E_i] \leq \sum_{c=1}^{\delta} \mu_i \, \mathrm{E}[A_{i,c}] = \mu_i \, \mathrm{E}[A_i]. \qquad (3.7)$$

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 (3.5) and (3.7) we know that given a feasible requirement vector $\mathbf{q}$, for all user subsets $S \subseteq N$,

$$\sum_{i \in S} q_i \mu_i \leq \sum_{i \in S} \mathrm{E}[Y_i] \mu_i \leq \mathrm{E}[U_S] \leq m\delta, \qquad (3.8)$$

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

A key part of this argument is the inequality (3.8), 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. 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 (3.8) and Theorem 3.3.1. Non-NBUE workloads are beyond the scope of this chapter. Yet for real-time computing workloads we expect NBUE to be a good assumption.

3.4 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 al-*

location policies, which decompose resource allocation into two sub-problems, see Figure 3.2:

1. User prioritization: in each period the system dynamically prioritizes users based on the history of outcomes.
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 the task subset via optimal scheduling policies.

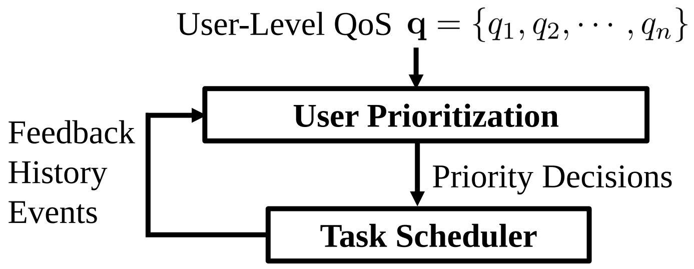

Figure 3.2: The framework for prioritization-based resource allocation policies.

In this chapter 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 3.4.1. 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.

3.4.1 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 3.4.1. *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$(R_{IB}) \subseteq F_{LDF+\mathcal{X}}$, where*

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

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 3.9.1.

Next we explore specific task schedulers and use Theorem 3.4.1 to study their performance.

3.4.2 Performance Analysis of LDF+Greedy Scheduling

Given an LDF-based user prioritization 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. The framework is exhibited in Figure 3.3. 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.

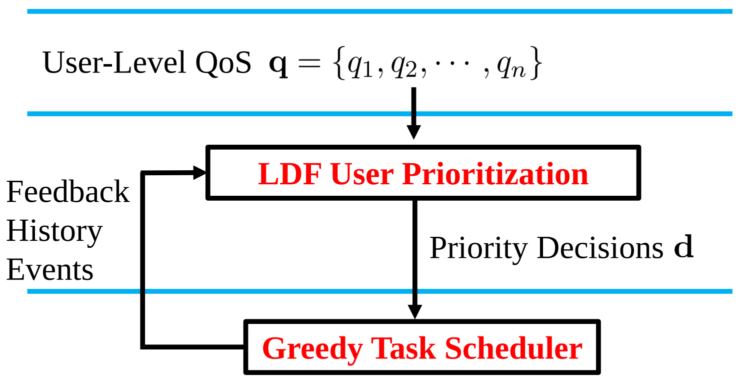
User-Level QoS $\mathbf{q} = \{q_1, q_2, \cdots, q_n\}$
LDF User Prioritization
Feedback
History
Events
Priority Decisions $\mathbf{d}$
Greedy Task Scheduler

Figure 3.3: The framework for LDF+Greedy scheduling.

Figure 3.4 shows an example realization of greedy task scheduling for a system with 2 cores and where the selected priority decision is $\mathbf{d} = (1, 2, \cdots, n)$. Note that the execution of tasks under the greedy scheduler is not pre-determined and actually depends on the realizations of tasks' workloads, still it is non-clairvoyant.

Next we characterize the performance of LDF+Greedy. To that end, we introduce a metric called the efficiency ratio, see e.g., [36]. The *efficiency ratio* of a non-clairvoyant resource allocation policy η is defined as

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

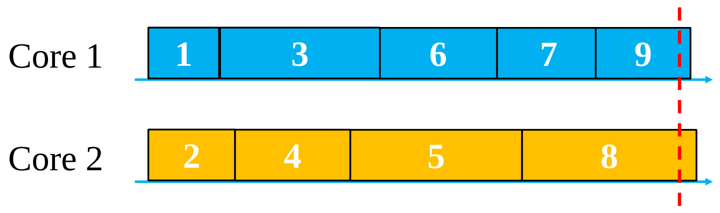

Figure 3.4: An example for the greedy task scheduler if the selected priority decision is $\mathbf{d} = (1, 2, \cdots, n)$.

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 the η policy is feasibility optimal.

Theorem 3.4.2. *For the SRT-MIC model with NBUE workloads, the efficiency ratio of LDF+Greedy exceeds γ_1 where*

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

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 $\max\limits_{i \in N} \mu_i$. Given the period is of length δ, the gap between LDF+Greedy and optimality is bounded by $\frac{\max\limits_{i \in N} \mu_i}{\delta}$. Note that again this argument is deceptively simplified since unfinished tasks might tend to have larger workloads. Also as for Theorem 3.3.1, this result does not necessarily hold for non-NBUE workloads. The formal proof is given below.

Proof. Given a requirement vector $\mathbf{q}$ fulfilled by resource allocation policy η, by (3.8) 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 $\sum\limits_{i \in S} W_i$ of users in S and the total available core time $m\delta$. We define $T_S = \min \left[\sum\limits_{i \in S} W_i, m\delta \right]$ and therefore, for all user subsets S, we have that

$$\sum_{i \in S} q_i \mu_i \leq \mathrm{E}[U_S] \leq \mathrm{E}[T_S]. \tag{3.9}$$

Thus, for a vector $\mathbf{q}$ satisfying (3.9) the aim is 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 3.4.1, it suffices to show that $\gamma_1 \mathbf{q} \in R_{\text{IB}}$. For LDF+Greedy, the expected vector $\mathbf{p}(\mathbf{d})$ described in Section 3.4.1 represents the expected numbers of timely completions for the 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. By (3.9) 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

$$\sum_{i \in S} \mu_i p_i(\mathbf{d}) \geq \gamma_1 \operatorname{E}[T_S] = \operatorname{E}[T_S] - \frac{\max\limits_{i \in N} \mu_i}{\delta} \operatorname{E}[T_S]. \tag{3.10}$$

First we rewrite $\sum\limits_{i \in S} \mu_i p_i(\mathbf{d})$ by similar approach used to obtain (3.5). As in the proof of Theorem 3.3.1, 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 (3.5), for the given S and $\mathbf{d}$, we have that

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

Now (3.10) follows by showing that

$$\operatorname{E}[U_S(\mathbf{d})] + \sum_{i \in S} \operatorname{E}[E_i(\mathbf{d})] \geq \operatorname{E}[T_S] \tag{3.11}$$

and

$$\sum_{i \in S} \operatorname{E}[A_i(\mathbf{d})] \mu_i \leq \frac{\max\limits_{i \in N} \mu_i}{\delta} \operatorname{E}[T_S], \tag{3.12}$$

respectively.

To demonstrate (3.11), 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, $u_S(\mathbf{d}) + \sum_{i \in S} e_i(\mathbf{d}) = \sum_{i \in S} w_i \geq t_S$, where w_i represents the realization of workload W_i. Therefore, (3.11) is verified.

Now it remains to show (3.12). 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 (3.12) it suffices to show that

$$\sum_{i \in S} \mathrm{E}[A_i(\mathbf{d})] \leq \frac{\mathrm{E}[T_S]}{\delta}. \tag{3.13}$$

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 (3.13) which in turn shows (3.10) and therefore, $\gamma_1 \mathbf{q} \in R_{\mathrm{IB}} \subseteq \mathrm{cl}(F_{\mathrm{LDF+Greedy}})$.

$\square$

Theorem 3.4.2 provides a lower bound on the efficiency ratio of LDF+Greedy, denoted by $\gamma_{\mathrm{LDF+Greedy}}$. The bound is tight in the sense that for any $\epsilon > 0$, there exists an SRT-MIC system with NBUE workloads such that $\gamma_{\mathrm{LDF+Greedy}} < 1 - \frac{\max_{i \in N} \mu_i}{\delta} + \epsilon$. Such a system is detailed in Appendix 3.9.2.

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 prior work in [32]. This is due to the memoryless property of the exponential (or geometric) distribution. 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.

3.4.3 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.

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 [14], that one can complete all tasks in a user subset S in a period by some optimal scheduling if and only if $\sum\limits_{i \in S} \mu_i \leq m\delta$. We consider one such optimal algorithm: Largest Local Remaining Execution time First (LLREF) [14]. Let us briefly describe how LLREF[5] 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 [14]. 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 3.4.2. 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:

[5]LLREF is defined to be applicable in more general settings where users might generate tasks with different period. We will discuss this in Section 3.6.

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 [17].

Definition 3.4.3. 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
$$j(\mathbf{d}) = \max\Big\{j|\sum_{i=1}^{j}\mu_{d_i} \le m\delta\Big\}. \tag{3.14}$$
Let $J(\mathbf{d}) = \{d_1, d_2, \cdots, d_{j(\mathbf{d})}\}$ represent the selected user subset[6].

2. LLREF for $J(\mathbf{d})$: the system uses LLREF scheduling for tasks in $J(\mathbf{d})$ in this period.

By [14], it follows that all tasks from $J(\mathbf{d})$ will complete.

[6]One trivial way to extend the task selection step is to continue checking users greedily according to $\mathbf{d}$ and adding users to $J(\mathbf{d})$ while guaranteeing $\sum_{i\in J(\mathbf{d})}\mu_i \le m\delta$. But that does not improve the results in the sequel and therefore, we do not discuss this extension.

