Winter 12-15-2021

Improving Resource Efficiency in Cloud Computing

Jiayi Song
Washington University in St. Louis

WASHINGTON UNIVERSITY IN ST.LOUIS

McKelvey School of Engineering
Department of Computer Science and Engineering

Dissertation Examination Committee:
Roch Guérin, Chair
Sanjoy Baruah
Roger Chamberlain
Sanmay Das
Peter Key
Henry Sariowan
William Yeoh

Improving Resource Efficiency in Cloud Computing

by
Jiayi Song

A dissertation presented to
The Graduate School
of Washington University in
partial fulfillment of the
requirements for the degree
of Doctor of Philosophy

May 2022
St. Louis, Missouri

Table of Contents

List of Figures

List of Tables

Acknowledgments

I would like to express my sincere gratitude to many individuals and institutions, whose generous support and help have led me to successfully finish this dissertation. I would like to first express my sincere gratitude to my PhD academic advisor Prof. Roch Guérin for his invaluable advice, continuous support, and patience during my PhD study. He not only guides me to think deeply and rigorously, but also shows me the importance of integrity, diligence, passion, and consideration. Next I would like to extend my sincere thanks to all the staff and faculty members in our department of Computer Science and Engineering for providing a friendly environment and abundant resources for my study and research. Furthermore, I am deeply grateful to Prof. Baruah, Prof. Yeoh, Prof. Chamberlain, Prof. Das, Dr. Sariowan and Dr. Key for being my dissertation committee. Last, I would like to offer my special thanks to my parents and all my friends for their companion and support in my graduate study journey.

Jiayi Song

Washington University in Saint Louis

May 2022

Dedicated to my parents, for their encouragement, love and support.

ABSTRACT OF THE DISSERTATION

Improving Resource Efficiency in Cloud Computing

by

Jiayi Song

Doctor of Philosophy in Computer Science

Washington University in St. Louis, 2022

Professor Roch Guérin, Chair

Customers inside the cloud computing market are heterogeneous in several aspects, *e.g.,* willingness to pay and performance requirement. By taking advantage of trade-offs created by these heterogeneities, the service provider can realize a more efficient system. This thesis is concerned with methods to improve the utilization of cloud infrastructure resources, and with the role of pricing in realizing those improvements and leveraging heterogeneity. Towards improving utilization, we explore methods to optimize network usage through traffic engineering. Particularly, we introduce a novel optimization framework to decrease the bandwidth required by inter-data center networks through traffic scheduling and shaping, and then propose algorithms to improve network utilization based on the analytical results derived from the optimization. When considering pricing, we focus on elucidating conditions under which providing a mix of services can increase a service provider's revenue. Specifically, we characterize the conditions under which providing a "delayed" service can result in a higher revenue for the service provider, and then offer guidelines for both users and providers.

Chapter 1

Introduction

1.1 Motivation

Cloud computing has experienced explosive growth and become the computing platform of choice for an increasingly diverse set of users. By judiciously taking advantage of this growing diversity, *e.g.,* differences in job requirements, tolerance to delay and interruption, job duration, data location, etc., the service provider can realize a more efficient system. In fact, most service providers, like Amazon[1] , Google Cloud[2] and Microsoft Azure[3], offer multiple options to accommodate customers' heterogeneity. Among these services, reserved, on-demand and spot instances[4] are of most relevance to this work.

All reserved, on-demand and spot instances let users dynamically subscribe compute capacity at certain time granularities. Reserved and on-demand instances are similar, except that

[1]Retrieved 2022, Jan 2, from `https://aws.amazon.com/ec2/pricing`.

[2]Retrieved 2022, Jan 2 from `https://cloud.google.com/compute/docs/instances/preemptible`.

[3]Retrieved 2022, Jan 2 from `https://azure.microsoft.com/en-us/blog/low-priority-scale-sets/`.

[4]Different providers may have different names for similar services. We adopt Amazon's notations in this proposal.

reserved instance has a much longer subscription time. To compensate for the longer duration, reserved instance has a considerably lower unit price compared with the corresponding on-demand instance. Both on-demand and spot instances have a relatively short subscription time, yet they exhibit two major differences. The first is that while the price of the on-demand instance is fixed, the price of the spot instance can vary over time. The second is that spot instances can be unexpectedly terminated. In exchange for the possibility of termination, spot instances are available at a significant discount compared to on-demand instances.

Users diversity can clearly take many different forms, and we focus on two core aspects with direct influence on a user's choice of whether to adopt the service, namely, job value and timeliness of job completion or sensitivity to delays caused by service delays or/and interruptions. The value of a compute job is obviously of relevance when it comes to determining what a user is willing to pay to have it executed. Potentially more interesting in the context of this work is a job's sensitivity to any delay in its execution, as by offering customers with low delay sensitivity a "delayed" service, *i.e.,* a service with a longer delay in exchanging for a lower price, the service provider potentially can meet more customer needs, and therefore, increase its overall revenue.

In this work, we propose to improve the efficiency in cloud computing taking advantage of customer heterogeneity regarding job value and sensitivity to delay. We first study the role of pricing in realizing improvements and leveraging heterogeneity. After that, we explore methods to boost utilization of infrastructure resources, where we particularly focus on the network side. Specifically,

- For the pricing part, we characterize the conditions under which providing a "delayed" service is beneficial, *i.e.,* results in a higher revenue, for the service provider, and then offer guidelines for both users and providers.

- For the infrastructure part, we consider the inter-data center network that carry flows with different latency requirements, and then explore methods to decrease the required (inter-data center) network bandwidth without violating each flow's deadline (delay target). Particularly, we propose to rely on scheduling and shaping of traffic flows in a manner that accounts for their delay requirements, so as to decrease the minimum required bandwidth for inter-data center networks.

Contributions from this thesis have resulted in several publications, with results related to pricing appearing in [171, 174], while [173] documents network optimization results in the single-node case with extensions to the multiple nodes cases forthcoming [172].

1.2 Organization of Thesis

This thesis presents the research of two interesting factors in improve cloud efficiency: pricing factor and network factor. In the following chapters, both topic will expanded in details.

In Chapter 2, we consider a cloud provider that seeks to maximize its revenue by offering services with different tradeoffs between cost and timeliness of job completion. Spot instances and preemptible instances are examples of such services, with, in both cases, possible service interruptions delaying a job's completion. Our focus is on exploiting heterogeneity across jobs in terms of value and sensitivity to execution delay, with a joint distribution that determines their relationship across the user population. We characterize optimal (revenue maximizing) pricing strategies and, in the case of spot instances, optimal bidding strategies as well as

identify conditions under which bidding at a fixed price is optimal.We show that correlation between delay sensitivity and job value needs to exceed a certain threshold for a service offering that differentiates based on speed of execution to be beneficial to the provider. We further assess the results' robustness under more general assumptions, and we offer guidelines for users and providers.

In Chapter 3, we consider traffic flows between datacenters over private networks. The operators of those networks have access to detailed traffic profiles with performance goals that need to be met as efficiently as possible, *e.g.,* realizing latency guarantees with minimal network bandwidth. We focus first on the most basic network configuration, namely, a single link network, and then extend our one-hop results to more general, multi-node networks. We study the extent to which traffic (re)shaping can be of benefit. Specifically, for the single link network we explore different types of schedulers of varying complexity in the form of optimal solutions, whereas for the multi-node network we propose algorithms for different types of schedulers. Our results demonstrate how judicious traffic shaping can help lower complexity schedulers perform nearly as well as more complex ones.

In Chapter 4, we first conclude our research about using pricing and network to improve resource efficiency in cloud computing. Then we discuss the expectations and suggestions of future work.

Chapter 2

Pricing Strategies for Delay Differentiated Cloud Services

2.1 Background and Motivations

Cloud computing has experienced explosive growth and become the computing platform of choice for an increapricingsingly diverse set of users. Cloud providers have responded to this growing diversity by offering different types of services, each with its own pricing scheme.

For example, Amazon, the largest cloud provider [33], offers multiple pricing options[5] to accommodate customers' heterogeneity, *i.e.,* differences in job requirements, tolerance to delay and interruption, job duration, data location, etc. Among these options, on-demand and spot instances are of most relevance to this work, as they embody different trade-offs between cost and characteristics of the service.

[5]Retrieved 2019, March 1, from `https://aws.amazon.com/ec2/pricing`.

Both *on-demand instances* and *spot instances* let users dynamically request compute capacity one hour (or one second for Linux instances[6]) at a time but they exhibit two major differences. The first is that while the price of on-demand instances is fixed, the price of spot instances can vary over time. The second and more important is that spot instances can be unexpectedly terminated. Specifically, spot prices are updated every hour, though after Amazon's recent changes to the service [19] they now typically change less frequently[7], and users register a *bid* that represents the maximum price they are willing to pay. When a user's bid falls below the spot price, its instance is terminated following a two-minute interruption notice. In exchange for the possibility of termination, spot instances are available at a significant discount compared to on-demand instances, and Amazon's Spot Instance Advisor[8] offers users advice and historical data on pricing and frequency of interruption for different instance types. Spot instances are, therefore, a potentially attractive option for jobs that can or have been instrumented to tolerate interruptions[9], and are willing to trade a lower cost for a possible delay in completing their execution. In this work, we focus on jobs that fall in this category.

Besides Amazon, Alibaba Cloud[10] and Packet[11] also offer services similar to spot instances (called preemptible instances and spot market, respectively), where the customer specifies a maximum price (bid) per hour for a specified instance type. If the bid is no less than the current market price, the instance is billed according to the current market price; otherwise, the instance is automatically released and restarts only when its bid again exceeds the market

[6]Retrieved 2019, March 1 from `https://aws.amazon.com/blogs/aws/new-per-second-billing-for-ec2-instances-and-ebs-volumes/`.

[7]Quoting from [14]: "The Spot price may change anytime, but in general, it will change once per hour and in many cases less frequently."

[8]Retrieved 2019, March 1 from `https://aws.amazon.com/ec2/spot/instance-advisor/`.

[9]We also note that spot instances can be attractive for short jobs that can complete before the spot price can be updated to create an interruption.

[10]Retrieved 2019, March 1 from `https://www.alibabacloud.com/help/doc-detail/52088.htm`.

[11]Retrieved 2019, March 1 from `https://support.packet.com/kb/articles/spot-market`.

price. Additionally, Microsoft Azure and Google Cloud, the two largest cloud providers after Amazon [33], offer services that can be viewed as offering a similar trade-off ([99] makes a similar claim, arguing that Amazon's new version of its spot service is now closer to these offerings from Google and Microsoft). Specifically, low-priority VMs[12] (Microsoft) and preemptible instances[13] (Google) are available at a lower price than standard on-demand instances, though one that is now fixed (the aspect of bidding is absent in both services). As with spot instances, this lower price is in exchange for the possibility of service interruptions through preemption. For simplicity, in the rest of the work we refer to the service offerings from Google and Microsoft as preemptible instances or VMs.

This work is concerned with a cloud computing offering that contemplates such a mix of services, and aims to develop a better understanding for when and why they may be of benefit to a cloud provider, *i.e.,* help it maximize its revenue, and for the pricing strategies that can then realize those benefits in the presence of diversity in jobs' profiles. Specifically, our primary goal is to elucidate conditions under which a cloud provider can increase its revenue by offering services that allow users to trade-off a lower cost for the possibility of service interruption, and therefore delays in jobs' completions. In addition, when bidding is involved, as is the case with spot instances, we also seek to devise effective (for cloud users) bidding strategies. As stated in Section 3.3.1, our focus is primarily on this latter scenario, *i.e.,* the variable prices of spot instances (at least their original version). This is because such a system is more general and, as argued in [99], offers greater benefits, including to users who, through their bidding, can chose the trade-off between price and performance that best fits their applications' profiles.

[12]Retrieved 2019, March 1 from `https://azure.microsoft.com/en-us/blog/low-priority-scale-sets/`.

[13]Retrieved 2019, March 1 from `https://cloud.google.com/compute/docs/instances/preemptible`.

2.1.1 Assumptions and Limitations

Job Profiles

Diversity in a job's profile clearly takes many different forms, and we focus on two core aspects with a direct influence on a user's willingness to pay, namely, job value and timeliness of job completion or sensitivity to delays caused by service interruptions. We acknowledge that many other factors influence both users and provider's decisions, *e.g.,* geographic constraints, type of compute instances, scheduling options, etc. However, a job's intrinsic value and the extent to which it is time-sensitive (or not) represent fundamental aspects of how cloud users and providers interact.

The value of a compute job is obviously of relevance when it comes to determining what a user is willing to pay to have it executed. Potentially more interesting in the context of a spot service, or more generally any service where a job's execution can be interrupted, *e.g.,* through preemption, is the job's sensitivity to delays in its successful completion. This sensitivity can be expected to affect a user's willingness to pay for different services, and in the case of dynamic prices, as with spot instance, the user's bidding strategy, *i.e.,* high bids ensure immediate execution while low bids are more likely to be interrupted multiple times and incur large completion delays. In such an environment, an important question for a provider seeking to maximize its revenue, is how to account for differences in job value and sensitivity to delay across users when deciding what services to offer and how to price them.

The first step in accounting for those differences is obviously for the cloud provider to learn about them. Job profiles are typically private information, but the cloud provider can acquire knowledge of the profiles' distribution over the user population. This information can be derived through methods from market research [144], *e.g.,* customer interviews, surveys and

questionnaires [41]. For example, a common approach for eliciting customer preferences along multiple dimensions, *e.g.,* value and delay sensitivity, is conjoint analysis [22, 165, 81], which relies on surveys and statistical techniques to assess customers valuations across different attributes in their purchasing decisions.