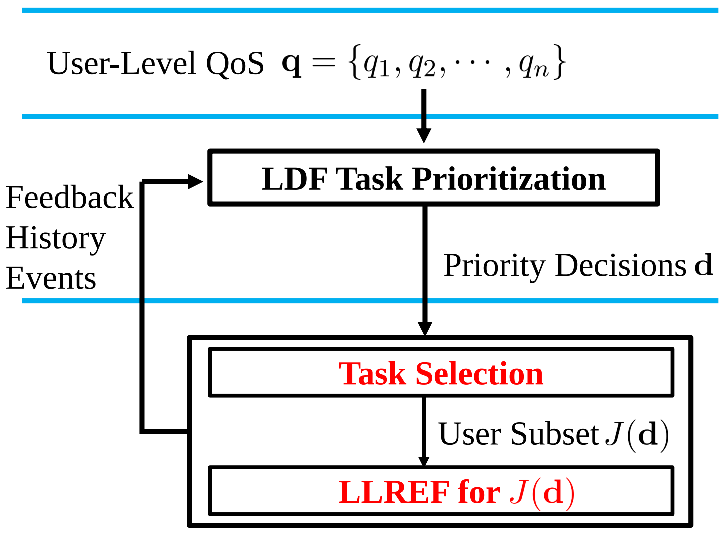
User-Level QoS $\mathbf{q} = \{q_1, q_2, \cdots, q_n\}$
LDF Task Prioritization
Feedback History Events
Priority Decisions $\mathbf{d}$
Task Selection
User Subset $J(\mathbf{d})$
LLREF for $J(\mathbf{d})$

Figure 3.5: The framework for LDF+TS/LLREF scheduling.

Paralleling Theorem 3.4.2, 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. The framework of LDF+TS/LLREF is exhibited in Figure 3.5.

Theorem 3.4.3. *For the SRT-MIC model with deterministic workloads, the efficiency ratio of LDF+TS/LLREF exceeds γ_2 where*

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

Intuitively, under TS/LLREF, the task selection rule guarantees that in any given period the wasted time $m\delta - \sum\limits_{i \in J(\mathbf{d})} \mu_i$ is less than $\max\limits_{i \in N} \mu_i$. 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 3.4.2 and is provided in Appendix 3.9.3.

The efficiency ratio lower bound γ_2 in this theorem is better than γ_1 obtained in Theorem 3.4.2, 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 but picking appropriate w_i^{est} is key to the performance. On one hand big w_i^{est} reduces the number of tasks selected per period and on the other hand small w_i^{est} may cause user i's task fail to complete. This will be explored in the simulation section.

3.4.4 Resource Requirements

So far we have analytically characterized the efficiency ratios of two LDF-based resource allocation policies. Another metric of interest is the resources needed, in terms of the number of cores m, 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.

3.4.4.1 Resource Requirements for Reservation-Based Static Sharing

Based on the definition of F_{RB} in 3.3.1, the required number of cores to fulfill the users' QoS requirements $\mathbf{q}$ under reservation-based static sharing is given by

$$m_{\text{RB}} = \left\lceil \frac{\sum\limits_{i \in N} w_i(q_i)}{\delta} \right\rceil, \tag{3.15}$$

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

3.4.4.2 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 3.3.1, we know m_η must satisfy $m_\eta \delta \geq \sum\limits_{i \in N} q_i \mu_i$, giving the following lower bound on the required number of cores:

$$\underline{m} \equiv \left\lceil \frac{\sum\limits_{i \in N} q_i \mu_i}{\delta} \right\rceil. \tag{3.16}$$

If we ignore the ceilings,

$$1 - \frac{\underline{m}}{m_{\text{RB}}} \simeq 1 - \frac{\sum\limits_{i \in N} q_i \mu_i}{\sum\limits_{i \in N} w_i(q_i)} \tag{3.17}$$

gives us a *upper bound* on the possible resource savings compared with reservation-based static sharing, i.e., the percentage of cores we can save by devising the best possible non-clairvoyant resource allocation policies. Clearly, this depends on the workload distributions and the requirement vector **q**. We will see in the simulation section that the proposed approaches can achieve this upper bound in some scenarios.

3.4.4.3 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 3.4.2 we know that LDF+Greedy may expect to waste up to $\max\limits_{i \in N} \mu_i$ time on each core in a period because of unfinished tasks. Thus, to complete an "effective" workload $\sum\limits_{i \in N} q_i \mu_i$, we propose an estimate for $m_{\text{LDF+Greedy}}$ as follows,

$$m^{\text{est}}_{\text{LDF+Greedy}} \equiv \left\lceil \frac{\sum\limits_{i \in N} q_i \mu_i}{\delta - \max\limits_{i \in N} \mu_i} \right\rceil. \tag{3.18}$$

If $\delta \gg \max\limits_{i \in N} \mu_i$, 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 proposition as follows. We also observe that the inequality holds true in the various simulation settings considered next.

Proposition 3.4.4. *For a SRT-MIC system model with homogeneous users where all users have i.i.d. NBUE task workloads with mean μ and the same QoS requirement q, if the period length satisfies $1-\frac{\mu}{\delta} > q$, then for any NBUE workload distribution and for any $\epsilon > 0$ satisfying $1-(1+\epsilon)\frac{\mu}{\delta} > q$, there exists n', such that for all $n \geq n'$,*

$$m = \left\lceil \frac{nq\mu}{\delta - (1+\epsilon)\mu} \right\rceil$$

is a sufficient number of cores to meet the QoS requirement for n users.

By letting ϵ approach 0, the m in this proposition approaches $m^{\text{est}}_{\text{LDF+Greedy}}$. This is due to the law of large numbers and we omit the proof.

3.5 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.

3.5.1 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 $\text{Gamma}(5, 1)$ 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 3.6. We choose this NBUE workload distribution as a representative one.

In the bottom panel in Figure 3.6, 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 (3.15) (3.16), the upper bound on savings becomes,

$$1 - \frac{m}{m_{\mathrm{RB}}} \simeq 1 - \frac{q\mu}{w(q)}, \tag{3.19}$$

where μ is the common mean workload and $w(q)$ is the common required static allocation. Given Gamma$(5, 1)$ as the workload distribution, 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%.

3.5.2 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 3.7, 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

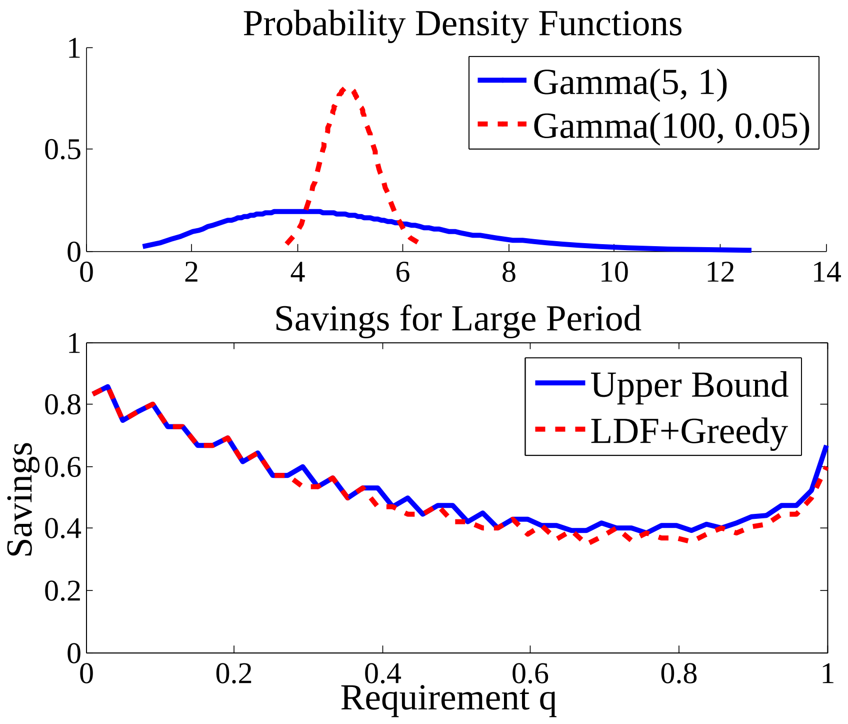

Figure 3.6: Top: the probability density functions for Gamma$(5, 1)$ and Gamma$(100, 0.05)$. Bottom: the resource savings for large period.

that the savings are monotonically decreasing in q, which is different from the "U" shape exhibited in Figure 3.6. Intuitively, this is because for deterministic workloads, by (3.19) we know $w(q)$ equals to μ and thus we get

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

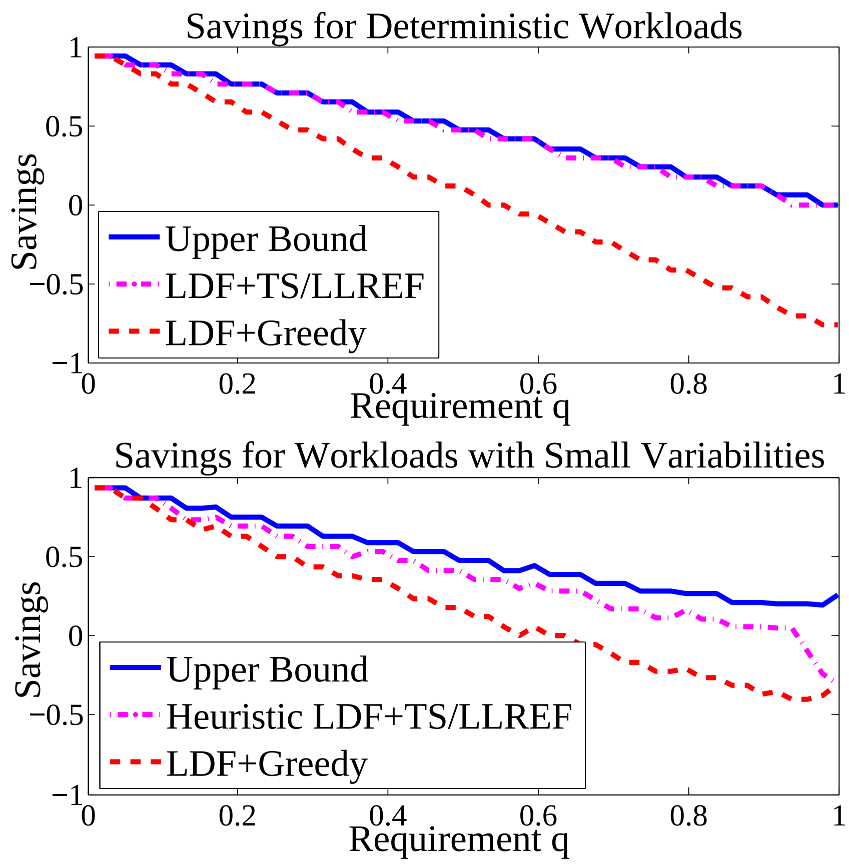

Figure 3.7: Top: the resource savings under deterministic workloads. Bottom: the resource savings under random workloads with small variability.

3.5.3 LDF+TS/LLREF for Workloads with Small Variability

For workloads with small variability, we envisage that the heuristic LDF+TS/LLREF described in Section 3.4.3 is a good non-clairvoyant policy. Consider an 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 3.6. 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 3.4.3. We conduct the same analysis for resource savings and exhibit the results in the bottom panel in Figure 3.7.

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} and that becomes a more critical problem as q becomes bigger. 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.

3.6 Possible Generalizations

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

Table 3.1: 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	$\{\mathbf{q} \in \mathbb{R}_+^n \mid \mathbf{q} \preceq \mathbf{1}, \sum_{i \in N} w_i(q_i) \leq m\delta, w_i(q_i) \leq \delta, \forall i \in N\}$	$\{\mathbf{q} \in \mathbb{R}_+^n \mid \mathbf{q} \preceq \mathbf{1}, \sum_{i \in N} q_i\mu_i \leq m\delta\}$	$1 - \frac{\max_{i \in N} \mu_i}{\delta}$		$1 - \frac{\max_{i \in N} \mu_i}{m\delta}$
Different speeds s_c	$\{\mathbf{q} \in \mathbb{R}_+^n \mid \mathbf{q} \preceq \mathbf{1}, B_n(\mathbf{q}) \leq S_m \cdot \delta, B_k(\mathbf{q}) \leq S_k \cdot \delta, 1 \leq k \leq m\}$	$\{\mathbf{q} \in \mathbb{R}_+^n \mid \mathbf{q} \preceq \mathbf{1}, \sum_{i \in N} q_i\mu_i \leq S_m \cdot \delta\}$	$1 - \frac{\max_{i \in N} \mu_i}{\bar{s} \cdot \delta}$	$1 - \frac{\max_{i \in N} \mu_i}{\min_{c \in C} s_c \cdot \delta}$	$1 - \frac{\max_{i \in N} \mu_i}{S_m \cdot \delta}$
Different periods δ_i	$\{\mathbf{q} \in \mathbb{R}_+^n \mid \mathbf{q} \preceq \mathbf{1}, \sum_{i \in N} \frac{w_i(q_i)}{\delta_i} \leq m, w_i(q_i) \leq \delta_i, \forall i \in N\}$	$\{\mathbf{q} \in \mathbb{R}_+^n \mid \mathbf{q} \preceq \mathbf{1}, \sum_{i \in N} \frac{q_i\mu_i}{\delta_i} \leq m\}$	N/A		$1 - \frac{\max_{i \in N} \frac{\mu_i}{\delta_i}}{m}$
Chains of subtasks $k(i)$	$\{\mathbf{q} \in \mathbb{R}_+^n \mid \mathbf{q} \preceq \mathbf{1}, \sum_{i \in N} w_i(q_i) \leq m\delta, w_i(q_i) \leq \delta, \forall i \in N\}$	$\{\mathbf{q} \in \mathbb{R}_+^n \mid \mathbf{q} \preceq \mathbf{1}, \sum_{i \in N} q_i\mu_i \leq m\delta\}$	$1 - \frac{\max_{i \in N} \mu_i}{\delta}$		$1 - \frac{\max_{i \in N} \mu_i}{m\delta}$

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.

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

For ease of reference, Table 3.1 provides a summary of various generalizations—the necessary notation is introduced in the sequel.

3.6.1 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., [17]. 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 $\overline{s} = \frac{\sum\limits_{c \in C} s_c}{m}$ 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 $n \geq m$ since otherwise one only needs the n fastest cores. Next we discuss generalizations of our results.

3.6.1.1 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 $1 \leq k \leq |Z|$, 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 $1 \leq k \leq n$, 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 [27, 28], we know that a static allocation is feasible if and only if the following conditions hold:

$$B_n(\mathbf{q}) \leq S_m \cdot \delta, \tag{3.20}$$

$$B_k(\mathbf{q}) \leq S_k \cdot \delta \text{ for } 1 \leq k \leq m. \tag{3.21}$$

Intuitively, (3.20) implies that the sum of required reservations does not exceed the total units of work that can be performed in a period. And (3.21) 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., [27, 28]. Therefore, the feasibility region of reservation-based static sharing F_{RB} is given by

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

This is consistent with our analysis when $s_c = 1$ for all $c \in C$, see Eq (3.3).

3.6.1.2 Outer Bound $R_{\mathbf{OB}}$ for the System Feasibility Region

For a system with different core processing speeds, the outer bound R_{OB} in Theorem 3.3.1 needs to be modified to

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

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 3.3.1: 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.

3.6.1.3 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[7] 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 3.4.2 we get the following corollary.

Corollary 3.6.1. *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*

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

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 the average processing speed of the k fastest cores is at least $\overline{s}$ for $1 \leq k \leq m$. Refer to Appendix 3.9.4 for the proof.

Non-Preemptive Greedy Task Scheduler: The non-preemptive greedy task scheduler starts by putting the task with the highest priority on

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

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 3.6.2. *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 Appendix 3.9.5 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.

3.6.1.4 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 [27, 28], 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 [28], a variant of LLREF for cores with different speeds, and Proportionate Fair (Pfair) [5].

Similar to the TS/LLREF task scheduler in Definition 3.4.3, we propose TS/U-LLREF or TS/Pfair where the task selection rule (3.14) naturally becomes

$$j(\mathbf{d}) = \max \Big\{ j | \sum_{i=1}^{j} \mu_{d_i} \leq S_m \cdot \delta \Big\}, \tag{3.22}$$

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

Under Assumption 1, and similarly to Theorem 3.4.3, 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 3.4.3 by simply replacing $m\delta$ with $S_m \cdot \delta$.

3.6.2 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,

$$X_i(t+1) = [X_i(t) + q_i \cdot \frac{\Delta}{\delta_i} - Y_i(t+1)]^+, \tag{3.23}$$

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.

3.6.2.1 Reservation-Based Static Sharing Policies

We first generalize the performance characterization of reservation-based static sharing policies. Similarly to the setting in 3.3.1, 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 $\sum\limits_{i\in N} \frac{w_i(q_i)}{\delta_i} > m$ we cannot meet the core time reservations $w_i(q_i)$ for all users. Indeed, by prior work, see e.g., [14, 17], we can characterize the feasibility region F_{RB} of reservation-based static sharing policies as follows,

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

Note that this is consistent with our analysis when all users have the same period, see Eq (3.3).

Given that $\sum\limits_{i \in N} \frac{w_i(q_i)}{\delta_i} \leq m$ 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. Refer to Appendix 3.9.8 for more details.

3.6.2.2 Outer Bound $R_{\mathbf{OB}}$ 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,

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

Intuitively, $\sum\limits_{i \in N} \frac{q_i \mu_i}{\delta_i}$ 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 3.3.1—refer to Appendix 3.9.6 for details.

3.6.2.3 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[8], 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 3.4.2. 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

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

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 3.4.3. 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 (3.14), 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,

$$j(\mathbf{d}) = \max\Big\{j \,|\, \sum_{i=1}^{j} \frac{\mu_{d_i}}{\delta_{d_i}} \leq m\Big\}. \tag{3.24}$$

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 3.4.3.

Corollary 3.6.3. *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*

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

Intuitively, under the task selection rule (3.24), 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 3.4.3 and we shall omit it.

Again, this result is consistent with our analysis when all users have the same period, see Theorem 3.4.3.

3.6.2.4 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.

3.6.3 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. By [60] we know user i's task workload W_i also has an NBUE distribution.

This generalized task model captures tasks that are completed in phases. For example, in the CRAN context each antenna generates a task associated with each subframe. A task may further consist of sub-tasks like encoding/decoding, modulation/demodulation, FFT/IFFT.

Suppose the system can observe the sub-task completions, these observations enable a broader range of non-clairvoyant resource allocation policies, which could potentially achieve better performance, i.e., a larger system feasibility region F. For example, now we can consider a resource allocation policy that stops processing a task if its first sub-task takes too long.

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. Next we shall discuss this in more detail.

3.6.3.1 Reservation-Based Static Sharing Designs

Given the sub-task workload distributions and the assumption of workload independence, we can get the workload distribution of W_i and thus $w_i(q_i)$ for all users $i \in N$. Therefore, the discussion of reservation-based static sharing policies in Section 3.3.1 still holds.

3.6.3.2 Outer Bound for the System Feasibility Region F

The definition of the outer bound region R_{OB} and Theorem 3.3.1 still holds, but the proof for Theorem 3.3.1 requires some modification. See Appendix 3.9.7 for the details.

3.6.3.3 LDF-Based System Designs

We can still use our proposed LDF-based resource allocation policies, i.e., LDF+Greedy and LDF+TS/LLREF, to process tasks consisting of sub-tasks. When applying these approaches, we consider each task as a whole task and do not use the sub-task information. This is reasonable because partially completing some sub-tasks does not help to meet the QoS requirements $\mathbf{q}$. Our performance characterization results Theorem 3.4.2, Theorem 3.4.3, etc., still hold.

As a summary, tasks consisting of sequences of sub-tasks with independent NBUE workloads do not change the results in this chapter.

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.