The details of our model for job profiles can be found in Section 3.3.1, but it allows for jobs with different sizes (total execution time), value generated per unit of execution time, and sensitivity to delay in execution completion. Heterogeneity across jobs is captured by making those quantities random variables with different distributions, and more importantly by introducing correlation between them. To keep the analysis tractable, correlation is present only between job value and sensitivity to delay, and we assume that both are independent of a job's size. This still allows for short jobs to be more valuable than large jobs, *i.e.,* by having a higher unit value, and for large jobs to be either more or less sensitive to service interruptions than short jobs. However, it does not allow for scenarios where large jobs are systematically associated with higher (or lower) unit value than short jobs, nor does it accommodate situations where large jobs are more (or less) sensitive to interruptions than short jobs. In spite of those limitations, the model still allows for a meaningful investigation of how heterogeneity in jobs' (unit) value and sensitivity to interruptions affects a cloud provider's revenue from different services and how to price them.

Pricing Strategies

Pricing is generally a function of both offer (provider's capacity) and demand (user). In this work, we make the assumption that pricing is *not* responsive to demand. The arguments on which we predicate this assumption are two-fold, but both rely on the scale of the facilities that host modern cloud services.

First, we assume, as others have done [114], that the provider capacity is large enough (near-infinite) to accommodate most demand. In particular, there is empirical evidence [193] that cloud providers know how to provision their resources to accommodate surges. As a matter of fact, this "auto-scaling" property of the cloud is often put forward as one of its main selling points [21, 149]. More generally, the assumption of near-infinite capacity is not unreasonable given the size of modern cloud computing facilities, and the fact that powered-down servers can be quickly brought online when needed [117, 134, 148]. More specifically, modern data centers usually range from $50,000$ to $80,000$ servers[14], and this sheer scale means that the odds of a new typical request finding a "full" system are very small. For example, assuming for simplicity that all instances require an entire server (this is likely conservative as most instances consume only a fraction of a typical server) and arrive according to a Poisson process, a simple approximation [87] for an $80,000$-server system with a load factor of 90% gives a blocking probability upper-bounded by 10^{-4}.

In practice and in spite of large transient spikes in load, cloud systems commonly[15] operate at an average utilization well below 90%, *e.g.,* of the order of 65%[16], which translates into even smaller blocking probabilities. Furthermore, cloud providers typically run multiple, geographically distributed data centers, *e.g.,* today Amazon has over 50 data centers in the US and more than 100 worldwide[17]. Although data co-location requirements or even limitations imposed by the cloud provider can affect a job's ability to run across multiple data centers, the availability of such an option can increase by two orders of magnitude the

[14]Retrieved 2019, March 1 from `https://www.forbes.com/sites/johnsonpierr/2017/06/15/with-the-public-clouds-of-amazon-microsoft-and-google-big-data-is-the-proverbial-big-deal/\#7c225a482ac3`.

[15]Google cloud may be an exception, as its cloud computing offering runs on the same infrastructure that Google uses internally to run its own end-user products. The latter can often soak-up much of the spare capacity available.

[16]Retrieved 2019, March 4, from `https://aws.amazon.com/about-aws/sustainability/`.

[17]Retrieved 2019, March 1 from `https://www.sdxcentral.com/articles/news/wikileaks-publishes-the-location-of-amazons-data-centers/2018/10/`.

number of servers accessible by jobs without such constraints. This in turn decreases their blocking probability by approximately the same factor.

Our second argument in support of pricing that is not responsive to demand is the combination of scale and the ever finer (time) granularity at which cloud resources are allocated and billed (recall the per second billing of EC2 Linux instances); a trend that is expected to continue [25]. This makes demand aware short-term price adjustments increasingly complex. In the case of Amazon spot instances, early empirical studies [24, 196] hinted at pricing that was often not responsive to demand. The recent change to Amazon spot service [19] points to even lesser variability, and therefore responsiveness (to variations in demand), with spot prices reflecting *"long-term trends in supply and demand for EC2 spare capacity*[18]*."* Like Amazon, Alibaba cloud makes historical pricing data available for its own spot service, and while we did not carry out a detailed investigation of their variations, observations based on a one-month window (February 23, 2019 to March 23, 2019) revealed trends essentially similar to those of Amazon's updated offering, *i.e.,* relatively regular patterns of small changes with occasional larger shifts likely caused by provisioning decisions. And clearly, the fixed prices of Google preemptible instances and Microsoft low-priority VMs are not responsive to short-term fluctuations in demand.

In summary, we assume that the scale of the infrastructure and the complexity of implementing and managing demand responsive pricing at this scale combine to create an environment where the pricing of cloud services is not responsive to short-term variations in demand. In other words, the infrastructure runs out of resources so rarely that the potential benefits of demand responsive pricing do not justify the added cost. Instead, pricing reflects differences in service characteristics, and in particular how those characteristics affect the timeliness

[18]Retrieved 2019, March 1 from `https://docs.aws.amazon.com/aws-technical-content/latest/cost-optimization-leveraging-ec2-spot-instances/how-spot-instances-work.html`.

of service completion, *e.g.,* because of the possibility of service interruption as captured by spot instances (Amazon, Alibaba, and Packet) and preemptible instances (Google and Microsoft). Note that we do not mean to imply that cloud demand never exceeds capacity. In particular, some of the cloud's largest tenants can generate peak workloads that may at time exceed available capacity, or transient imbalances across regions could occasionally deplete resources in a given geography. Our primary argument is that such occurrences are rare and not reflected in short-term price variations.

As alluded to earlier, under these assumptions, our focus is on exploring how a cloud provider should select and price the services it offers towards maximizing profit, given information on the profiles of potential jobs. Specifically, given information regarding market segmentation along jobs' value and sensitivity to delay and how they relate to each other (are correlated), what services should the provider offer and at what price. The family of services under consideration involves adding uncertainty regarding resource availability, *e.g.,* through variations in pricing as in spot services or through direct preemption, in exchange for a lower price, potentially akin to the "damaged good" approach originally used by IBM in its high-speed printer offerings [61]. In exploring pricing strategies, we also leverage the cloud's scale and neglect the impact of the few cases where demand may exceed capacity. In other words, given information on jobs' profiles, prices are set so as to maximize the demand offered to the cloud, *i.e.,* the set of jobs that, given the services offered by the cloud and their pricing, derive value from using the cloud. The question we address is whether introducing a "lower quality" (because of the possibility of interruptions) but lower price service can help the provider increase its revenue.

We make this more formal in Section 3.3.1, but assuming that services are characterized by a price and a probability of interruption, *i.e.,* being unable to run in a given "slot," a pricing strategy then involves selecting both. In the case of spot instances, the provider

selects the spot prices and when to change them. Together, these determine the probability of interruption associated with a given service cost, with more expensive services (higher bids) having a lower probability of interruption. Historical pricing data can then, as we shall see in Section 2.4, allow users to select the best bidding strategy given their profile. In the simpler case of preemptible instances with fixed prices, the provider again sets the price and controls preemption decisions. Neither the odds and timing of preemptions nor how quickly preempted instances can typically restart are explicitly stated by either Microsoft or Google[19], though mechanisms are available that, at least for Google[20], can let users estimate them from empirical data, and therefore evaluate the benefits of the service, *i.e.,* estimate how much longer a job may take to complete.

In exploring the benefits of offering such combinations of services, our investigation reveals several interesting features. In particular, we identify that introducing services that offer a trade-off between cost and timeliness of execution, as spot instances and preemptible instances do, is of benefit, *i.e.,* increases the provider's revenue, only if the *correlation* between jobs' value and their sensitivity to execution delays exceeds a certain threshold. Further, a pricing strategy that simply assigns different odds of interruption to services with different prices (and correspondingly select those prices), can in many of those cases (see Section 2.5.1) successfully extract most of the value from the market. In the case of a spot service for which both pricing and bidding are relevant, under the simplifying assumptions that jobs that bid do so till completion and have a utility that decreases linearly as a function of their execution delay, we also find that a fixed bidding strategy, *i.e.,* a strategy that repeatedly bids at the same fixed value, is optimal for users that decide to bid, and the decision to bid or not is

[19]Google mentions that preemptible instances are preempted at least once every 24 hours with preemption rates that vary between 5% and 15% over that period (`https://cloud.google.com/compute/docs/instances/preemptible\#preemption_process`, retrieved 2019, July 24).

[20]Retrieved 2019, July 24 from `https://gatkforums.broadinstitute.org/firecloud/discussion/10513/preemptible-instances-historical-data-of-google-compute-engine-vm-running-time-before-preemption`.

solely a function of the job's value. The robustness of the results is investigated numerically under more general conditions, including allowing job termination prior to completion and simple forms of convex and concave delay penalties.

The need for a strong correlation between job value and sensitivity to delay offers cloud providers useful insight by illuminating when the deployment of a multi-price system can realize an effective market segmentation and increase revenue. Conversely, when such a multi-price offering is provided through multiple (randomly[21]) varying prices, as in a spot service, our investigation provides intuition for simple bidding strategies under different combinations of job value and sensitivity to delay, including when bidding at a constant price is effective and conversely when and why changing one's bid can improve the odds of a favorable outcome.

The rest of the work is structured as follows. Section 3.2 briefly reviews previous works of relevance. Section 3.3.1 introduces our model more formally. Section 2.4 investigates user's bidding strategies under the general framework of a spot service, while Section 2.5 explores the provider's pricing strategies. The robustness of the results are tested numerically under more general assumptions in Section 2.6. Section 2.7 summarizes the work's findings and identifies potential future directions. A glossary summarizing the notation used in the work's main body is provided in Appendix A.1.

2.2 Related Works

In this section, we briefly review the vast literature on pricing in computing systems and highlight a few recent relevant works, including works that have explicitly targeted developing a better understanding for spot pricing and its benefits.

[21] At least from the user's perspective.

The idea of using pricing for resource allocation in computing systems with jobs that are heterogeneous in either value or sensitivity to delay is not new. It dates back to the 1960's with pricing for shared computing time *e.g.,* [62, 177]. In this early context, computing resource were typically constrained, with pricing used to realize an allocation of resources that maximized a global utility function across heterogeneous users. The finite resource assumption naturally lent itself to a queueing system formulation very different from that of this chapter. The 1990 paper by Mendelson and Whang [140] offers a representative example. It considers an M/M/1 queue with multiple classes of jobs with different valuations and sensitivity to delay, and investigates pricing policies that maximize utility (social welfare) across classes. Conversely, [3] considers the provider's revenue maximization problem, and demonstrates that the damaged goods strategy of [61] also applies in this context.

However, while ideas of pricing regularly surfaced in the academic computing literature, their use in practice was limited [167] as pricing was never really necessary for the continued development and operation of large computing systems. Most such systems were centrally controlled, *e.g.,* by the organization running the mainframe, which made defining usage policies easy, so that the complexity and cognitive overhead of mechanisms that price resources were not justified. Thus, research focused on scheduling algorithms to maximize utilization of shared resources rather than pricing policies to achieve explicit social goals [167].

The emergence of the cloud, with computing as a utility, changed this with pricing emerging as a major control knob, *e.g.,* see [116, 130] for recent surveys of related pricing schemes, and [108] for a discussion on the role of pricing in cloud computing. Users became accustomed to thinking about paying for individual jobs, with timeliness part of their assessment.

In this context, the most relevant works are [199, 205], [1, 114, 164, 64, 97] (see also [2] for an updated and expanded version of [1]), and [51].

Both [205] and [199] focus on deriving optimal (spot) bidding strategies that minimize the users' overall price paid for the service. However, neither explicitly considers the impact on a job's utility of the delay penalty associated with different strategies. Specifically, [205] targets fixed bidding strategies in an infinite-capacity market, when jobs incur a recovery cost each time they are interrupted. Too many interruptions can result in a job never completing if the recovery overhead of each interruption exceeds the average amount of work performed in each round. The paper identifies an optimal bidding strategy that minimizes cost while guaranteeing a finite execution time. [199] comes closest to incorporating some aspect of delay sensitivity by exploring bidding strategies when jobs have deadlines (a deadline constraint is a special case of the convex delay penalty we investigate in Section 2.6.3). In that context, it devises a dynamic bidding strategy that minimizes a job's cost while meeting its deadline.

The works of [1, 114] explicitly target cloud computing services under a (mostly) infinite capacity setting, while seeking to understand how job value and sensitivity to delay affect cloud service offerings. [64] relates closely to [1] as both rely on a queueing-theoretic framework, but it assumes a finite cloud capacity and accounts for operational costs on the provider side and preemption costs on the customer side. Both offer interesting initial insight, especially conditions under which a spot service can or cannot outperform an on-demand service, but leave several questions unanswered, in particular regarding the role of correlation between a job's value and its sensitivity to delay. As we shall see, this plays an important role in determining to what extent a spot service can add value to an on-demand service offering. [164] is more qualitative in nature, but its main observation that a simple fixed bidding strategy is useful in practice echoes our result of Section 2.4. [97] studies a cloud market with unit demand, risk-averse bidders, and shows that a market with both on-demand and spot

services generates higher revenue and higher welfare than systems with only spot service. It, however, does not explicitly incorporate delay into its utility function.

The setting of [51] is different in that it considers an inventory management system with customers that seek to purchase a single type of product, *e.g.,* furniture, but the coupling of product valuation and sensitivity to delivery delay bears similarities to the trade-off cloud computing users face when bidding for spot instances, *i.e.,* bid high in exchange for a lower execution delay. There are, however, enough differences between the two environments, *e.g.,* the presence of an inventory cost and the requirement that a user's delay disutility rate be perfectly positively correlated with her valuation, that the results of [51] are not readily applicable to our problem.

2.3 Model Formulation

Because of its greater generality, our investigation is carried out under the auspices of a spot service, for which both pricing (by the provider) and bidding (by the user) are relevant. Specifically, we consider a setting with a single cloud provider, *i.e.,* a monopoly[22] environment, where, as alluded to earlier, the provider is assumed to have access to "infinite" compute resources, with prices set independent of the (instantaneous) demand. In other words, we assume that the provider offers a spot service with spot prices drawn from a set of n prices $\boldsymbol{p} = (p_1, p_2, ..., p_n), p_1 < p_2 < ... < p_n$, with a probability distribution $\boldsymbol{\pi} = (\pi_1, \pi_2, ...\pi_n)$.

Spot prices are updated periodically by randomly selecting a price from $\boldsymbol{p}$ according to $\boldsymbol{\pi}$, with, as we shall see, such a randomized pricing strategy offering a simple revenue enhancing

[22]Even if as of today (2019) Amazon does not dominate the cloud market as it once did, it remains bigger than its next 4 largest competitors [33]. In addition, the switching cost of migrating from one provider to another, *i.e.,* moving data, learning a different API, etc., remains relatively high, so that a monopoly assumption for existing customers is not unreasonable.

option. Throughout the chapter, we denote as a "slot" the (fixed) time between successive instances of possible price updates by the provider. Jobs with bids at or above the spot price pay the spot price and are allowed to execute, but their execution is stopped whenever their bid falls below the spot price[23].

We note that under this framework and as discussed in Section 3.1, bidding at a given price translates into a corresponding probability of service interruption, *i.e.,* the probability that the spot price exceeds the bid, and a given (average) service price, *i.e.,* as realized over a typical instantiation of spot prices. We also note that with this definition of a spot service, bidding at the highest price all but eliminates the possibility of service interruptions, *i.e.,* is equivalent to an on-demand service[24]. In other words, a spot service with a *single* price is de facto equivalent to an on-demand service with no delay and/or interruptions in the execution of a job. Correspondingly, if the optimal spot pricing policy involves only one price, the conclusion is that there is then no benefit (to the provider) in offering such a service in addition to an on-demand service. Last but not least, as alluded to earlier, we also note that the special case of only two (spot) prices is essentially equivalent to the combination of an on-demand service (bidding at the highest spot price) and a service akin to Google's and Microsoft's preemptible instances, with a preemption probability equal to the probability that the spot price is at its high value. Appendix A.6.5 formalizes this equivalence.

Demand for computing services is heterogeneous and originates from a large job population. Each job is characterized by its its profile (t, v, κ), which consists of its total computation time or length $t > 0$, its value v per unit of computation time, *i.e.,* the value of a job of length t is

[23]We note that in 2015 Amazon introduced a spot service, Spot Blocks, where users can specify a duration of up to 6 hours during which they are guaranteed not to be interrupted. The pricing of this service is, however, different from that of standard spot instances, *i.e.,* in the form of a discount relative to on-demand pricing. We leave investigating this additional option as future work.

[24]In practice though, the highest spot price would be set to exceed the on-demand price, if only to avoid offering a semantically similar service (no interruption) but at a lower (average) cost.

vt, and a measure of its sensitivity to computation delays, as captured by a parameter κ. For example, a value of $\kappa = 0$ corresponds to jobs that are completely insensitive to execution delays, *e.g.,* a batch job used to run in the background, while as κ increases so does the job's sensitivity to delay. In other words, κ determines the scale at which a job's utility decreases when its completion time increases because of interruptions. Users are assumed to know t, v and κ for all their jobs. For notation purposes, a job is represented through its triplet (t, v, κ), with (v, κ) denoting the profile of all jobs with unit value v and delay sensitivity κ. Job lengths are assumed independent of v and κ, and are drawn from a probability distribution with density function[25] $f(t)$. Note that the independence of v and t implies that a long job with a small v can be less valuable than a short job with a large v. Conversely, jobs' value and sensitivity to delay are drawn from a distribution with joint density function $q(v, \kappa)$. Hence, we allow for correlation between v and κ, *e.g.,* high value jobs can be more sensitive to delay (positive correlation). The assumption that t is independent of v and κ, however, implies that, as pointed out in Section 2.1.1, we cannot capture situations where a job's size affects its per unit value or sensitivity to delay. Extending the investigation to include more complex correlation such as those would be of interest.

Users are aware of $\boldsymbol{p}$ and $\boldsymbol{\pi}$, *e.g.,* from data published by the cloud provider (as with Amazon or Alibaba spot instances), and use this information to select bidding strategies for new jobs. A bidding strategy $\Gamma_{\boldsymbol{p},\boldsymbol{\pi}}(t, v, \kappa)$ (denoted simply as Γ or $\Gamma(t, v, \kappa)$ when unambiguous) is a function of $\boldsymbol{p}$ and $\boldsymbol{\pi}$ as well as (t, v, κ), and specifies both a first bidding price, as well as bids at subsequent bidding instances, possibly as a function of spot prices realizations. This makes for a complex strategy space though it includes simple strategies. For example, Γ could be a state-dependent strategy whose next step only depends on the number of previous winning bids, or Γ could be as simple as bidding at the same price until the job completes,

[25]The description assumes that t, v, and κ are continuous random variables. Similar expressions are readily available for discrete random variables.

i.e., a fixed bidding strategy. Given a bidding strategy Γ and a realization of spot prices $<\boldsymbol{p}>_{\boldsymbol{r}}$, the utility of a completed job (t, v, κ) is of the form

$$u_r(t, v, \kappa, \Gamma, <\boldsymbol{p}>_{\boldsymbol{r}}) = vt - p_r(t, v, \kappa, \Gamma, <\boldsymbol{p}>_{\boldsymbol{r}}) - d_r(t, v, \kappa, \Gamma, <\boldsymbol{p}>_{\boldsymbol{r}}),$$

where vt is the realized job's value upon completion (there is no partial value for incomplete jobs), $p_r(t, v, \kappa, \Gamma, <\boldsymbol{p}>_{\boldsymbol{r}})$ is the total price paid by the job under strategy Γ and spot prices realization $<\boldsymbol{p}>_{\boldsymbol{r}}$, and $d_r(t, v, \kappa, \Gamma, <\boldsymbol{p}>_{\boldsymbol{r}})$ is the delay penalty incurred by the job given its delay sensitivity κ, its use of strategy Γ, and spot prices realization $<\boldsymbol{p}>_{\boldsymbol{r}}$.

Intuitively, the delay penalty $d_r(t, v, \kappa, \Gamma, <\boldsymbol{p}>_{\boldsymbol{r}})$ should be an increasing function of κ and of the execution delay $t_r(t, v, \kappa, \Gamma, <\boldsymbol{p}>_{\boldsymbol{r}})$ experienced by the job[26] beyond its execution time t (the delay is 0 when bidding at p_n and maximum when bidding at p_1). There are many possible choices for such a function, and in Section 2.6.3 we experiment with concave and convex increasing functions, albeit specialized to simple cases of piece-wise linear functions. However, for analytical tractability, we assume here a linear function of the form $\kappa t_r(t, v, \kappa, \Gamma, <\boldsymbol{p}>_{\boldsymbol{r}})$, so that

$$u_r(t, v, \kappa, \Gamma, <\boldsymbol{p}>_{\boldsymbol{r}}) = vt - p_r(t, v, \kappa, \Gamma, <\boldsymbol{p}>_{\boldsymbol{r}}) - \kappa t_r(t, v, \kappa, \Gamma, <\boldsymbol{p}>_{\boldsymbol{r}}).$$

Note that a linear delay penalty implies a linearly decreasing utility function (when jobs are not executing), which implicitly assumes that customers are risk-neutral. This ignores the possibility of risk-averse customers (with strictly concave utility functions), as introduced in [97]. The extensions of Section 2.6.3 address this issue to some extent by, among other things, considering convex delay penalties (that map to concave utility functions).

[26] We define $t_r(t, v, \kappa, \Gamma, <\boldsymbol{p}>_{\boldsymbol{r}})$ only on completed jobs.

Returning to our model, given $\boldsymbol{p}, \boldsymbol{\pi}$, and job profile (t, v, κ), a user selects among bidding strategies to maximize the job's *expected utility*, where expectation is computed over spot prices realizations. When the maximum expected utility is negative, the user refrains from bidding. In this case, the expected utility is taken to be zero. Otherwise, the job's proceeds to bid according to the strategy Γ^* that maximizes its expected utility. We make two assumptions regarding bidding strategies. The first is that once bidding starts it continues *until job completion, i.e.,* early termination is not allowed[27]. The second is that bidding strategies are restricted to always bid at p_1 (the lowest spot price) or higher. Bidding below p_1 is equivalent to not bidding, and could therefore bypass our requirement that bidding continues until job completion. We call strategies that satisfy both assumptions *persistent strategies.*

Those assumptions are in part motivated by the fact that a strategy's expected utility is computed over all possible spot price realizations. Hence, under a given realization even the best strategy needs not always realize a non-negative utility for a given job, *e.g.,* unlucky sequences of spot prices can readily produce a negative utility. This could in turn produce (dynamic) strategies that at some point determine that the best option is to stop bidding (or constantly bid below p_1). This would then result in some jobs not running to completion, which adds significant complexity to the derivation of a strategy's expected utility. The above two assumptions allow us to circumvent those complexities, which we explore numerically in Section 2.6.2 that considers strategies that allow a job's termination when the expected residual utility from continuing to bid stops being positive[28].

[27]This is not unreasonable given the sunk cost associated with deploying and initiating a job.

[28]In such cases, the (non-positive) utility contributed by a terminated job consists solely of the execution payments made prior to termination, *i.e.,* there is neither partial value nor delay penalty for the job's incomplete execution.

Under those assumptions, a job's expected utility under bidding strategy Γ is of the form

$$U_{\boldsymbol{p},\boldsymbol{\pi}}(t, v, \kappa, \Gamma) = vt - P_{\boldsymbol{p},\boldsymbol{\pi}}(t, v, \kappa, \Gamma) - \kappa T_{\boldsymbol{p},\boldsymbol{\pi}}(t, v, \kappa, \Gamma), \tag{2.1}$$

where $P_{\boldsymbol{p},\boldsymbol{\pi}}(t, v, \kappa, \Gamma)$ is the expected execution price of the job under bidding strategy Γ, and $T_{\boldsymbol{p},\boldsymbol{\pi}}(t, v, \kappa, \Gamma)$ is the expected delay beyond the job's execution time t under Γ.

Given a pricing configuration $(\boldsymbol{p}, \boldsymbol{\pi})$, selecting an optimal bidding strategy $\Gamma^*_{\boldsymbol{p},\boldsymbol{\pi}}(t, v, \kappa)$ for a job (t, v, κ), therefore, consists of solving the following problem:

$$\Gamma^*_{\boldsymbol{p},\boldsymbol{\pi}}(t, v, \kappa) = \arg\max_{\Gamma} U_{\boldsymbol{p},\boldsymbol{\pi}}(t, v, \kappa, \Gamma), \tag{2.2}$$

with the user proceeding to bid iff $\Gamma^*_{\boldsymbol{p},\boldsymbol{\pi}}(t, v, \kappa)$ yields a non-negative expected utility.

While users are interested in knowing how to select an optimal bidding strategy given $\boldsymbol{p}, \boldsymbol{\pi}$, and (t, v, κ), a cloud provider seeks to select $\boldsymbol{p}$ and $\boldsymbol{\pi}$ given some knowledge about job profiles and strategies, so as to maximize its expected per job revenue $R_{f,q}(\Gamma^*(\boldsymbol{p}, \boldsymbol{\pi}))$, *i.e.,* solve

$$(\boldsymbol{p^*}, \boldsymbol{\pi^*}) = \arg\max_{\boldsymbol{p},\boldsymbol{\pi}} R_{f,q}(\Gamma^*(\boldsymbol{p}, \boldsymbol{\pi})), \tag{2.3}$$

where the notation $\Gamma^*(\boldsymbol{p}, \boldsymbol{\pi})$ seeks to indicate the dependency of the users' strategy on $\boldsymbol{p}$ and $\boldsymbol{\pi}$, and $R_{f,q}(\Gamma^*(\boldsymbol{p}, \boldsymbol{\pi}))$ is given by

$$R_{f,q}(\Gamma^*(\boldsymbol{p}, \boldsymbol{\pi})) = \iiint_{t,v,\kappa} f(t) q(v, \kappa) P_{\boldsymbol{p},\boldsymbol{\pi}}(t, v, \kappa, \Gamma^*) dt\, dv\, d\kappa.$$

In solving the optimization of Eq. (2.3), we assume that the triplets (t, v, κ) are private information, but that the cloud provider knows $f(t)$ and $q(v, t)$, *e.g.,* through the kind of methods mentioned in Section 3.1.

In the next two sections, we proceed to first characterize Γ^* and then $(\boldsymbol{p^*}, \boldsymbol{\pi^*})$ under the above assumptions. Note that this can be cast as a standard Stackelberg game: The service provider chooses and announces $(\boldsymbol{p^*}, \boldsymbol{\pi^*})$ to maximize its revenue based on users' behavior, and this is followed by users choosing to bid according to $\Gamma^*_{\boldsymbol{p^*},\boldsymbol{\pi^*}}$ so as to maximize their utility.

2.4 Optimal Bidding Strategy

In this section, we show that given our previous assumptions, an optimal fixed bidding strategy, *i.e.,* a strategy that bids at the same value until job completion, always exists and only depends on κ.