3.7 User Management for Computing Systems

In this dissertation we have taken a computing system to be either a cluster of machines or a centralized server with a large number of cores. A large-scale cloud-based infrastructure generally consists of several such systems, and thus requires a centralized user/stream management strategy to add new users to existing systems, and/or to move users across computing systems. In this section, we consider a cloud-based infrastructure consisting of several SRT-MIC systems with NBUE workloads and discuss the problem of how to allocate users across such possibly heterogeneous systems.

3.7.1 System Model for User Management

We consider a cloud-based infrastructure consisting of several SRT-MIC systems, each of which serves a set of users with NBUE workloads according to some optimized non-clairvoyant resource allocation policies, such as LDF+Greedy or LDF+TS/LLREF we proposed. The users generate tasks periodically, with the same period δ, as discussed in Section 3.2. Let L be the set of all SRT-MIC systems. Each SRT-MIC system $l \in L$ has $m^{(l)}$ identical cores. For simplicity, we assume different SRT-MIC systems have identical cores, but possible different numbers of cores. The set of users and the associated cardinality for system $l \in L$ is denoted by $N^{(l)}$ and $n^{(l)}$, respectively. The QoS requirement vector for system $l \in L$ is denoted by $\mathbf{q}^{(l)} = (q_1^{(l)}, q_2^{(l)}, \cdots, q_{n^{(l)}}^{(l)})$ where $q_i^{(l)}$ represents the requirement on timely task completion for user $i \in N^{(l)}$.

We can evaluate the performance of an SRT-MIC system based on measurements of the history outcomes. For example, for each SRT-MIC system $l \in L$ we can measure the achieved time-averaged[9] task completion vector $\mathbf{p}^{(l)} = (p_1^{(l)}, p_2^{(l)}, \cdots, p_{n^{(l)}}^{(l)})$ where $p_i^{(l)}$ represents the achieved time-averaged number of tasks completed on time per period for user $i \in N^{(l)}$. The SRT-MIC system $l \in L$ is called *feasible* if $p_i^{(l)} \geq q_i^{(l)}$ for all users $i \in N^{(l)}$. Here we assume all SRT-MIC systems are feasible. For each user $i \in N^{(l)}$ we can also measure the time-averaged time spent on that user per period, denoted by $t_i^{(l)}$.

Suppose a new user u is to be admitted and served by the cloud infrastructure and that its QoS requirement is q_u and mean workload μ_u. Our objective is to allocate this new user to a SRT-MIC system in such a way that we meet the user's QoS requirements without violating that of existing users.

Based on the discussion in this chapter, we have seen that it is very hard to accurately predict the impact of adding a new user to an SRT-MIC system. Moreover, in practice it could be challenging to get fine-grained information about workload distributions. In this section we propose a suboptimal user management policy that is based on monitoring the current performance of

[9]In practice we can measure the average over a reasonable time window.

the SRT-MIC systems.

3.7.2 Measurement-Based User Management Policy

We are motivated by the inequality (3.8) in the characterization of our outer bound feasibility region R_{OB}. By letting $S = \{i\}$ in (3.8) we know for each SRT-MIC system $l \in L$ and each user $i \in N^{(l)}$, we have

$$p_i^{(l)} \mu_i \leq t_i^{(l)}, \tag{3.25}$$

and the equality holds if user i generates tasks with exponential workloads in the continuous-time scenario (or geometric workloads in the discrete-time scenario). We define $\frac{t_i^{(l)} q_i^{(l)}}{p_i^{(l)}}$ to be the estimated required average time for user i to meet QoS requirement $q_i^{(l)}$.

Our policy consists of two parts: *feasibility checking*, to find SRT-MIC systems that could admit this new user, and *scoring*, to select one out of the feasible candidate SRT-MIC systems. A similar two-part approach is adopted in the Google Borg system [68] to allocate tasks, not necessarily soft real-time tasks, to machines.

In *feasibility checking*, we aim to find a set of SRT-MIC systems that have enough "resources" to admit the new user u. For each SRT-MIC system $l \in L$, we define the remaining available resources $r^{(l)}$ as follows,

$$r^{(l)} = m^{(l)}\delta - \sum_{i \in N^{(l)}} \frac{t_i^{(l)} q_i^{(l)}}{p_i^{(l)}}.$$

Intuitively, $\sum_{i \in N^{(l)}} \frac{t_i^{(l)} q_i^{(l)}}{p_i^{(l)}}$ represents the sum required time for users in $N^{(l)}$ to meet the QoS requirement $\mathbf{q}^{(l)}$ and $r^{(l)}$ represents the remaining time system l

can spend on additional users. All the quantities that we need here, i.e., $m^{(l)}$, δ, $q_i^{(l)}$, $p_i^{(l)}$ and $t_i^{(l)}$, are easy to obtain or measure from system l.

We mark the system $l \in L$ as a feasible candidate system if

$$r^{(l)} \geq q_u \mu_u + h,$$

where h is a non-negative safety guard. We use L^* to represent the set of candidate systems, i.e.

$$L^* = \{l | r^{(l)} \geq q_u \mu_u + h\}.$$

We introduce this safety guard h because there is generally a gap between $p_i^{(l)} \mu_i$ and $t_i^{(l)}$ in (3.25). In the sequel we will discuss how to adapt the value of h based on the “correctness” of our user allocation decisions.

In *scoring*, we compute a score for each candidate system $l \in L^*$ according to some scoring function $f(l)$ and pick the SRT-MIC system with the largest score. We may use different scoring functions for different purposes. For example, to enable a tight “packing” of users across the SRT-MIC systems, we could let $f(l) = -r^{(l)}$ which is equivalent to picking the system $l \in L^*$ with smallest remaining space $r^{(l)}$. That enables us to consolidate the SRT-MIC systems so as to shut down some SRT-MIC systems to save energy or to use those resources for other purposes. But under this scoring function, we may get penalized for even small errors in the estimations of h, μ_u, etc.

Alternatively, one can be conservative and let $f(l) = r^{(l)}$ which is equivalent to picking the system $l \in L^*$ with the largest remaining space $r^{(l)}$. Such a choice would result in the users' workloads being balanced across SRT-MIC systems. A potential problem for using this scoring function is that the available "space" be separated across SRT-MIC systems and we may not be able to find a candidate system for a new user with a large $q_u\mu_u$. Some other more complicated scoring functions may take into account the mixing of users with high QoS requirements and low QoS requirements. In practice we can design an appropriate scoring function based on the specific design requirements/objectives.

Once we allocate the new user u to an SRT-MIC system l according to the above policy, we check the correctness of this allocation by observing the achieved task completion vector and comparing it with the QoS requirements. If the system l is no longer feasible after adding user u, we remove user u from l and re-allocate the user according to the above policy. But this time we no longer mark system l as a candidate system for user u.

To realize efficient resource sharing we propose to adapt the safety guard h based on the correctness of our previous user allocation decisions. One might for example start with $h = \delta/2$ and we allocate a new user u to system l according to the policy above. If the user allocation decision turns out to be incorrect, that implies the value of h is too small and we double the value of h. Otherwise, we want to decrease the value of h to be more aggressive. Note that h only plays a role when $r^{(l)}$ is close to $q_u\mu_u$. If $r^{(l)} \gg q_u\mu_u$, the correctness of the decision is straightforward and does not provide much insight on whether h is a good value. Therefore, in such scenario we keep h unchanged. Formally, if $r^{(l)}$ is close to $q_u\mu_u$, by which we mean $q_u\mu_u + 2h \geq r^{(l)} \geq q_u\mu_u + h$, and the decision is correct, then we decrease the value of h by a small constant ϵ while guaranteeing h is non-negative; if the decision is incorrect, we double the value of h. In this way we dynamically adapt the value of h based on the correctness of user allocation decisions.

3.8 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.

3.9 Appendix

3.9.1 Proof of Theorem 3.4.1

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 $\boldsymbol{a} \circ \mathbf{b} = (a_1b_1, a_2b_2, \cdots, a_nb_n)$ the entrywise product.

Given $\mathbf{q} \in \operatorname{int}(R_{\mathrm{IB}})$, 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 $\mathbf{q}' = \mathbf{q} + \epsilon\mathbf{1} \in R_{\mathrm{IB}}$. By definition of R_{IB}, there exists a vector $\boldsymbol{\alpha} \succ \mathbf{0}$ such that for all $S \subseteq N$,

$$\sum_{i \in S} \alpha_i q'_i \leq \min_{\mathbf{d} \in D(S)} \sum_{i \in S} \alpha_i p_i(\mathbf{d}). \tag{3.26}$$

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

$$\begin{aligned}
&\mathrm{E}\left[L(\mathbf{X}(t+1)) - L(\mathbf{X}(t))|\mathbf{X}(t)=\mathbf{x}\right] \\
=\;& \mathrm{E}\left[\sum_{i=1}^{n}\alpha_i(X_i(t+1)^2 - X_i(t)^2)|\mathbf{X}(t)=\mathbf{x}\right] \\
\leq\;& \mathrm{E}\left[\sum_{i=1}^{n}\alpha_i((X_i(t)+q_i-Y_i(t+1))^2\right. \\
&\left. -X_i(t)^2)|\mathbf{X}(t)=\mathbf{x}\right] \\
=\;& \mathrm{E}\left[\sum_{i=1}^{n}\alpha_i(q_i-Y_i(t+1))^2\right. \\
&\left. +2\langle\boldsymbol{\alpha}\circ\mathbf{X}(t),\mathbf{q}-\mathbf{Y}(t+1)\rangle|\mathbf{X}(t)=\mathbf{x}\right] \\
\leq\;& \mathrm{E}\left[\sum_{i=1}^{n}\alpha_i(q_i^2+Y_i(t+1)^2)\right. \\
&\left. +2\langle\boldsymbol{\alpha}\circ\mathbf{X}(t),\mathbf{q}-\mathbf{Y}(t+1)\rangle|\mathbf{X}(t)=\mathbf{x}\right] \\
=\;& \mathrm{E}\left[\sum_{i=1}^{n}\alpha_i(q_i^2+Y_i(t+1)^2)\right. \\
&\left. +2\langle\boldsymbol{\alpha}\circ\mathbf{X}(t),\mathbf{q}'-\mathbf{Y}(t+1)\rangle|\mathbf{X}(t)=\mathbf{x}\right] \\
&-2\epsilon\langle\mathbf{x},\boldsymbol{\alpha}\rangle
\end{aligned} \tag{3.27}$$

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

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

By reordering users according to priorities, we get

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

By the LDF policy we know $x_{d_i} \geq x_{d_{i+1}}$. By (3.26) 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 (3.27)

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

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[10] 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.