Since jobs that start bidding are not terminated, from Eq. (2.1) maximizing utility is equivalent to minimizing total expected cost

$$C_{\boldsymbol{p},\boldsymbol{\pi}}(t, v, \kappa, \Gamma) = P_{\boldsymbol{p},\boldsymbol{\pi}}(t, v, \kappa, \Gamma) + \kappa T_{\boldsymbol{p},\boldsymbol{\pi}}(t, v, \kappa, \Gamma), \tag{2.4}$$

For ease of presentation, we also assume that job lengths are integer multiple of the slot length, and w.l.o.g. assume the slot length to be 1. We narrow the strategy space to those with bid values in $\{p_1, ..., p_n\}$, which as per Lemma 24 in Appendix A.3 does not affect the optimal strategy.

2.4.1 Characterizing the Optimal Bidding Strategy

The derivation of the optimal strategy involves a number of technical steps, starting with jobs of length one-slot for which we establish that the optimal strategy is a fixed bidding strategy, *i.e.,* a strategy that bids at the same value until the job completes. We relegate

those technical steps to Appendix A.3, and proceed to state our main result that identifies the structure of the optimal bidding strategy for jobs of arbitrary length. Specifically,

Proposition 1. [OPT_BID] *Given a pricing strategy* $(\boldsymbol{p}, \boldsymbol{\pi})$ *and a linear penalty function for a job's execution delay, among persistent strategies, a strategy that bids at* $b^*(v, \kappa)$ *in every slot is an optimal bidding strategy for job* (t, v, κ)*, where* $b^*(v, \kappa)$ *is the optimal bid for job* $(1, v, \kappa)$*.*

The proof of Proposition 1 is in Appendix A.4. It establishes not only that a fixed bidding strategy is optimal, but also shows that the optimal bid, $b^*(v, \kappa)$, is independent of the job length t. This is in part because under the assumption that bidding once started does not terminate, the bidding decision that maximizes the expected utility going forward is unaffected by the outcome of previous bids. Additionally, this "memorylessness" also means that maximizing the expected utility from successfully completing one unit of work extends to multiple units of work.

Given the result of Proposition 1, the next step is to characterize $b^*(v, \kappa)$, or more precisely, $b^*(\kappa)$, since, as we shall see, the optimal bid only depends on a job's sensitivity to delay κ.

The job's value, v, however still plays a role in determining whether or not to bid, *i.e.,* v does not influence the bid value, but it is instrumental in deciding whether to bid (or not). Note that $b^*(\kappa)$ can exceed v, as the expected payment for any bid value $b > p_1$ is smaller than b. In characterizing $b^*(\kappa)$, we also show that it can be computed using a simple linear search.

If a user bids for a job (t, v, κ) using a fixed bidding strategy with a bidding price b, the expected fraction of time or probability that the job is active, *i.e.,* wins its bid, is

$$0 < \alpha(b) = \sum_{p_i \leq b} \pi_i \leq 1. \tag{2.5}$$

The expected payment per unit of time *given* that the job is active is then

$$\overline{b} = \frac{\sum_{p_i \leq b} \pi_i p_i}{\alpha(b)}, \tag{2.6}$$

so that the expected payment is

$$P(t, v, \kappa, b) = \overline{b}t. \tag{2.7}$$

and the average job completion time is

$$\begin{aligned} t + T(t, v, \kappa, b) &= \frac{t}{\alpha(b)} \\ \Rightarrow T(t, v, \kappa, b) &= t\left(\frac{1}{\alpha(b)} - 1\right) \end{aligned} \tag{2.8}$$

Substituting Eq. (2.7) and Eq. (2.8) in Eq. (2.4), the expected total cost (payment plus delay penalty) $C(t, v, \kappa, b)$ for the strategy that bids at b is given by

$$C(t, v, \kappa, b) = t\left(\overline{b} + \kappa\left(\frac{1}{\alpha(b)} - 1\right)\right). \tag{2.9}$$

Note that for a given b, $C(t, v, \kappa, b)$ is proportional to t (as is $U(t, v, \kappa, b)$). Note also that the optimal bidding price need not to be unique. W.l.o.g, assume that users bid at the lowest

optimal price. Recalling that $p_1 < p_2 < \ldots < p_n$, the optimal fixed bidding price is then

$$\begin{aligned} b^*(v,\kappa) &= \min \arg\min_{i\in\{1,2,\ldots n\}} C(t,v,\kappa,p_i) \\ &= \min \arg\min_{i\in\{1,2,\ldots n\}} C(1,v,\kappa,p_i), \end{aligned}$$

where the second equality comes directly from $C(t,v,\kappa,b)$'s proportionality to t (and/or Proposition 1). The next proposition offers additional insight into a job's optimal bid as a function of its profile.

Proposition 2. *Under persistent strategies, a job's optimal fixed bid $b^*(\kappa)$ is independent of v and t, and non-decreasing in κ. Specifically, a job with $\kappa \in (\widetilde{\kappa}_{i-1}, \widetilde{\kappa}_i]$ will bid at p_i if it bids, where $\widetilde{\kappa}_i = \sum_{j\leq i}(p_{i+1} - p_j)\pi_j$ for $i \geq 1$, and $\widetilde{\kappa}_0 = -\epsilon < 0$,*

where $\widetilde{\kappa}_i$ is a threshold in a job's delay sensitivity associated with a change in the value of a job's optimal bid to p_i.

The proof is in Appendix A.4, and the proposition provides and explicit value for a job's optimal bid as a function of its delay sensitivity and the set of prices advertised by the cloud provider. We also note that the property that a job's optimal bidding price increases with κ is intuitive, as a higher delay sensitivity implies a greater willingness to pay to avoid delays. Additionally, although Proposition 2 states that a job's valuation v plays no role in its selection of an optimal bidding price, v affects its decision to bid and, as we shall see in Section 2.5, the distribution of v plays a role in the choice of the provider's pricing strategy.

The results of Proposition 2 can also be used to determine a job's optimal bidding price using a simple search procedure, as stated next.

Corollary 3. *A job's optimal bidding price under a persistent strategy can be determined using a simple search,* e.g., *a binary or a Fibonacci search.*

The result is a direct consequence of the proof of Proposition 2, which characterizes the evolution of a job's utility as bids increase from p_1 to p_n. It shows that a job's utility function admits a single maximum as bids keep increasing. This maximum can, therefore, be discovered with a simple search, *e.g.,* a basic binary search or a Fibonacci search [113]. The corollary's proof is constructive in nature and takes advantage of the structure of a job's utility characterized in Proposition 2, which shows that it admits a single maximum as a function of the bidding price.

2.4.2 Numerical Examples

In this sub-section, we introduce a few numerical examples to illustrate the properties of the optimal bidding strategy under a given pricing strategy. We, therefore, fix the pricing strategy to $(p_1, p_2, \pi) = (2, 4, 0.5)$, *i.e.,* two prices that are equally likely with p_2 twice as high as p_1, and we vary job configurations.

In exploring possible bidding strategies, we note that given the memoryless nature of spot price selection, any policy can be decomposed into a sequence of individual (possibly different) policies to be followed after each winning bid (or when the job is first created), *i.e.,* a policy that determines the job's bidding behavior until the next win. Furthermore, in this simple two-price setting, because bids can only take two values, p_1 or p_2, and a bid at p_2 is guaranteed to win, the space of possible strategies is limited. Specifically, a job can only choose from three possible strategies between successive wins: 1) bid at p_1 until the next win, 2) bid at p_2 (a bid at p_2 is guaranteed to win), and 3) bid at p_1 for up to l slots (for some $l > 0$), before switching to a bid at p_2 if all l bids at p_1 fail. We use the following notation to capture all three strategies: Denote as $\Gamma(l), l \geq 0$, the strategy that bids at p_1 for l slots and if unsuccessful l times switches to bidding at p_2. $\Gamma(\infty)$ is the strategy that bids at p_1 until success, and $\Gamma(0)$ immediately bids at p_2. The strategy space for a job of length t,

therefore, consists of the 3^t possible combinations of the three base strategies $\Gamma(\infty), \Gamma(0)$, and $\Gamma(l), 0 < l < \infty$.

Since, as mentioned earlier, maximizing expected utility is equivalent to minimizing expected cost, we focus on the latter. Recall that cost has two components: payments and delay penalty. The expected costs under $\Gamma(\infty), \Gamma(0)$, and $\Gamma(l), 0 < l < \infty$, can be readily obtained as follows:

$$\begin{aligned} C(\Gamma(\infty)) &= p_1 + \kappa\left(\frac{1}{\pi} - 1\right) \\ C(\Gamma(0)) &= \pi p_1 + (1-\pi)p_2 \\ C(\Gamma(l)) &= \sum_{i=0}^{l-1} \pi(1-\pi)^i[p_1 + \kappa i] + (1-\pi)^l[\pi p_1 + (1-\pi)p_2 + \kappa l] \end{aligned}$$

In other words, bidding at p_1 is preferable to bidding at p_2 only if the expected delay penalty $\kappa(1/\pi - 1)$ is smaller than the expected cost increase of $(1-\pi)(p_2 - p_1)$, *i.e.,* $\kappa < \pi(p_2 - p_1)$. The hybrid strategy $\Gamma(l), 0 < l < \infty$, offers an intermediate trade-off, but as established in Proposition 1, is never able to outperform the better of either $\Gamma(\infty)$ or $\Gamma(0)$.

To illustrate this, we compare the expected utility for the 9 different combinations of base strategies for job $(t, v, \kappa) = (2, 2, 0.5)$. The resulting values are in the next table.

	$\Gamma(\infty)$	$\Gamma(0)$	$\Gamma(l_2)$
$\Gamma(\infty)$	$\mathbf{-1}$	-1.5	$-\frac{1}{2}^{l_2+1} - 1$
$\Gamma(0)$	-1.5	-2	$-1.5 - \frac{1}{2}^{l_2+1}$
$\Gamma(l_1)$	$-\frac{1}{2}^{l_1+1} - 1$	$-1.5 - \frac{1}{2}^{l_1+1}$	$-\frac{1}{2}^{l_1+1} - \frac{1}{2}^{l_2+1} - 1$

where entry, say, $(\Gamma(\infty), \Gamma(0))$, gives the expected utility if the job starts with strategy $\Gamma(\infty)$ and switches to $\Gamma(0)$ after its first win, and l_1 and l_2 denote the value of $l, 0 < l < \infty$, for the $\Gamma(l)$ strategy used for the job's first and second work units, respectively.

From the table, we see that repeatedly bidding at p_1 yields the highest expected utility. This is expected since $\kappa = 0.5 < \pi(p_2 - p_1) = 1$, and consistent with Proposition 1, *i.e.,* a fixed bidding strategy is "optimal." However, because the resulting expected utility is negative, the job will refrain from bidding.

This outcome can be changed simply by increasing the job's value to $v = 4$. Specifically, considering job $(t, v, \kappa) = (2, 4, 0.5)$, the expected utilities across the different combinations of base strategies are now as follows:

	$\Gamma(\infty)$	$\Gamma(0)$	$\Gamma(l_2)$
$\Gamma(\infty)$	**3**	2.5	$3 - \frac{1}{2}^{l_2+1}$
$\Gamma(0)$	2.5	2	$2.5 - \frac{1}{2}^{l_2+1}$
$\Gamma(l_1)$	$3 - \frac{1}{2}^{l_1+1}$	$2.5 - \frac{1}{2}^{l_1+1}$	$3 - \frac{1}{2}^{l_1+1} - \frac{1}{2}^{l_2+1}$

As expected from Proposition 2, the optimal bidding strategy (bidding at p_1) is unaffected by the increase in value, but the larger job value now results in a positive expected utility. Hence, the job proceeds to bid at p_1.

Next, we illustrate the influence of κ on a job's bidding strategy. For that purpose, consider job $(t, v, \kappa) = (2, 4, 2)$ with the same size and value as job $(2, 4, 0.5)$ but a higher κ value. The next table gives the job's expected utility under different combinations of base strategies:

	$\Gamma(\infty)$	$\Gamma(0)$	$\Gamma(l_2)$
$\Gamma(\infty)$	0	1	$\frac{1}{2}^{l_2}$
$\Gamma(0)$	1	**2**	$1 + \frac{1}{2}^{l_2}$
$\Gamma(l_1)$	$\frac{1}{2}^{l_1}$	$1 + \frac{1}{2}^{l_1}$	$\frac{1}{2}^{l_1} + \frac{1}{2}^{l_2}$

The table shows that increasing κ from 0.5 to 2 has decreased the expected utility of bidding at p_1 from 3 to 0, so that the optimal bidding strategy is now to bid at p_2.

Our last experiment involves illustrating that a job's length, t, does indeed not affect its bidding strategy nor does it affect its decision to bid or not. For that purpose, we reuse the previous job profile but reduce its length from 2 to 1, *i.e.,* we focus on job $(t, v, \kappa) = (1, 4, 2)$. Its expected utility under the three base strategies is reported below

	$\Gamma(\infty)$	$\Gamma(0)$	$\Gamma(l)$
$U(1, 4, 2, \Gamma)$	0	**1**	$\frac{1}{2}^l$

which shows that the optimal strategy is still to bid at p_2, and remains positive.

2.5 Optimal Pricing Strategy

In this section, we turn to characterizing how a cloud service provider should price its spot service *i.e.,* choose its prices and price distribution, to maximize its expected revenue given that users bid according to the optimal bidding strategy of the previous section. Of interest are conditions identifying when a spot service is effective in helping the provider increase its expected revenue over an on-demand service.

For simplicity, we limit ourselves to binary job profiles with only two job values $(0 < v_1 < v_2)$ and delay sensitivities $(0 \leq \kappa_1 < \kappa_2)$. We call this a *binary system*. As pointed out in Section 2.1 and as we shall see next and more formally in Appendix A.6.5, a spot service with only two prices, as is ultimately the case for a binary system, can be viewed as equivalent to the combination of an on-demand service and a preemptible instance service. In this case, the preemption probability (for jobs that bid at the low price) is given by the probability assigned to the high spot price. While obviously a simplification, this system still captures job heterogeneity along two key dimensions, value and delay sensitivity, and allows us to

incorporate correlation between them. As we shall see, the latter plays an important role in the structure of the optimal pricing configuration.

We start with a simple result that limits the number of prices the provider needs to consider in a binary system. For ease of exposition, we again relegate all proofs to appendices.

Lemma 4. *In a binary system with* $0 < v_1 < v_2$ *and* $0 \leq \kappa_1 < \kappa_2$*, the optimal pricing system needs at most two prices.*

Denote the two prices of a two-price strategy in a binary system as p_1 and p_2, where $0 < p_1 < p_2$, with π the probability of p_1 being selected[29]. Our goal is to characterize p_1, p_2, and π, as functions of $v_1, v_2, \kappa_1, \kappa_2$ and the correlation between them. In this base setting, our original question boils down to identifying under which conditions a bona fide two-price strategy (a strategy with two distinct prices, both advertised with non-zero probability, *i.e.,* $0 < \pi < 1$, and used for bidding purposes by different sets of customers) outperforms[30] all one-price strategies? As a starting point to this investigation, we first characterize the best optimal one-price strategy in a binary system.

Lemma 5. *In a binary system, the per unit of work expected revenue of the best one-price strategy is of the form*

$$R = \max\{v_1, (1-r)v_2\} \tag{2.10}$$

where $1 - r$ *is the fraction of* v_2 *jobs in the system.*

In other words, the best one-price strategy either has a price $p^* = v_1$, in which case all jobs bid at p^* for an expected unit revenue of v_1, or it has a price of $p^* = v_2$ that only v_2 jobs can afford, and correspondingly an expected unit revenue of $(1-r)v_2$.

[29] When $\pi = 0$ or 1, the system defaults to a single price system.

[30] For the sake of definiteness, a two-price strategy outperforms one-price strategies if and only if it yields a larger expected revenue per unit of work than any one-price strategy.

A two-price strategy will seek to leverage price discrimination to increase revenue. In particular, it will try to set p_1 and p_2 to extract close to v_1 and v_2 from jobs with value v_1 and v_2, respectively. Unfortunately, the introduction of a delay penalty makes this impossible. In particular, even if p_2 can be set close to v_2, p_1 will have to be set to a value lower than v_1 to compensate for the value lost to delayed execution when bidding at p_1 (a function of π).

In general, a two-price strategy will entice high-value, high-delay sensitivity jobs, *i.e.,* (v_2, κ_2) jobs, to bid at p_2, but high-value, low-delay sensitivity jobs, *i.e.,* (v_2, κ_1) jobs, will bid at p_1. Additionally, while low-value, low-delay sensitivity jobs, *i.e.,* (v_1, κ_1) jobs, also bid at p_1 (if set properly), low-value, high-delay sensitivity jobs, *i.e.,* (v_1, κ_2) jobs, will be priced out of the system as they can neither afford p_2, nor tolerate the delay associated with p_1.

Hence, for a two-price strategy to outperform the optimal one-price strategy of Lemma 5, revenue increases from some job categories must exceed the losses incurred from other categories. The outcome depends on both the structure of the best one-price strategy and the relative proportion of jobs from different types. In the next sub-section, we make this intuition more precise and characterize under which conditions a two-price strategy may be able to outperform the best one-price strategy. In particular, we highlight the role of the correlation coefficient ρ (see Eq. (A.4) in Appendix A.6.1 for a precise definition) between job value and delay sensitivity.

2.5.1 Characterizing Two-Price Strategies

To study the role of ρ, we consider binary systems with fixed marginals, and assume that jobs have value v_1 with probability r and delay sensitivity κ_1 with probability s. The next proposition characterizes the optimal pricing strategy in this configuration. In particular, it

identifies the presence of a correlation threshold ρ^*, which will be formally characterized in Corollary 7.

Proposition 6. [CORR_THRESH] *Given a binary system with fixed marginals for job value and delay sensitivity, either there exists a value ρ^* such that for $\rho \leq \rho^*$ a one-price spot service is optimal and for $\rho > \rho^*$ a two-price service is optimal, or else either one-price or two-price is optimal independent of ρ.*

As we shall see in Section 2.6, Proposition 6 appears to hold under more general conditions than those of this section.In particular, Section 2.6 considers three types of relaxations of our assumptions. They include allowing early termination of a job's bidding, *e.g.,* because of too many unsuccessful initial bids; considering (piece-wise linear) convex and concave rather than linear delay sensitivity functions; and finally allowing for more complex job profiles than those allowed by a binary system, *e.g.,* jobs whose value and delay sensitivity span a range of continuous values. Interestingly, while Proposition 6 appears robust to those relaxations, the same does not apply to results of Section 2.4. In particular, a fixed-price bidding policy does not remain optimal whenever terminations are allowed or delay sensitivity is not linear anymore.

In addition and as alluded to earlier and formalized in Appendix A.6.5, Proposition 6 can also be shown to be applicable to preemptible instances that trade-off a lower price for the possibility of interrupted/delayed execution. Alternatively, note that trading off a lower price for a longer execution time can also be realized by controlling the "speed" of the processors available for different types of instances, using either CPU throttling techniques [30] or hypervisor schedulers, *e.g.,* Xen's credit scheduler[31]. The challenge for the provider is then to select the set of processor speeds to offer so as to maximize its revenue based on customer

[31]Retrieved 2019, March 22 from `http://wiki.xen.org/wiki/Credit_Scheduler`.

affinities. In all these cases, customers trade-off a lower price for a possibly delayed job completion time.

Returning to Proposition 6, the next corollary shows that it can be strengthened and also offers an explicit expression for ρ^* that we subsequently use to gain additional insight.

Corollary 7. *Given a binary system with fixed marginals r and s for job value and delay sensitivity, and a given correlation coefficient ρ between job value and delay sensitivity, there exists a value $\rho^* \geq 0$ given by*

$$\rho^* = \frac{\max\{1-r, \frac{v_1}{v_2}\} - \frac{s(v_1\kappa_2 - v_2\kappa_1)}{v_2(\kappa_2-\kappa_1)} - (1-r)(1-s)}{\sqrt{rs(1-r)(1-s)}} \tag{2.11}$$

such that

- *when $\rho^* \in [0,1)$, then for $\rho \leq \rho^*$ a one-price spot service is optimal, and for $\rho > \rho^*$ a two-price spot service is optimal;*
- *otherwise a one-price spot service is always optimal independent of ρ.*

Both the proposition and the corollary state that a threshold exists in the correlation between value and sensitivity to delay, below which a spot price (two-price) solution never outperforms an on-demand service (one-price). In other words, unless a sufficient fraction of jobs are high-value and high-delay sensitivity, the market segmentation associated with a spot service does not improve revenue. This is consistent with our discussion at the beginning of this section, and is reasonably intuitive. A larger ρ implies a job market that is more clearly segmented between high-value, high-delay sensitivity jobs, *i.e.,* (v_2, κ_2) jobs, and low-value, low-delay sensitivity jobs, *i.e.,* (v_1, κ_1) jobs. This greater polarization of job profiles makes the price discrimination of the two-price strategy more effective. A similar intuition holds for preemptible instances/VMs and offerings of processors with different speeds.

In particular, from Proposition 2 and Lemma 28 in Appendix A.6, we know that under an optimal two-price strategy (p_1^*, p_2^*, π^*), (v_1, κ_1) and (v_2, κ_1) jobs will bid at p_1^*, while (v_2, κ_2) jobs will bid at p_2^*, and (v_1, κ_2) jobs will not bid. Hence, the revenue loss or gain that a two-price strategy generates over the best one-price strategy is a function of the relative proportion of those three types of jobs. Because ρ affects those proportions, it is intuitive that it should play a role in the efficacy of a two-price strategy.

Another property worth highlighting is that while ρ plays a role in determining whether a one-price or a two-price strategy yields a higher revenue, it does not influence the optimal prices. We state this result more explicitly in the next proposition.

Proposition 8. *In a binary system with given job values, delay sensitivities, and corresponding marginals, the prices used in the best one-price and two-price strategies are independent of the correlation coefficient ρ between job value and delay sensitivity.*

The proposition is made more explicit in the next sub-section, where we characterize the optimal two-price strategy and explore how it is affected by different factors, but we provide some intuition for the result. As just stated, under the optimal two-price strategy, jobs of type (v_1, κ_1) and (v_2, κ_1) bid at p_1^*, jobs of type (v_2, κ_2) bid at p_2^*, and jobs of type (v_1, κ_2) do not bid. The price p_1^* is set just low enough so as to entice the (v_1, κ_1) jobs to bid (at p_1^*) in spite of the resulting execution delays, while the price p_2^* is set so as to entice the (v_2, κ_2) jobs to bid at p_2^* rather than incur the delay penalty associated with bidding at p_1^*. Neither of those decisions are affected by the value of the correlation coefficient ρ, which only influences the relative proportion of jobs of each type, and therefore the revenue generated by the optimal two-price strategy. Hence, the result that the prices of the optimal two-price strategy do not depend on ρ.

Last, while it is obvious that when $\rho = -1$, *i.e.,* the only two types of jobs are (v_1, κ_2) and (v_2, κ_1), a two-price strategy can never be effective, the extent of ρ's influence is unclear. For that purpose, we explicitly explore next the case $\rho = 1$, *i.e.,* the only two types of jobs are (v_1, κ_1) and (v_2, κ_2), as a means of revealing the possible influence of other factors. This is the subject of the next proposition.

Proposition 9. *When in a binary system job value and delay sensitivity are perfectly positively correlated,* i.e., *the system only has* (v_1, κ_1) *and* (v_2, κ_2) *jobs, where* $0 < v_1 < v_2$ *and* $0 \leq \kappa_1 < \kappa_2$*, then using* s *to denote the fraction of* (v_1, κ_1) *jobs, we have*

- *When* $\kappa_2(1-s) - \kappa_1 > 0$ *and* $v_1\kappa_2 > v_2\kappa_1$*, a two-price spot service is optimal;*
- *Otherwise, a one-price spot service is optimal.*

The proposition highlights specific additional conditions that need to be satisfied for a two-price strategy to be effective even under perfect market segmentation. In particular, $\kappa_2(1-s) - \kappa_1 > 0$ implies $(1-s) > \frac{\kappa_1}{\kappa_2}$ and $v_1\kappa_2 > v_2\kappa_1$ gives $\kappa_1 < \kappa_2\frac{v_1}{v_2}$. Hence, the proposition states that for a two-price spot service to generate a higher revenue, the fraction $(1-s)$ of (v_2, κ_2) jobs must exceed a certain threshold, and the (v_1, κ_1) jobs must have a low enough delay sensitivity κ_1. This is again intuitive. The former ensures that enough jobs are willing to pay the higher price, while the latter indicates that the jobs that bid at the low price p_1 are sufficiently insensitive to delay to tolerate the long delay caused by the low winning probability of p_1 bids[32]. subsectionDiscussion and Insights

This section is devoted to exploring properties of the optimal two-price strategy, when it exists, and how it is affected by different system parameters. Our focus is on developing

[32]Recall that the average unit revenue from a job bidding at p_2 is $\pi p_1 + (1-\pi)p_2$. Keeping this unit revenue high, *i.e.,* close to v_2, typically calls for a small π, and consequently long delays for jobs bidding at p_1.

insight into how the optimal two-price strategy is affected by different system parameters, and we relegate the more technical exploration of how ρ^* varies as a function of those parameters to Appendix A.6.6.

Lemma 32 (see Appendix A.6) provides a starting point to this investigation in that it offers explicit expressions for the optimal two-price strategy, namely,

$$(p_1^*, p_2^*, \pi^*) = \left(\frac{\kappa_2 v_1 - \kappa_1 v_2}{\kappa_2 - \kappa_1}, v_2 + \kappa_2 - \epsilon, \ \frac{\kappa_2 - \kappa_1}{\kappa_2 - \kappa_1 + v_2 - v_1} \right), \tag{2.12}$$

where $0 < \epsilon$ is an arbitrary small but strictly positive quantity.

Structure of the Optimal Two-Price Strategy

To better understand the structure implied by Eq. (2.12), note that since a job bidding at p_2^* pays an expected unit cost of $\pi^* p_1^* + (1 - \pi^*) p_2^*$, simple algebraic manipulations yield that under this strategy its expected unit cost is $v_2 - \epsilon$. In other words, the pricing strategy extracts nearly all the value from jobs with unit value v_2 and bidding at p_2^*. Conversely, the expected unit cost of a job bidding at p_1^* is $p_1^* + \kappa_i(1/\pi^* - 1), i = 1, 2$, which after algebraic manipulations transforms to v_1 for jobs with delay sensitivity κ_1, *i.e.,* the pricing strategy extracts all the value from jobs with unit value v_1 and bidding at p_1^*. This is consistent with our earlier intuition that a two-price strategy would seek to leverage price discrimination to extract as much value as possible from jobs with different valuation.

Also of interest is the difference $\Delta_{21}^*(v_2, \kappa_2)$ in expected cost between bidding at p_2^* and bidding at p_1^* for (v_2, κ_2) jobs, *i.e.,* the type of jobs that a two-price strategy wants to see bidding at p_2^*. Specifically, using the fact that the costs of bidding at p_2^* and p_1^* are $\pi^* p_1^* + (1 - \pi^*) p_2^*$ and $p_1^* + \kappa_2(1/\pi^* - 1)$, respectively, simple algebraic manipulations yield

an expression of the form

$$\Delta^*_{21}(v_2, \kappa_2) = \epsilon \frac{(v_2 - v_1)}{\kappa_2 - \kappa_1 + v_2 - v_1} \tag{2.13}$$

In other words, under the optimal two-price strategy, the benefit for (v_2, κ_2) jobs of bidding at p_2^* over that of bidding at p_1^* has an infinitesimal margin, *i.e.,* $\Delta^*_{21}(v_2, \kappa_2) < \epsilon$.