[10]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 [15].

3.9.2 Lower Bound in Theorem 3.4.2 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 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

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

3.9.3 Proof of Theorem 3.4.3

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

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

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

The goal is to show $\gamma_2\mathbf{q} \in \mathrm{cl}(F_{\mathrm{LDF+TS/LLREF}})$. 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 3.4.1 it suffices to show $\gamma_2\mathbf{q} \in R_{\mathrm{IB}}$ 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 $\mathbf{d} \in D(S)$,

$$\sum_{i\in S} \mu_i p_i(\mathbf{d}) \geq \gamma_2 \sum_{i\in S} \mu_i q_i. \tag{3.28}$$

We show this in the following two cases.

If $\sum\limits_{i\in S} \mu_i \leq m\delta$, the task selection rule (3.14) 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 $\gamma_2 \leq 1$, we have $\sum\limits_{i\in S} \mu_i p_i(\mathbf{d}) = \sum\limits_{i\in S} \mu_i \geq \gamma_2 \sum\limits_{i\in S} \mu_i q_i$.

Otherwise, $\sum_{i\in S}\mu_i > m\delta$ and then not all users in S are selected. The task selection rule (3.14) 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,

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

This proves (3.28) and therefore,

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

3.9.4 Proof of Corollary 3.6.1

The proof is similar to that of Theorem 3.4.2. To avoid duplication here we only discuss the differences in the associated arguments. First, $m\delta$ should be replaced by $S_m \cdot \delta$. Second, instead of showing (3.12), one needs to show

$$\sum_{i\in S}\mathrm{E}[A_i(\mathbf{d})]\mu_i \le \frac{\max_{i\in N}\mu_i}{\bar{s}\cdot\delta}\,\mathrm{E}[T_S], \tag{3.29}$$

for which it suffices to show that

$$\sum_{i \in S} \mathrm{E}[A_i(\mathbf{d})] \leq \frac{\mathrm{E}[T_S]}{\overline{s} \cdot \delta}. \tag{3.30}$$

We still define $A_S(\mathbf{d}) = \sum_{i \in S} A_i(\mathbf{d})$. Under preemptive greedy task scheduling, if $A_S(\mathbf{d}) = k$ for $k = 0, 1, \cdots, m$, then there are k unfinished tasks on the k fastest cores, implying that the k fastest cores are busy processing tasks from users in S throughout the period. Therefore, $\sum_{i \in S} W_i \geq S_k \delta$ and thus $T_S \geq S_k \cdot \delta$. Clearly by the definition of S_k we know

$$S_1 \geq \frac{S_2}{2} \geq \cdots \frac{S_m}{m} = \overline{s}.$$

Therefore, $T_S \geq S_k \cdot \delta \geq k\overline{s}\delta$.

Thus it follows that

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

This proves (3.30) and concludes the proof.

3.9.5 Proof of Corollary 3.6.2

The proof is similar as that of Corollary 3.6.1. But this time, instead of showing (3.29), we shall show

$$\sum_{i \in S} \mathrm{E}[A_i(\mathbf{d})]\mu_i \leq \frac{\max_{i \in N} \mu_i}{\min_{c \in C} s_c \cdot \delta} \mathrm{E}[T_S], \tag{3.31}$$

for which it suffices to show that

$$\sum_{i \in S} \mathrm{E}[A_i(\mathbf{d})] \leq \frac{\mathrm{E}[T_S]}{\min\limits_{c \in C} s_c \cdot \delta}. \tag{3.32}$$

This is true because, if $A_S(\mathbf{d}) = k$ for $k = 0, 1, \cdots, m$, then there are k unfinished tasks, implying that there are k cores busy processing tasks from users in S throughout the period. Thus, $T_S \geq k \cdot \min\limits_{c \in C} s_c \cdot \delta$.

Thus it follows that

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

which proves (3.32).

3.9.6 Proof of $R_{\mathbf{OB}}$ when users generate tasks with different periods

The proof of this generalization is similar to that of Theorem 3.3.1. The main differences lie in the definitions of the random variables Y_i, A_i, E_i and U_S. In this setting, for each user i we define Y_i to be the random variable that represents the number of tasks completed on time over a typical super period Δ. For a feasible $\mathbf{q}$, by the Ergodic Theorem, we know $q_i \cdot \frac{\Delta}{\delta_i} \leq \mathrm{E}[Y_i]$ for all $i \in N$. We further define A_i to be the number of user i's unfinished tasks over a typical super period and define E_i to be the total residual workloads of user i's unfinished tasks over a typical super period. For each subset of users

$S \subseteq N$, we define U_S to be a random variable denoting the total core time spent on users in S in a typical super period. We can still get equation (3.5) and by $\mathrm{E}[E_i] \leq \mathrm{E}[A_i]\mu_i$ we can get that

$$\sum_{i \in N} q_i \cdot \frac{\Delta}{\delta_i} \cdot \mu_i \leq \sum_{i \in N} \mathrm{E}[Y_i]\mu_i \leq \mathrm{E}[U_N] \leq m \cdot \Delta.$$

Therefore,

$$\sum_{i \in N} \frac{q_i \mu_i}{\delta_i} \leq m.$$

3.9.7 Proof of $R_{\mathbf{OB}}$ under generalized sub-task model

In systems where each task consists of a sequence of sub-tasks, the definition of the outer bound region R_{OB} and Theorem 3.3.1 still holds, but the proof for Theorem 3.3.1 requires some modification, specifically (3.6) in the proof no longer holds.

Recall that in the proof of Theorem 3.3.1 we want to show $\mathrm{E}[E_i] \leq \mu_i \, \mathrm{E}[A_i]$ for all users i, where $\mathrm{E}[E_i]$ is the mean residual workload of user i's unfinished tasks and $\mathrm{E}[A_i]$ is the mean number of user i's unfinished tasks. Our approach is to define $A_{i,c}$ to be the indicator random variable that user i's task is unfinished and is processed for c time units in a typical period. By total probability we have that

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

Under the original SRT-MIC system model where $k(i) = 1$, by NBUE property $\mathrm{E}[E_i | A_{i,c} = 1] = \mathrm{E}[W_i - c | W_i > c] \leq \mu_i$ and that enables us to show $\mathrm{E}[E_i] \leq$

$\mu_i \,\mathrm{E}[A_i]$.

However, under this generalized task model, $\mathrm{E}[E_i|A_{i,c} = 1]$ may no longer equal to $\mu_{i,c} = \mathrm{E}[W_i - c|W_i > c]$. This is because in some resource allocation policies, the event $A_{i,c} = 1$ could give more information than $W_i > c$. For example, suppose user i generates tasks with two sub-tasks, i.e., $k(i) = 2$. Consider a policy that always finishes user i's sub-task 1 and then stops. Suppose the period length δ is large enough to complete user i's sub-task 1. In this scenario we know for all c, $\mathrm{E}[E_i|A_{i,c} = 1] = \mathrm{E}[W_i^{(2)}]$ which may not equal to $\mu_{i,c}$.

Next we shall show $\mathrm{E}[E_i] \leq \mu_i \,\mathrm{E}[A_i]$ is still true under the generalized task model for a user i with $k(i) = 2$. The proof can be easily extended to general $k(i)$. We define $I_i^{(1)}$ to be the indicator random variable that sub-task 1 from user i completes in a typical period. By total probability we have that

$$\begin{aligned}
&\mathrm{E}[E_i|A_{i,c} = 1] \qquad (3.33)\\
&= \mathrm{E}[E_i|A_{i,c} = 1, I_i^{(1)} = 1] \Pr(I_i^{(1)} = 1|A_{i,c} = 1)\\
&\quad + \mathrm{E}[E_i|A_{i,c} = 1, I_i^{(1)} = 0] \Pr(I_i^{(1)} = 0|A_{i,c} = 1).
\end{aligned}$$

Given that $I_i^{(1)} = 1$, the residual workload E_i is only the remaining workload of sub-task 2 and by the NBUE property of sub-task 2, we know

$$\mathrm{E}[E_i|A_{i,c} = 1, I_i^{(1)} = 1] \leq \mu_i^{(2)} \leq \mu_i.$$

Similarly, if $I_i^{(1)} = 0$, then E_i is the sum of the remaining workload of sub-task 1, and the whole workload of sub-task 2 which is independent of the event

$A_{i,c} = 1$. Therefore, by the NBUE property of sub-task 1, we have that

$$\begin{aligned}
\mathrm{E}[E_i | A_{i,c} = 1, I_i^{(1)} = 0] &\leq \mu_i^{(1)} + \mathrm{E}[W_i^{(2)} | A_{i,c} = 1, I_i^{(1)} = 0] \\
&= \mu_i^{(1)} + \mu_i^{(2)} = \mu_i.
\end{aligned}$$

Now by (3.33) we get that

$$\begin{aligned}
\mathrm{E}[E_i | A_{i,c} = 1] &\leq \mu_i \Pr(I_i^{(1)} = 1 | A_{i,c} = 1) \\
&\quad + \mu_i \Pr(I_i^{(1)} = 0 | A_{i,c} = 1) \\
&\leq \mu_i.
\end{aligned}$$

Therefore,

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

The other part of the proof of Theorem 3.3.1 remains unchanged, and thus, our discussion of R_{OB} still holds.