In hindsight, this is reasonable. The service provider has three control parameters at its disposal, p_1, p_2, and π. Increasing the first two improves the value extracted from jobs bidding at those prices, but if set too high can either prevent jobs from bidding altogether or entice a high value job to bid at the lower price p_1 (if the delay penalty is low enough). The third parameter π controls the delay penalty of bidding at p_1. A large gap between the expected costs of bidding at p_2 and p_1 for (v_2, κ_2) jobs would allow the provider to increase both π and p_1, and therefore its revenue from (v_1, κ_1) jobs, while adjusting p_2 to remain revenue neutral (keep $\pi p_1 + (1 - \pi)p_2$ constant) for (v_2, κ_2) jobs.

The previous remarks notwithstanding, we also note that an optimal two-price pricing strategy need not always be feasible (let alone perform better than a one-price strategy). In particular, the expression for p_1^* yields $p_1^* < 0$ when $\frac{\kappa_2}{\kappa_1} < \frac{v_2}{v_1}$. In other words, for a two-price strategy to be effective, the relative delay penalty of delay sensitive jobs needs to exceed the relative value advantage of high value jobs. Consider an extreme scenario where this is violated, namely, $\kappa_1 = \kappa_2$. In this scenario, Proposition 2 (see also Proposition 23 in Appendix A.2) that states that a job's bid does not depend on its value or length, implies that the price discrimination ability of a two-price strategy is ineffective, and all jobs bid at only one price irrespective of the pricing strategy. In general, a small value for $\frac{\kappa_2}{\kappa_1}$, *i.e.,* a small difference in delay sensitivity, forces a small π to ensure that high value jobs bid at

p_2. As π becomes too small, the large delay penalty makes bidding at p_1 unattractive to v_1 jobs, while they cannot afford p_2. Hence, a two-price policy cannot be effective.

Another interesting property confirmed by Eq. (2.12) is that, as stated in Proposition 8, the optimal two-price strategy is independent of ρ. As discussed earlier, this is because ρ only affects the magnitude of the revenue realized under the optimal two-price strategy, and not *how* to realize it. Specifically, under fixed marginals, increasing ρ decreases the number of jobs of type (v_1, κ_2) and (v_2, κ_1) to proportionally increase the number of jobs of type (v_1, κ_1) and (v_2, κ_2). The increase in (v_1, κ_1) jobs is revenue neutral since those added jobs compensate for fewer (v_2, κ_1) jobs (they both bid at p_1^*), while the increase in (v_2, κ_2) jobs is revenue positive since those jobs replace (v_1, κ_2) jobs that previously did not bid. Hence, as ρ increases so does the revenue generated by the optimal two-price strategy, so that it may eventually exceed that of the optimal one-price strategy. This is illustrated in Figure 2.1, which plots the expected unit revenue under the optimal pricing strategy for a representative configuration. In this configuration, a one-price strategy is optimal when $\rho \leq 0.3$, but is outperformed by a two-price strategy when $\rho > 0.3$. Once ρ exceeds $\rho^* = 0.3$, revenue under a two-price strategy grows linearly[33] with ρ as the fraction of (v_2, κ_2) jobs increases.

Dependence on System Parameters

Next we explore the impact of job parameters, *i.e.,* v_1, v_2, κ_1, and κ_2, on the optimal two-price strategy (as per Lemma 5, their impact on the best one-price strategy is immediate). Differentiating Eq. (2.12) readily yields that p_1^* and π^* increase with v_1 and κ_2, and decrease

[33]Linearity is a direct consequence of basic algebraic manipulations after combining Eq. (2.11) and Lemma 32.

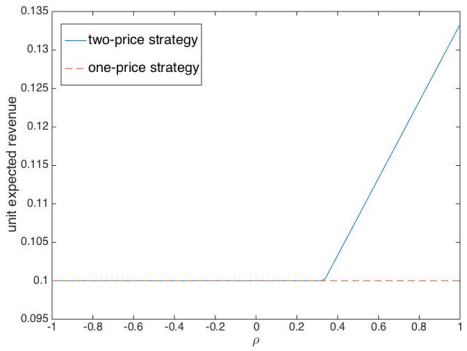


Figure 2.1: unit expected revenue for the optimal one-price and two-price strategies when $v_1 = 0.1$, $v_2 = 0.2$, $\kappa_1 = 0.1$, $\kappa_2 = 0.4$, $r = 0.5$ and $s = 0.5$.

with v_2 and κ_1:

$$\frac{\partial(p_1^*, \pi^*)}{\partial v_1} = \left(\frac{\kappa_2}{\kappa_2 - \kappa_1}, \frac{\kappa_2 - \kappa_1}{(\kappa_2 - \kappa_1 + v_2 - v_1)^2}\right), \quad \frac{\partial(p_1^*, \pi^*)}{\partial \kappa_1} = \left(\frac{-\kappa_2(v_2 - v_1)}{(\kappa_2 - \kappa_1)^2}, \frac{-(v_2 - v_1)}{(\kappa_2 - \kappa_1 + v_2 - v_1)^2}\right),$$
$$\frac{\partial(p_1^*, \pi^*)}{\partial v_2} = \left(\frac{-\kappa_1}{\kappa_2 - \kappa_1}, \frac{-(\kappa_2 - \kappa_1)}{(\kappa_2 - \kappa_1 + v_2 - v_1)^2}\right), \quad \frac{\partial(p_1^*, \pi^*)}{\partial \kappa_2} = \left(\frac{\kappa_1(v_2 - v_1)}{(\kappa_2 - \kappa_1)^2}, \frac{v_2 - v_1}{(\kappa_2 - \kappa_1 + v_2 - v_1)^2}\right). \tag{2.14}$$

Below we briefly develop some partial intuition behind these results.

A higher v_1 lets the service provider increase p_1^* because of the added value available to jobs that bid at p_1^*. Additionally, because a higher p_1^* makes it less desirable to bid at p_1^*, vs. bidding at p_2^*, this allows the provider to increase π^*, and correspondingly lower the delay penalty for bidding at p_1^*. This in turn can allow a further increase of p_1^*. Note though that, as alluded to earlier, keeping the combined impact of increasing π^* and p_1^* cost neutral for jobs bidding at p_2^*, *i.e.*, $\pi^* p_1^* + (1 - \pi^*)p_2^*$, will call for a slight decrease in p_2^*.

Turning to the impact of a higher κ_2, a greater delay sensitivity for κ_2 jobs allows the service provider to increase π^* without enticing (v_2, κ_2) jobs to switch to bidding at p_1^*. This increase in π^* again allows the service provider to increase p_1^*.

Considering next the effect of an increase in v_2, a higher v_2 allows the service provider to increase p_2^* and improve its expected revenue from (v_2, κ_2) jobs bidding at p_2^*. However, recalling Eq. (2.13), the small margin between bidding at p_2^* vs. p_1^* for those jobs, means that the provider must at the same time increase the cost of bidding at p_1^*. This can be done either by increasing p_1^* or by decreasing π^*. An increase in p_1^* calls for an increase in π^* to avoid losing (v_1, κ_1) jobs, but the resulting lower delay benefits κ_2 jobs more, and would therefore entice (v_2, κ_2) jobs to switch to bidding at p_1^*. Hence, increasing p_1^* is not an option. In contrast, while a decrease in π^* is accompanied by a decrease in p_1^* to avoid losing (v_1, κ_1) jobs, the greater sensitivity of κ_2 jobs to the increase in delay ensures that (v_2, κ_2) jobs keep bidding at p_2^*. Hence, an increase in v_2 results in a decrease in π^* and p_1^*.

Finally, a higher κ_1 value forces the service provider to either decrease p_1^* or increase π^* to keep attracting (v_1, κ_1) jobs. However, since an increase in π^* benefits κ_2 jobs more, it proceeds to decrease p_1^*. Furthermore, because the decrease in p_1^* makes bidding at p_1^* more attractive (to (v_2, κ_2) jobs), the service provider must also proceed to slightly decrease π^* to counter this effect. Hence, an increase in κ_1 also results in a decrease in π^* and p_1^*.

2.6 Robustness Evaluation

This section offers an assessment of the extent to which findings from the previous sections remain valid under more general conditions. Specifically and as detailed in the next subsection, we consider three "relaxations" of the assumptions used to derive the results of Sections 2.4 and 2.5. Recall that we assumed a linear delay penalty, meaning that when bidding started it ran until a job's completion without allowing for early termination, and that we derived our results on pricing strategies under the assumption of a binary system (job value and delay sensitivity only took two values). We now consider non-linear delay

sensitivity functions (piece-wise linear convex and concave) as well as the possibility of early job termination, *e.g.,* after too many unsuccessful bids. We also consider job profiles whose parameters are not limited to discrete values, but instead can span continuous ranges.

The primary targets of the investigation are the results of Propositions 1 (OPT_BID) and 6 (CORR_THRESH), namely, the optimality of a fixed bidding strategy and the existence of a correlation threshold below which a spot service offers no revenue benefits. As we shall see, Proposition 1 (OPT_BID) is relatively fragile but Proposition 6 (CORR_THRESH) appears to hold under more general conditions than those of Section 2.5, including not being limited to linear delay penalties and/or binary systems.

2.6.1 Assumptions, Relaxations, and Summary

We recall the main assumptions behind Propositions 1 (OPT_BID) and 6 (CORR_THRESH), and describe our plan for testing their robustness through different relaxations.

Assumptions

Proposition 6 (CORR_THRESH) relied on three major assumptions, with the first two also applicable to Proposition 1 (OPT_BID):

a1 Once an initial bid has been submitted for a job, bidding continues until the job completes;

a2 The penalty incurred for delaying the completion of a job is a linear function of the delay beyond the job's execution time;

a3 Job value and sensitivity to delay are both bimodal, *i.e.,* limited to high and low values.

Those assumptions were for the sake of tractability, but are obviously limiting in practice. For example, an abnormally long string of unlucky bids may delay a job's completion past the point where it is useful. The user may then decide to terminate the job and stop bidding. Conversely, some jobs may be mostly insensitive to a small amount of delay, and only start incurring a penalty once their delay exceeds a threshold. This violates the assumption of a linear delay penalty. Finally, while separating jobs based on high and low values and delay sensitivities captures basic job categories, greater diversity is likely to be the norm in practice. Hence, it is desirable to determine whether Propositions 1 (OPT_BID) and 6 (CORR_THRESH) still hold, when relaxing the above assumptions. We describe next our approach for investigating this question.

Relaxations

Specifically, we consider the following relaxations, one at the time:

r1 [*job termination*] The bidding process can be terminated prior to the successful completion of a job. Specifically, we allow jobs to terminate if the expected residual utility associated with continuing to bid is no longer positive;

r2 [*non-linear delay penalty*] The penalty associated with delaying the completion of a job beyond its execution time is a non-linear function of the delay. Specifically, we consider piece-wise linear convex and concave delay penalty functions;

r3 [*arbitrary job profiles*] Job profiles, *i.e.,* combinations of job value and sensitivity to delay, can span an arbitrary range of values and correspondingly follow arbitrary distributions. Specifically, we take job profiles where values and delay sensitivities are uniformly distributed over a continuous range.

Relaxation r1 calls for identifying a plausible termination criterion. A reasonable choice is, as alluded to earlier, for a user to terminate bidding when the job's expected residual utility is no longer positive. As shown in Appendix A.7.1 this can be decided through backward induction. The more challenging part is that this change in bidding strategy also affects the provider's choice of prices, *i.e.,* its pricing strategy. This coupling makes evaluating the impact of such a change on Propositions 1 (OPT_BID) and 6 (CORR_THRESH) challenging. As a result and for the sake of tractability, we adopt a step-wise approach to exploring this question.

We first proceed to identify the optimal bidding strategy for a given pricing strategy when job termination is allowed. As alluded to, this relies on backward induction, and while it provides some insight into the structure of the bidding strategy, it does not explicitly characterize it. It is, therefore, used to experimentally explore whether the outcome is a strategy that deviates from a fixed bidding strategy. Next, rather than attempt to characterize an optimal pricing policy, we instead numerically investigate for each system configuration (see Section 2.6.1) "all" possible pricing strategies, *i.e.,* triplets (p_1, p_2, π), as we vary the correlation ρ between job value and sensitivity to delay. In other words, for each value of ρ, we step through all combinations of p_1, p_2, and π, using a small, fixed step size (precision) in each dimension. The goal is to determine if a correlation threshold exists beyond which the best "one-price" strategy is outperformed by a "two-price" (more than one price) strategy. For simplicity, assumptions a2 and a3 are kept while investigating r1's impact.