3.9.8 Achieving $F_{\mathbf{RB}}$ via LLREF scheduling

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 challenge is how to allocate $w_i(q_i)$ to each user i in each period. We convert this to the following equivalent hard real-time scheduling problem. Consider a system where each user $i \in N$ periodically generates tasks with period δ_i and deterministic task workload $w_i(q_i)$. The tasks are available for processing at the beginning of periods and need to be completed by the end of the corresponding periods. The objective is to schedule these tasks on m identical cores to guarantee that all tasks complete on time without exception. One solution is to use the LLREF scheduling policy which always gives a feasible schedule if it is possible. In Section 3.4.3 we have introduced LLREF policy when users have the same periods. Next we introduce how to apply LLREF to solve this hard real-time scheduling problem where users generate tasks with different periods.

LLREF divides the timeline into intervals by task releases/deadlines. In each interval of length τ, the *local workload* of each user $i \in N$ is defined as $\frac{\tau}{\delta_i} w_i(q_i)$. Therefore, to complete all tasks on time it suffices to complete the local workloads of all users in each interval. To achieve that, in each interval we adopt the LLREF policy introduced in Definition 3.4.2 to process local workloads for all users. This LLREF policy solves the hard real-time scheduling problem.

By adopting this policy we can get a static time allocation such that each user $i \in N$ gets core time reservation $w_i(q_i)$ in each period, which further guarantees that each user i meets the QoS requirement q_i.

Chapter 4

Conclusion and Challenges for Cloud-Based Infrastructure

Cloud-based computing infrastructure can be efficient to support soft real-time applications with compute/communicate deadlines and QoS requirements. In this dissertation, we consider the problem of compute resource allocation in such cloud-based systems and our main contribution is a theoretical study on the efficiency and optimality of deficit-based resource allocation policies.

Beyond the resource allocation problem, there are many challenges that infrastructure providers face as they consider evolving towards a cloud-based architecture. One important application of cloud-based infrastructure supporting real-time traffic is the Radio Access Networks (RAN). In this chapter, we focus on the RAN context and discuss some of the key technical insights and challenges associated with this possible evolution of cloud-based architecture.

4.1 Current Trends: Cloud-Based Radio Access Network (CRAN)

In order to handle the exponential growth in traffic associated with delivering content (video, images, data) to and from mobile devices, operators are relying on network densification leveraging a heterogeneous mix of macro/micro/femto cells along with Wi-Fi access points to enhance network capacity where it is most needed. The next generation (5G) wireless technologies currently under study further take aim at this problem through enhanced massive MIMO physical layer technologies, and the short range but high capacity mmWave band. At the core these are based on densification and increasingly demanding computation to realize the baseband processing associated with achieving the targeted 1000x increases in capacity.

Whether increasing capacity through densification or through the design of higher capacity physical layer standards one ends up with a massive deployment of compute resources dedicated to baseband processing necessary to enable wireless connectivity. Indeed, it is estimated that the amount of compute resources associated with today's wireless infrastructure far exceeds that of current cloud compute services [29]. Perhaps disappointingly such infrastructure is poorly utilized. It is not unusual for a base station's average load to be no more than 20%. With densification one might see even lower utilizations, since each base station will aggregate traffic from smaller regions/number of users, potentially leading to increased burstiness in load and lower utilization.

Needless to say that the capital and operational costs (CAPEX/OPEX) of such wireless deployments are high. These are not simply those of the base stations, access points, antennas, compute resources, fronthaul/backhaul, but also include the housing, cooling, maintenance and powering of such an extensive and distributed infrastructure. By contrast cloud based computational/storage services have, for the most part, leveraged consolidation of resources enabling extremely cost efficient centralized resource management. One exception to this would be content delivery networks (CDNs) whose focus on caching content at the network "edge" aims to reduce backhauling costs and improve performance by serving users' requests from servers which are closer by. Given the enormous infrastructure investments that wireless infrastructure operators have and will be making, there has been substantial interest in attempting to replicate the cloud compute model in this setting.

In other words, the challenge is to develop the technologies needed to achieve Cloud-based Radio Access Network (CRAN), where the radio resources are distributed and baseband signal processing are centralized. A simplified CRAN architecture is exhibited in Figure 4.1. Broadly speaking such an infrastructure would make use of co-located shared compute resources to meet the baseband processing needs of close by radio resources. Such sharing might be realized through virtualization of compute resources enabling improved flexibility and higher reliability. In this setting the degree of centralization is limited by the relatively high bandwidth baseband signals that need to transported from spatially distributed antennas to the cloud and the tight processing constraints. Such limitations may however provide a further opportunity to also aggregate additional computational workloads associated with other services requiring computation close to users, e.g., latency sensitive and/or real-time processing associated with control, transportation, power networks. An alternative approach to overcome such limitations is to consider flexibly partitioning processing tasks across compute resources at the radio resources and in the cloud.

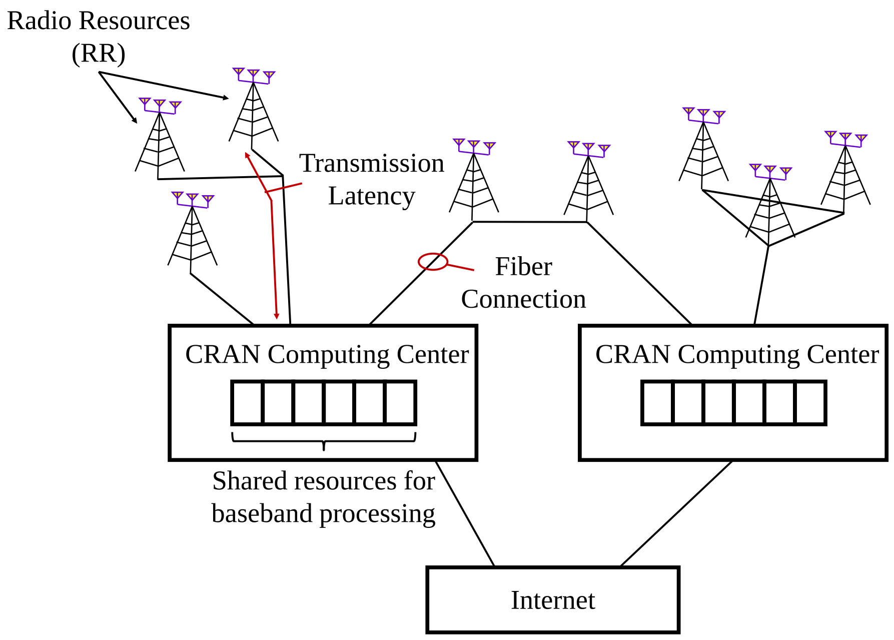

Figure 4.1: The CRAN architecture.

The goal of this chapter is to explore opportunities and challenges, that wireless infrastructure providers face as they consider evolving towards a more cloud-based architecture.

4.2 Technical Challenges

In this section we identify and discuss key technical challenges to achieve CRAN in various areas:

- Scale of aggregation and fronthaul network.
- Cloud-enabled applications/techniques.
- Virtualizing real-time processing in the centralized computing center.

4.2.1 Scale of Aggregation and Fronthaul Network

In order to virtualize compute resources for baseband processing, the received uplink signals associated with wireless subframes are sampled and sent from radio resources to the cloud for timely decoding and processing such that downlink signals requiring timely channel measurements, acknowledgements, etc., can be sent back to the radio resources for transmission. Cellular system standards require this process to be carried out within several milliseconds. While compute resource sharing enables hardware resource savings in CRAN, it poses unique challenges in making the right decision with respect to the degree of centralization and the design of the fronthaul network.

Challenge 1: How much centralization is possible/desirable?

The processing workload associated with each radio resource typically varies from subframe to subframe due to time-varying user activity, user movement and channel quality. Intuitively, a larger degree of centralization enables more resource consolidation and thus more savings to acquire from statistical multiplexing of compute resources. By aggregating close-by radio resources with independent or negatively correlated workloads, e.g., due to user movement within a small area, one can realize the multiplexing gains and share the compute resources more efficiently. Moreover on a larger time scale, recent studies, e.g. [13], have indicated that the users in cellular systems generally move in a spatial-temporal pattern. For example, in dense urban areas the users move from residential areas to business areas in the morning and back in the late afternoon. Consequently, the aggregate processing workloads associated with base stations experience large long-term and predictable variations throughout the day, i.e., the "tidal effect". If the spatial scale of aggregation, i.e., the coverage area of a centralized infrastructure, is large and covers the "tidal effect", one can consolidate the resources in different areas, e.g., the residential and business areas, and achieve higher savings. These observations are exhibited in Figure 4.2. At the top of Figure 4.2, we show the workloads of several radio resources, and the total workloads of the business and residential area during a day. Due to the short-term workload variations associated with radio resources and the "tidal effect", it can be seen that a larger spatial scale of aggregation implies higher multiplexing gains.

However, given the stringent real-time deadline associated with baseband signal processing and transmission, the spatial scale of aggregation is limited by the transmission latency to/from the cloud. A computing center that covers a larger area requires more time to transmit signals back and forth,

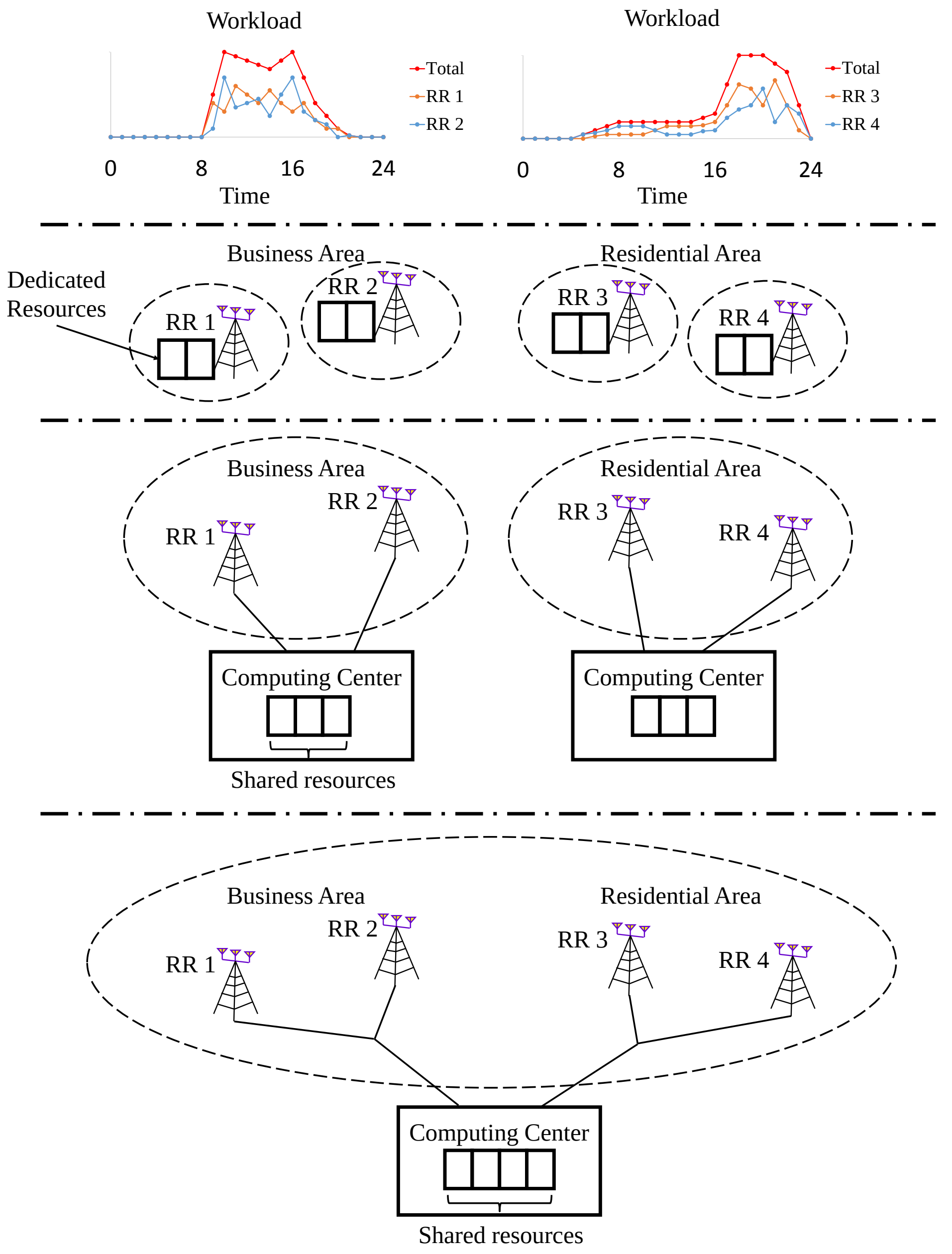

Figure 4.2: Different scale of aggregation implies different resource savings. In traditional RAN, each radio resource requires 2 units of compute resources. If one can aggregate the workloads for the business and residential area, respectively, 3 units of compute resources are required for each computing center. Moreover, if one aggregates the business and residential area, further savings can be achieved.

and therefore leaves less time for the processing in the cloud. This effectively requires the processors in the cloud to be faster and also makes it harder to share compute resources efficiently. The fronthaul/backhaul network cost in terms of optical fiber deployment and maintenance, energy consumption, etc., is another factor that impacts the desirable spatial scale of aggregation. Advanced techniques being considered for the next generation cellular system, e.g., mmWave transmission/reception and massive MIMO with high densities of radio resources, may dramatically increase the generated baseband signals in each subframe, and thus the transportation cost, making it possibly hard to afford a large scale of aggregation.

Challenge 2: How should one architect the fronthaul network?

In CRAN, the fronthaul network, which connects distributed radio resources to the associated computing centers, is required to be reliable and fault-tolerant, i.e., able to provide an alternative route for baseband signal transmission in case of connection breaks. It is also desirable to design the fronthaul network to reduce the deployment and maintenance cost. Different fronthaul topologies, e.g., star vs. tree, ring topologies, typically result in different costs and transmission delays.

Furthermore, if the spatial scale of aggregation is small as compared to the scale in which the long-term "tidal effects" occur, it is potentially beneficial to consider dynamic radio resource association policies permitting one to dynamically re-route the traffic associated with radio resources to the appropriate cloud centers in order to track and exploit the workload movement in the system and further save resources. This poses an interesting question as to whether flexible fronthaul networks provide significant cost savings.

The need for high-speed low-latency fronthaul networks is perhaps the Achilles's heel of the CRAN architecture. Indeed this has affected the markets where operators find CRAN architecture attractive, e.g., those where government encourages fiber deployment or there is already significant dark fiber available, e.g., China, South Korea and Japan. That said the benefits and flexibility of moving to a cloud based RAN on commodity hardware may outweigh these issues.

Challenge 3: To what degree can compression reduce fronthaul costs?

The high performance requirements of fronthaul connections, in terms of capacity, reach and especially latency, mandate point-to-point fiber connection between radio resources and the computing centers via the standard Common Public Radio Interface (CPRI). The commonly deployed 1Gbps backhaul links cannot be reused to support CPRI and thus, the fronthaul network and the associated baseband signal transmission accounts for a large fraction of cost in CRAN. An opportunity/challenge that arises is to develop advanced baseband signal compression techniques aimed at dramatically decreasing the amount of baseband signals that need to be transmitted and thus the associated transmission cost.

Still, CPRI is compatible with several other common physical layer standards and helps keep the network cost down in the future. New technologies, e.g., transmission over microwave, might also substantially reduce these costs.

4.2.2 Cloud-Enabled Applications/Techniques

Due to the centralized processing of baseband signals associated with disparate radio resources, CRAN facilitates the cooperation amongst distributed radio resources and provides great opportunities for advanced coordination and control techniques.

Challenge 4: What are the advantages/challenges of facilitating Coordinated Multi-Point (CoMP) transmission/reception in CRAN?

There is continuous interest in realizing high per-user throughput, particularly for edge users. CoMP techniques provide a path to address this problem. This can be particularly advantageous when leveraging dense but spatially distributed radio resources. By clustering neighboring radio resources into Virtual Base Stations (VBSs) and using cooperation amongst them to reduce or eliminate mutual interference, edge users, which were traditionally poorly served, can become "central" to one or more VBSs, i.e., are well situated relative to their serving VBSs. Ideally, if there is enough freedom in choosing VBSs each user could be "central". The centralization of computation in CRAN would certainly facilitate such dynamic use of CoMP transmission/reception.

Given the benefits of CoMP and assuming no computational constraints, one could in principle coordinate across all radio resources connected to the same computing center leveraging spatial diversity (in antennas) and removing all interference. However, previous work, e.g. [47], has shown that coordination across a large collection of radio resources does not bring as much benefit as expected because of measurement and signaling overheads. In practice cooperation is only possible, or desirable, for VBSs of limited size.

If we cluster radio resources into static sets of VBSs there may still be users at the edge of neighboring VBSs suffering from interference. To realize "no-edge" wireless networks it is thus necessary to dynamically use different VBSs in different sub-bands (or at different times) guaranteeing that each user can be "central" to a VBS. Advanced power control and user selection strategies are also essential to reap the benefits of CoMP to deliver good performance to every user. Some preliminary related work include [26, 31, 37, 46, 51, 56, 63, 69, 70].

Challenge 5: Can CRAN better manage heterogeneous networks consisting of macro, femto and small cells?

To meet the surging mobile traffic demands, multiple cellular standards, including GSM, WCDMA, LTE, etc., have been developed and coexist in current cellular network. In parallel, Wi-Fi and femtocell networks are also deployed but in a more distributed and self-organized manner. Similarly to the macro cell aggregation, these small cells may have their own centralized hub, which may or may not be connected to the macro cell computing center. The management of such a heterogeneous and multi-standard network is challenging.

The centralized infrastructure in CRAN provides a way to coordinate and mitigate the mutual interference. For example, in a system where femto cells and macro cells coexist, one may decide to turn off the macro cells in a small area for a short time to mitigate the interference between femto and macro cells.

Challenge 6: How can service providers leverage CRAN/SDN flexibility?

A benefit of cloud-computing is the abstraction it gives system administrators to enable centralized control of the shared resources and the network to meet application/service requirements. This is consistent with the trend towards Software Defined Networking (SDN). For example, in the CRAN context, given the "tidal effect" in a city, the administrators could config the behavior of the network, e.g., the dynamic radio resource association policy and the scheduling algorithm, on a daily basis to track the movement of the workloads. But if there is a football game or a concert in the city which causes traffic boost in a small area, the administrators can have the power of managing the resources for the event. Advanced research on SDN in cellular systems has the potential to adapt to dynamic traffic changes through flexible control.

4.2.3 Virtualizing Real-Time Processing in Computing Centers

The soft real-time requirements of baseband signal processing and the virtualization/abstraction of compute resources pose significant challenges on the design of the shared computing centers. This dissertation is devoted to the design of compute resource allocation policies, i.e., dynamically allocating compute resources and processing capacity to the baseband processing associated with radio resources taking into account the dynamic workloads, different QoS requirements, the criticality of the workloads, etc. Beyond that, the challenges in the design of hardware architecture, e.g., making the right tradeoff between flexibility and efficiency in the processor architecture, although beyond the scope of this dissertation, are also essential to realizing efficient utilization of the compute resources.

4.3 Outlook

There has been substantial interest in building cloud-based computing centers to deal with applications with real-time constraints and reduce costs associated with network construction and maintenance. Some of the challenges to enable such an evolution are highlighted in this chapter. This dissertation has been focused on the analysis and design of compute resource allocation policies, which fits in this vision as one of the fundamental questions on how to manage and support soft real-time applications in cloud-based infrastructure.

Bibliography

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

[2] Ali Allahverdi and Yuri Sotskov. Two-machine flowshop minimum-length scheduling problem with random and bounded processing times. *International Transactions in Operational Research*, 10:65–76, 2003.