Relaxation r2 is specialized to two specific non-linear delay penalty functions, namely, a piece-wise linear convex delay penalty, $D_1(\kappa, t)$, and a piece-wise linear concave delay penalty,

$D_2(\kappa, t)$, as follows:

$$\begin{aligned} D_1(\kappa, t) &= \kappa \ \max\{0, T(t) - \theta\}, \\ D_2(\kappa, t) &= \kappa \ \min\{T(t), \theta\}. \end{aligned}$$

where as before t is the job's service time, κ its delay sensitivity, $T(t)$ its total expected execution delay where for simplicity we have omitted dependency on v, κ and the bidding strategy Γ, and θ is a threshold value. In the case of D_1, this threshold corresponds to the execution delay the job can incur without suffering a penalty, while in the case of D_2 it is the amount of execution delay beyond which the job stops experiencing a penalty. Both convex and concave functions are plausible in practice. A convex delay penalty is representative of jobs that need to be completed within a certain deadline and only incur a penalty beyond it, *e.g.,* a product release or tax filings. Conversely, a concave delay penalty corresponds to jobs for which value is directly tied to timeliness of execution, but eventually stabilizes (possibly at 0) once delay exceeds a threshold, *e.g.,* computing stock trading prices or updating truck routes to minimize fuel consumption after traffic updates.

As for relaxation r1, identifying the optimal bidding strategy relies on backward induction, while assessing whether Proposition 6 (CORR_THRESH) still holds is again carried out by checking if and when the best one-price policy is outperformed by a two-price policy as correlation between job value and delay sensitivity is varied. For simplicity, assumptions a1 and a3 remain in effect while exploring the impact of relaxation r2.

Relaxation r3 follows a mostly similar approach as relaxations r1 and r2, namely, assumptions a1 and a2 are kept while job profiles now extend over continuous ranges for both job value and delay sensitivity. The investigation is, however, simplified by the fact that, as

Proposition 1 (OPT_BID) did not rely on assumption a3[34], a fixed bidding strategy remains optimal under relaxation r3. This facilitates testing whether Proposition 6 (CORR_THRESH) also holds.

Experimental range

When exploring relaxations r1 and r2, we reuse, with two exceptions (see Sections 2.6.2 and 2.6.3), the experimental configuration of Section 2.5. Specifically, we fix both v_1 and κ_1 to be 0.1, vary v_2 and κ_2 from 0.2 to 1 with a step size of 0.1, and choose the distribution marginals r and s independently from values in $\{0.2, 0.5, 0.8\}$. We also rely on the same bimodal job duration distribution where jobs have duration 1 with probability d, and 5 otherwise. Different values of d, namely, $d = 0.2, 0.5, 0.8$, are considered.

For each configuration across this range of parameters, we run a set of experiments where we vary the correlation between v and κ. Note that the range of infeasible correlations depends on the values of r and s. For example, when $r = s = 0.2$, a correlation of -1 is infeasible. This is because the fraction of (v_2, κ_2) jobs is at least 0.6, whereas a correlation of -1 implies that all jobs are either of (v_1, κ_2) or of (v_2, κ_1).

Summary

As mentioned above, Proposition 1 (OPT_BID) is unaffected by relaxation r3, irrespective of the (joint) distributions of the parameters of job profiles. However, as we shall see, it does not hold under relaxations r1 and r2, where experiments reveal that a dynamic bidding strategy dominates the fixed bidding strategy, though in most configurations the difference

[34]Section 2.4 assumes a given job (t, v, κ) and pricing strategy $(\boldsymbol{p}.\boldsymbol{\pi})$. Results derived in the section are, therefore, independent of the (joint) distributions of job parameters.

is small. Under a dynamic bidding strategy, users update bids in each slot based on their bidding history.

The situation is different when it comes to Proposition 6 (CORR_THRESH), as while it could a priori be affected by all three relaxations r1, r2, and r3, it remains empirically valid across all the scenarios we explore, and this irrespective of the users' bidding strategies (although we only report results for dynamic bidding strategies, we experimented with both fixed and dynamic strategies). In other words, the fact that correlation between job value and delay sensitivity must exceed a threshold for a spot service to improve revenue appears to hold in the presence of early terminations, under piece-wise linear convex and concave delay penalties, and beyond the limited setting of binary systems, *i.e.,* as shown with job profiles that span a continuous range under a uniform distribution.

For the reader's convenience, we summarize in table form below the set results established in the remainder of this section:

Relaxation	**Proposition 1** (OPT_BID)	**Proposition 6** (CORR_THRESH)
r1 [job termination when expected job utility stops being positive]	affected	holds
r2 [convex/concave piece-wise linear delay penalty]	affected	holds
r3 [uniform distribution in fixed range]	holds (for any distribution)	holds

2.6.2 Allowing Job Terminations

In this sub-section, we investigate the impact of relaxation r1 on Propositions 1 (OPT_BID) and 6 (CORR_THRESH), namely, whether they are affected by allowing customers to terminate jobs when a job's residual expected utility is no longer positive.

Proposition 1 (OPT_BID)

Given a job and pricing configuration, the optimal bidding strategy can be characterized by backward induction (see Appendix A.7.1) that accounts for how much delay a job has experienced so far as well as its residual service time to completion. This is because bidding decisions are now dependent on the sequence of spot prices a job encounters, as it affects its odds of termination. While this alone does not disprove Proposition 1 (OPT_BID), it hints at the possibility.

We explore this question using numerical experiments based on the system configurations introduced in Section 2.6.1, where for a given pricing we compute the optimal bidding strategy for each job type. Those experiments yielded several instances where the optimal bidding strategy deviated from fixed bidding, *i.e.,* Proposition 1 (OPT_BID) no longer held under relaxation r1. We provide next an illustrative example.

Consider the pricing strategy $(p_1, p_2, \pi) = (0.08, 0.6, 0.7)$ and job $(t, v, \kappa) = (5, 0.3, 0.4)$. The optimal bidding strategy computed by backward induction is found to be of the following form:

	$\widehat{t}=1$	$\widehat{t}=2$	$\widehat{t}=3$	$\widehat{t}=4$	$\widehat{t}=5$
$\widehat{T}=0$	p_2	p_2	p_1	p_1	p_1
$\widehat{T}=1$	p_2	p_1	p_1	p_1	p_1
$\widehat{T}=2$	p_1	p_1	p_1	p_1	0
$\widehat{T}=3$	p_1	0	0	0	0

where $\widehat{t}$ is the job's required residual service, $\widehat{T}$ is the job's current execution delay, and the value in entry $(\widehat{T},\widehat{t})$ corresponds to the optimal bid given the job's bidding history as captured by $(\widehat{T},\widehat{t})$. An entry value of 0 denotes job termination. A job's bidding decisions and outcomes determine its trajectory through the table (it starts in the top right corner and moves left/down after a winning/losing bid). The table illustrates that after enough unlucky bids, the job opts to terminate[35]. More importantly for our purpose, the table shows that while the job starts bidding at p_1, it may switch to p_2 if it experiences enough successful bids.

The strategy is somewhat intuitive given termination and the job's profile. The job starts bidding at p_1 given the relatively low value of p_1 and the high value of π, and the fact that the delay cost is bounded because termination is allowed. If the job is unlucky in its bids, *i.e.,* $\widehat{T}$ increases, then the job continues to bid at p_1 as the expected cost of bidding at p_2 remains too high, and termination limits its cost exposure when bidding at p_1. For example, when $(\widehat{T},\hat{t})=(3,1)$, a bid at p_2 has an expected residual utility of 0.064, while a bid at p_1 has an expected residual utility of 0.154. Conversely, when both $\widehat{T}$ and $\widehat{t}$ are small, switching to bidding at p_2 becomes attractive. The added cost of bidding at p_2 is only incurred over a small number of slots and ensures access to the full job value, while bidding at p_1 still carries the risk of termination.

[35]Not shown in the table is the fact that bidding always stops once $\widehat{T}$ exceeds 3.

Performance of non-termination strategies

The previous sub-section showed that when job can terminate before completion, fixed bidding strategies are no longer optimal. However, they remain attractive because of their simplicity. It is, therefore, of interest to explore the magnitude of the penalty incurred by using them while also forfeiting termination[36]. In carrying out this evaluation, we assume that the provider computes its pricing strategy assuming that jobs use optimal bidding strategies, including termination.

As in the previous section, we initially relied on the system configurations introduced in Section 2.6.1, where $\kappa_1 = 0.1$. However, these configurations all resulted in one-price optimal strategies, *i.e.,* scenarios where the topic of bidding is moot. The primary reason is that under termination, a two-price strategy cannot afford too small a value of π (recall that a small π implies a long execution delay when bidding at p_1). When π is large (and termination is allowed), even κ_2 users initially bid at p_1. This prevents two-price strategies from outperforming the best one-price strategy. Avoiding this calls for a small κ_1 value that in turn allows a small π value. We, therefore, repeated our experiments using $\kappa_1 = 0.01$.

The results illustrated that eliminating termination had little impact, except on *small* jobs, *i.e.,* $t = 1$. Specifically, the possibility of termination allows $(1, v_1, \kappa_2)$ jobs, *i.e.,* low-value, high delay sensitivity jobs, to attempt one bid at p_1. If they are lucky, they incur a cost of p_1 and no delay, and if their bid fails, they simply terminate, incurring no cost (and no value). Without the possibility of termination, they are forced to either bid at p_2 that they cannot afford, or bid at p_1 that has an expected delay that yields a negative utility.

[36]Note that termination decisions are also a form of dynamic bidding, and computing termination decisions has a similar computational cost as dynamic bidding.

A similar situation can exist for $(1, v_2, \kappa_2)$ jobs, for which forfeiting the possibility of termination can under certain configurations result in a significant reduction in utility (around 50% on average). This is because in those configurations, following the same strategy as $(1, v_1, \kappa_2)$ jobs, *i.e.,* bid at p_1 and terminate if unsuccessful, can yield a higher expected utility than bidding at p_2. Specifically, for reasons similar to those of Section 2.5.1, the provider sets p_2 so that a bid at p_2 leaves (v_2, κ_2) jobs only with a marginally positive utility. In contrast, a bid at p_1 followed by termination if unsuccessful yields an expected utility of $\pi(v_1 - p_1)$, which can be significantly larger.

Although termination can generate a larger expected utility for small jobs, it may not be a realistic option for all types of jobs. In particular, utility functions may be more complex than the stylized model assumed in this chapter. For example, dependencies may exist between jobs, *e.g.,* Map-Reduce, that would amplify the penalty associated with not completing a job. Termination could on the other hand be of benefit for recurring jobs, where missing one iteration may be of limited consequence since the result will be subsequently updated, *e.g.,* an incremental security scan.

Proposition 6 (CORR_THRESH)

We turn next to exploring the impact of relaxation r1 on Proposition 6 (CORR_THRESH). The approach is similar to that used for Proposition 1 (OPT_BID), *i.e.,* we numerically search across a range of configurations for the presence of a correlation threshold above which a two-price strategy becomes optimal. As in the previous sub-section, we use the configurations of Section 2.6.1, but with $\kappa_1 = 0.01$. The results of those experiments confirmed the validity of Proposition 6 (CORR_THRESH) even when terminations are allowed. Fig. 2.2 offers a representative example that plots ρ^* as a function of κ_2. As in the absence of termination, ρ^* decreases with κ_2. In our experiments, the possibility of termination also resulted in a higher

ρ^* value than when terminations were not an option. This is intuitive since termination can lower the revenue that the provider can extract when trying to entice jobs to choose a lower price in exchange of a delay penalty.

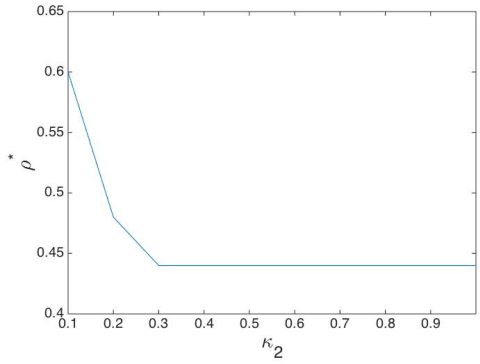


Figure 2.2: ρ^* as a function of κ_2 for dynamic bidding strategies with termination when $\kappa_1 = 0.01, v_2 = 0.2, r = 0.5, s = 0.5, d = 0.2$

2.6.3 Convex and Concave Delay Penalty Functions

In this sub-section, we investigate the impact of relaxation r2 on Propositions 1 (OPT_BID) and 6 (CORR_THRESH), namely, whether their results hold under non-linear delay penalty functions, *i.e.,* the convex and concave, piece-wise linear functions, $D_1(\kappa, t)$ and $D_2(\kappa, t)$, introduced earlier. As in Section 2.6.2, we first focus on Proposition 1 (OPT_BID), which as we shall see is again found not to hold. Hence, we also proceed to evaluate the impact on a job's utility of restricting itself to simpler fixed bidding strategies. Finally, we turn to Proposition 6 (CORR_THRESH), and whether it holds under convex and concave delay penalties.

The investigation is again numerical in nature, with experiments spanning a range of configuration combinations, now extended to include the threshold $\theta \in \{2, 5, 10\}$ used in the delay

penalty functions $D_1(\kappa, t)$ and $D_2(\kappa, t)$. For each combination, the correlation parameter ρ is varied within the range of feasible values. As before, and, with one exception, we rely on the job configurations introduced in Section 2.6.1. The exception is when investigating the impact of a convex delay penalty, for which it is necessary to modify the range of experiments under consideration, *i.e.,* by setting $\kappa_1 = 0.0001$ instead of $\kappa_1 = 0.1$.

The need for a small κ_1 under a convex delay penalty is intuitive. Under a convex delay penalty, a two-price strategy outperforms one-price strategies only if, in addition to the conditions discussed in Section 2.5, it also accounts for the fact that rather than systematically bidding at p_2, (v_2, κ_2) jobs bid at p_1 in their first θ slots[37]. Minimizing the revenue loss associated with those lower initial bids calls for selecting a very small π (the probability that p_1 is selected as the spot price). This in turn requires a correspondingly small value for κ_1 to mitigate the delay impact on (v_1, κ_1) jobs, and ensure they still generate a positive utility when bidding at p_1. Further, the need for a small κ_1 value increases with θ. For example[38], the κ_1 values for which a two-price strategy can outperform one-price strategies are $\kappa_1 \leq 0.006$ when $\theta = 2$, and $\kappa_1 \leq 0.0003$ when $\theta = 5$ or $\theta = 10$.