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

[4] Alia Atlas and Azer Bestavros. Statistical Rate Monotonic Scheduling. In *Proceedings of RTSS 1998*, pages 123–132, December 1998.

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

[6] Carlos Bernardos, Antonio La Oliva, Pablo Serrano, Albert Banchs, Luis Miguel Contreras, Hao Jin, and Juan Carlos Zúñiga. An architecture for software

defined wireless networking. *IEEE Wireless Communications*, 21(3):52–61, 2014.

[7] Guillem Bernat and Alan Burns. Combining $\binom{n}{m}$-Hard deadlines and Dual Priority Scheduling. In *Proceedings of RTSS 1997*, pages 46–57, December 1997.

[8] David Blackwell. An Analog of the minimax theorem for vector payoffs. *Pacific Journal of Mathematics*, 6(1):1–8, November 1956.

[9] J. Blazewicz, M. Drabowski, and J. Weglarz. Scheduling multiprocessor tasks to minimize schedule length. *IEEE Transactions on Computers*, C-35:389–393, 1986.

[10] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge university press, 2009.

[11] J. Bruno, E. G. Coffman Jr., and R. Sethi. Scheduling independent tasks to reduce mean finishing time. *Communications of the ACM*, 17:382–387, 1974.

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

[13] China Mobile. C-RAN The Road Towards Green RAN, Oct 2011.

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

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

[16] J.G. Dai and Balaji Prabhakar. The throughput of data switches with and without speedup. In *Proceedings of INFOCOM 2000*, pages 556–564, March 2000.

[17] Robert I. Davis and Alan Burns. A Survey of Hard Real-Time Scheduling for Multiprocessor Systems. *ACM Computing Surveys*, 43, October 2011.

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

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

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

[21] D. Down and S.P. Meyn. Piecewise linear test functions for stability and instability of queueing networks. *Queueing Systems*, 27:205–226, April 1997.

[22] Yuhuan Du and Gustavo de Veciana. Efficiency and Optimality of Largest Deficit First Prioritization: Resource Allocation for Real-Time Applications. *INFOCOM 2016*. In submission.

[23] Yuhuan Du and Gustavo de Veciana. Efficiency and Optimality of Largest Deficit First Prioritization: Resource Allocation for Real-Time Applications. *ACM Transactions on Modeling and Performance Evaluation of Computing Systems*. In submission.

[24] Yuhuan Du and Gustavo de Veciana. Scheduling for Cloud-Based Computing Systems to Support Soft Real-Time Applications. *INFOCOM 2016*. In submission.

[25] Yuhuan Du and Gustavo de Veciana. Scheduling for Cloud-Based Computing Systems to Support Soft Real-Time Applications. *ACM Transactions on Modeling and Performance Evaluation of Computing Systems*. In submission.

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

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

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

[29] Alan Gatherer. Personal communication.

[30] Moncef Hamdaoui and Parameswaran Ramanathan. A Dynamic Priority Assignment Technique for Streams with (m, k)-Firm Deadlines. *IEEE Transactions on Computers*, 44:1443–1451, December 1995.

[31] Mingyi Hong, Ruoyu Sun, Hadi Baligh, and Zhiquan Luo. Joint base station clustering and beamformer design for partial coordinated transmission in heterogeneous networks. *IEEE Journal on Selected Areas in Communications*, 31, February 2013.

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

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

[34] I-Hong Hou and P. R. Kumar. Scheduling Heterogeneous Real-Time Traffic over Fading Wireless Channels. *IEEE/ACM Transactions on Networking*, 22:1631–1644, October 2014.

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

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

[37] Robert W. Heath Jr., Tao Wu, Young Hoon Kwon, and Anthony C. K. Soong. Multiuser mimo in distributed antenna systems with out-of-cell interference. *IEEE Transactions on Signal Processing*, 59(10), 2011.

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

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

[40] Joseph Y.-T. Leung. A New Algorithm for Scheduling Periodic, Real-Time Tasks. *Algorithmica*, 4:209–219, June 1989.

[41] Zukui Li and Marianthi Ierapetritou. Process scheduling under uncertainty: Review and challenges. *Computers and Chemical Engineering*, 32:715–727, 2008.

[42] C. L. Liu and James W. Layland. Scheduling Algorithms for Multiprogramming in a Hard Real-Time Environment. *Journal of the ACM*, 20(1):46–61, January 1973.

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

[44] Jane W. S. Liu. *Real-Time Systems*. Prentice Hall, April 2000.

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

[46] Xin Liu, Edwin K.P. Chong, and Ness B. Shroff. A framework for opportunistic scheduling in wireless networks. *Computer Networks*, 41:451–474, March 2003.

[47] Angel Lozano, Robert W. Heath Jr., and Jeff G. Andrews. Fundamental limits of cooperation. *IEEE Transactions on Information Theory*, PP:1, March 2013.

[48] Jason Mars, Lingjia Tang, Robert Hundt, Kevin Skadron, and Mary Lou Soffa. 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*, 2011.

[49] Nicholas McKeown. Scheduling Algorithms for Input-Queued Cell Switches. 1995. Ph.D. dissertation.

[50] Nick McKeown, Adisak Mekkittikul, Venkat Anantharam, and Jean Walrand. Achieving 100% Throughput in an Input-Queued Switch. *IEEE Transactions on Communications*, 47(8):1260–1267, August 1999.

[51] W. Mennerich and W. Zirwas. User centric coordinated multi point transmission. In *Proc. IEEE 72nd Vehicular Technology Conference Fall (VTC 2010-Fall)*, pages 1–5, September 2010.

[52] S. P. Meyn and R. L. Tweedie. *Markov Chains and Stochastic Stability*. Cambridge University Press, 2008.

[53] A. Müller and D. Stoyan. *Comparison Methods for Stochastic Methods and Risks*. Wiley, March 2002.

[54] Sirajum Munir, Shan Lin, Enamul Hoque, S. M. Shahriar Nirjon, J. A. Stankovic, and K. Whitehouse. Addressing Burstiness for Reliable Communication and Latency Bound Generation in Wireless Sensor Networks. In *IPSN 2010*, pages 303–314, April 2010.

[55] M. J. Neely. Delay Analysis for Max Weight Opportunistic Scheduling in Wireless Systems. *IEEE Transactions on Automatic Control*, 54:2137–2150, September 2009.

[56] E. Pateromichelakis, M. Shariat, A. ul Quddus, and R. Tafazolli. On the evolution of multi-cell scheduling in 3gpp lte / lte-a. *IEEE Communications Surveys and Tutorials*, 15:701–717, May 2013.

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

[58] Michael L. Pinedo. *Scheduling: Theory, Algorithms, and Systems.* Springer, 2012.

[59] Lui Sha, Tarek Abdelzaher, Karl-Erik ÅRZÉN, Anton Cervin, Theodore Baker, Alan Burns, Giorgio Buttazzo, Marco Caccamo, John Lehoczky, and Aloysius K. Mok. Real Time Scheduling Theory: A Historical Perspective. *Real-Time Systems*, 28:101–155, November-December 2004.

[60] Moshe Shaked and J. George Shanthikumar. *Stochastic Orders.* Springer, 2007.

[61] Sanjay Shakkottai and R. Srikant. Scheduling Real-Time Traffic With Deadlines over a Wireless Channel. *Wireless Networks*, 8:13–26, January 2002.

[62] Sanjay Shakkottai and Alexander L. Stolyar. Scheduling algorithms for a mixture of real-time and non-real-time data in HDR. In *Proceedings of the International Teletraffic Congress*, pages 793–804, 2001.

[63] A. L. Stolyar and H. Viswanathan. Self-organizing dynamic fractional frequency reuse for best-effort traffic through distributed inter-cell coordination. In *Proceedings of INFOCOM 2009*, pages 1287–1295, April 2009.

[64] Alexander L. Stolyar. Maxweight Scheduling in a Generalized Switch: State Space Collapse and Workload Minimization in Heavy Traffic. *The Annals of Applied Probability*, 14(1), February 2004.

[65] Leandros Tassiulas and Anthony Ephremides. Stability Properties of Constrained Queueing Systems and Scheduling Policies for Maximum Throughput in Multihop Radio Networks. *IEEE Transactions on Automatic Control*, 37:1936–1948, December 1992.

[66] Leandros Tassiulas and Anthony Ephremides. Dynamic Server Allocation to Parallel Queues with Randomly Varying Connectivity. *IEEE Transactions on Information Theory*, 39(2):466–478, March 1993.

[67] P.M. van de Ven, S.C. Borst, and L. Ying. Inefficiency of MaxWeight scheduling in spatial wireless networks. *Computer Communications*, 36:1350–1359, July 2013.

[68] Abhishek Verma, Luis Pedrosa, Madhukar Korupolu, David Oppenheimer, Eric Tune, and John Wilkes. Large-scale cluster management at Google with Borg. In *Proceedings of EuroSys 2015*, 2015.

[69] G. Wunder, M. Kasparick, A. Stolyar, and H. Viswanathan. Self-organizing distributed inter-cell beam coordination in cellular networks with best effort traffic. *8th International Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks (WiOpt 2010)*, pages 295–302, June 2010.

[70] Taesang Yoo and Andrea Goldsmith. On the optimality of multiantenna broadcast scheduling using zero-forcing beamforming. *IEEE Journal on Selected Areas in Communications*, 24, March 2006.

Vita

Yuhuan Du is a PhD candidate in department of Electrical and Computer Engineering at The University of Texas at Austin. He received his B.E. degree in Electronics Engineering from Tsinghua University in China in 2011, and his M.S.E. degree from The University of Texas at Austin in 2013. He has been working under the supervision of Prof. Gustavo de Veciana at The University of Texas at Austin since 2011. He interned at Microsoft and Yelp during summer 2012 and 2014, respectively.

Permanent email: dyhuan123@gmail.com

This dissertation was typeset with LaTeX[†] by the author.

[†]LaTeX is a document preparation system developed by Leslie Lamport as a special version of Donald Knuth's TeX Program.