Proposition 1 (OPT_BID)

As the proof of Proposition 25 highlights, the optimality of a fixed bidding strategy depends heavily on the linearity of the delay penalty function. Hence, we expect that violating this property, as relaxation r2 does, will invalidate Proposition 1 (OPT_BID). As before, we rely on backward induction (see again Appendix A.7) to characterize the optimal bidding strategy, and readily find configurations where it deviates from a fixed bidding strategy. This holds for both $D_1(\kappa, t)$ and $D_2(\kappa, t)$. We provide illustrative examples next.

[37] They incur no penalty for up to θ slots.

[38] We set both the marginals r and s for v and κ equal to 0.2 and the probability that a job is small, *i.e.,* equal to 1 (vs. 5 for large jobs), to $d = 0.2$.

Example 1: Dynamic bidding under convex delay penalty Consider a pricing strategy $(p_1, p_2, \pi) = (0.09, 0.2, 0.4)$, and the convex delay function $D_1(\kappa, t)$ with $\theta = 2$. Using again the same notation, the optimal bidding strategy computed by backward induction for job $(t, v, \kappa) = (5, 0.2, 0.2)$ is of the form:

	$\hat{t} = 1$	$\hat{t} = 2$	$\hat{t} = 3$	$\hat{t} = 4$	$\hat{t} = 5$
$\widehat{T} = 0$	p_1	p_1	p_1	p_1	p_1
$\widehat{T} = 1$	p_1	p_1	p_1	p_1	p_1
$\widehat{T} = 2$	p_2	p_2	p_2	p_2	p_2

where bids for values of $\widehat{T} \geq 2$ are always at p_2.

This strategy is intuitive given a convex delay penalty with $\theta = 2$ and the job's profile. Bidding starts at the lower price, since there is initially no penalty in delaying a job's execution. However, after $\theta = 2$ failed bids, the desire to avoid a delay penalty takes precedence, and bidding switches to the higher price. This is because the expected additional cost of bidding at p_2 rather than p_1 is smaller than the expected delay penalty when bidding at p_1, *i.e.,* $(1 - \pi)(p_2 - p_1) < \frac{\kappa(1-\pi)}{\pi}$ or $0.066 < 0.3$.

Example 2: Dynamic bidding under concave delay penalty Consider next the pricing strategy $(p_1, p_2, \pi) = (0.06, 0.25, 0.5)$, the concave delay penalty $D_2(\kappa, t)$ with again $\theta = 2$, and a job with profile $(t, v, \kappa) = (5, 1, 0.2)$. The optimal bidding strategy computed by backward induction for the job is now of the form:

	$\widehat{t} = 1$	$\widehat{t} = 2$	$\widehat{t} = 3$	$\widehat{t} = 4$	$\widehat{t} = 5$
$\widehat{T} = 0$	p_2	p_2	p_1	p_1	p_1
$\widehat{T} = 1$	p_2	p_1	p_1	p_1	p_1
$\widehat{T} = 2$	p_1	p_1	p_1	p_1	p_1

where bids for values of $\widehat{T} \geq 2$ are always at p_1. More specifically, the optimal bidding strategy for job $(t, v, \kappa) = (5, 1, 0.2)$ starts by bidding at p_1, and then either continues at p_1 until completion, or, if experiencing enough early "lucky" wins, eventually switches to bidding at p_2 to complete the job.

To explain the change in bidding strategy, consider the job's trajectory through the optimal bidding strategy table. It starts in the upper right corner, $\widehat{T} = 0, \hat{t} = 5$, and moves through the table based on the outcome of its strategy. A successful bid, whether at p_2 or p_1, results in a move to the left to entry $(\widehat{T}, \hat{t} - 1)$. An unsuccessful bid (at p_1) results in a downward move to $(\widehat{T} + 1, \hat{t})$. Focusing on those downward transitions, the concave nature of the delay penalty implies that the expected delay penalty for continuing to bid at p_1 is now lower, while the expected cost penalty of bidding at p_2 is unchanged (the amount of residual work is the same). Hence, if bidding at p_1 was preferable it remains so after losing a bid. Contrast this with a successful bid, *i.e.,* a lateral move to $(\widehat{T}, \hat{t} - 1)$. In this case, the expected per slot cost penalty of bidding at p_2 is unchanged, but the expected per slot delay penalty goes up. This is because the job is no closer to the threshold beyond which there is no delay penalty, so that the relative weight of unlucky bids goes up (unlike a linear delay penalty, under D_2 only a fixed number of unlucky bids matter). This explains why the optimal strategy can switch to bidding at p_2 after enough lucky bids at p_1. Note that this also establishes that once a job starts bidding at p_2, it will continue until completion.

Returning to job $(t, v, \kappa) = (5, 1, 0.2)$, the combination $\kappa = 0.2$ and $\pi = 0.5$ results in a strategy that initially favors bidding at p_1 (the expected delay penalty is smaller than the expected cost penalty). Assume next a scenario where the job has enjoyed several lucky bids at p_1 and finds itself at $(\widehat{T}, \hat{t}) = (1, 2)$, at which point it experiences another successful bid at p_1 and moves to $(\widehat{T}, \hat{t}) = (1, 1)$. The strategy table tells us that the job should switch to bidding at p_2. As we have alluded to, this is because the expected delay penalty of bidding

at p_1 is now larger than the expected cost penalty of bidding at p_2. To verify this, we compute the expected residual costs of each strategy (sum of expected price plus expected delay penalty), before and after the last successful bit at p_1. In slot $(1, 2)$ the expected residual cost when bidding at p_2 is 0.31 vs. 0.2675 when bidding at p_1, *i.e.,* bidding at p_1 is indeed the better option. In contrast and, as expected, in slot $(1, 1)$ those costs are now 0.155 vs. 0.16, so that bidding at p_2 has become preferable.

Performance of fixed bidding strategies

The previous section illustrated that under non-linear delay penalties, dynamic bidding strategies are the solution of choice. However, because of the simplicity of fixed bidding strategies, it is again of interest to evaluate whether dynamic strategies warrant their added complexity. As before, in carrying out this evaluation, we assume that the provider sets its pricing strategy assuming that jobs seek to maximize their utility, *i.e.,* rely on dynamic bidding when it benefits them.

Performance under convex delay penalty As alluded earlier, experiments were conducted using the configurations of Section 2.6.1 but with $\kappa_1 = 0.0001$. In those experiments, switching from a dynamic to a fixed bidding strategy resulted in (v_2, κ_2) jobs refraining from bidding altogether. Recalling the discussion of Section 2.5.1 this is likely because of the very thin utility margin that the optimal two-price strategy leaves to jobs of type (v_1, κ_1) and (v_2, κ_2). This margin is computed assuming that (v_2, κ_2) jobs will bid at p_1 in their first θ slots before switching to bidding at p_2. Restricting jobs to fixed bidding policies increases the cost incurred by (v_2, κ_2) jobs, so that their expected utility becomes negative, which

prevents them from bidding[39]. Other job types are only marginally affected (on average less than 10%), since even under dynamic policies they limit themselves mostly to bidding at p_1.

In summary, under a convex delay penalty, limiting high-value, high-delay sensitivity jobs to fixed bidding strategies can have a significant impact on their utility, *i.e.,* prevent them from bidding altogether.

Performance under concave delay penalty For this investigation, we return to the original configurations of Section 2.6.1, *i.e.,* set $\kappa_1 = 0.1$.

The scenario from Example 2 in the previous section indicates that scenarios do exist where dynamic bidding can be beneficial. However, the example was constructed to establish that fixed bidding needs not be optimal for a given two-price configuration. The question we explore here is somewhat different, namely, to what extent does using fixed bidding hurt jobs utility when pricing is set assuming that jobs employ dynamic bidding when it improves their expected utility. Under those assumptions, we were unable to find configurations where dynamic bidding actually improved job's utility. In other words, the provider selected prices so as to discourage the behavior of Example 2. Recall that in Example 2 $(5, v_2, \kappa_2)$ jobs were bidding at p_1 unless they experienced enough successful bids, at which point they switched to p_2. A decrease in p_2 would succeed in discouraging those jobs from considering bidding at p_1 in the first place, which would in turn eliminate incentives for using dynamic bidding.

In short, it appears that in practice, dynamic bidding is of little benefit when the delay penalty is a concave function and the provider sets prices that optimize its revenue.

[39]Note that unlike the previous section, dynamic bidding alone does not enable $(1, v_1, \kappa_2)$ jobs to bid in spite of the initial insensitivity to delay afforded by θ. The inability to terminate unlucky jobs still prevents those jobs from bidding.

Proposition 6 (CORR_THRESH)

We next turn to exploring the impact of relaxation r2 on Proposition 6 (CORR_THRESH). The approach is similar to that used for Proposition 1 (OPT_BID), namely, we numerically evaluate whether the proposition holds across a range of configurations. The outcome was that either a one-price strategy was consistently optimal, or there existed a correlation threshold above which a two-price strategy became optimal. In other words, the results of the experiments were consistent with Proposition 6 (CORR_THRESH). We provide representative examples next.

Example 1: Dynamic bidding under convex delay penalty As explained earlier, in investigating a convex delay penalty, we rely on a value of $\kappa_1 = 0.0001$ in the configurations of Section 2.6.1. A representative outcome is reported in Fig. 2.3, which plots the value of

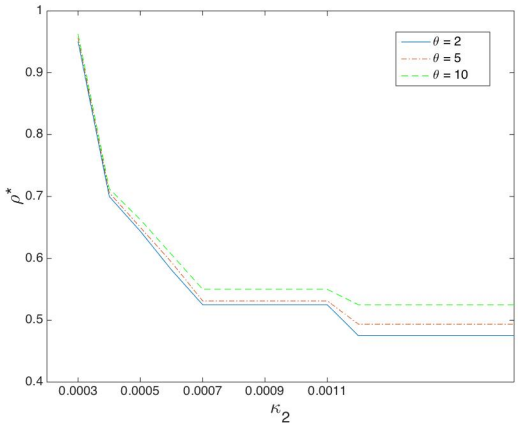


Figure 2.3: ρ^* as a function of κ_2 for convex delay penalty when $\kappa_1 = 0.0001$, $v_2 = 0.2$, $r = 0.3$, $s = 0.2$, $d = 0.2$, and $\theta = 2, 5, 10$

ρ^* as a function of κ_2 for different θ values ($\theta = 2, 5, 10$). The figure highlights two intuitive trends.

The first is that ρ^* is a decreasing function of κ_2 with a step-like behavior. The step-like behavior is simply a reflection of the relatively coarse users' bidding decisions, *i.e.,* no bid, bid at p_1, or bid at p_2. Switches from one decision to another are driven by increases in the user's sensitivity to delay κ. However, once a decision has been made, further increases in κ only reinforce it and never change it. Turning next to the decrease in ρ^* as κ_2 increases, note from the scale of the x-axis of the figure that it happens for very small values of κ_2 (and κ_1). In other words, in a range where κ_2 users are also mostly insensitive to delay. In that range, enticing (v_2, κ_2) users to bid at p_2 calls for a p_2 value that is very close to p_1. As κ_2 increases, so can p_2, which in turn increases the revenue derived from (v_2, κ_2) users, so that fewer of them (smaller ρ^*) are needed to outperform the best one-price policy.

The second trend revealed by Fig. 2.3 is that ρ^* increases with θ. This is again because under a convex delay penalty, a larger θ value lowers the revenue that can be extracted from (v_2, κ_2) users under a two-price policy (they bid at p_1 for longer). Offsetting this decrease requires increasing the number of such users, and consequently a higher ρ^* value.

Example 2: Dynamic bidding under concave delay penalty We carry out experiments for different θ values returning to the original job configurations introduced in Section 2.6.1, *i.e.,* $\kappa_1 = 0.1$. Results of a representative set of experiments are shown in Fig. 2.4 for different θ values. Note that the $\theta = 5$ line overlaps with the $\theta = 10$ line.

The figure plots ρ^* as κ_2 varies and other job distribution parameters are kept fixed. As expected, ρ^* is again a decreasing function of κ_2 with a step-like behavior due to the discrete users' bidding decisions. However, because θ plays an opposite role as under a convex penalty function, ρ^* now decreases with θ.

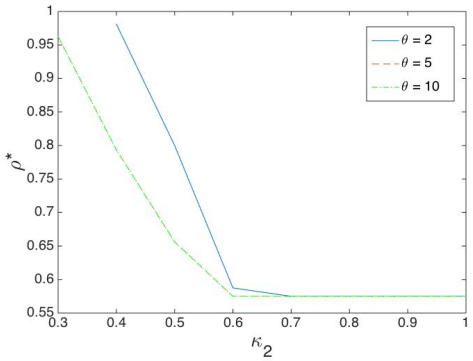


Figure 2.4: ρ^* as a function of κ_2 for concave delay penalty when $\kappa_1 = 0.1, v_2 = 0.2, r = 0.2, s = 0.2, d = 0.2$, and $\theta = 2, 5, 10$

2.6.4 Continuous Distributions

Our investigation has so far been limited to binary systems with two discrete values for each of v and κ, *i.e.,* four job types. In this sub-section, we instead assume that v and κ span a continuous range of values with uniform marginal distributions across those ranges. We assume that v has a uniform marginal distribution between $v_{\min}$ and $v_{\max}$, and that κ has a uniform marginal distribution between $\kappa_{\min}$ and $\kappa_{\max}$. The minimum values for job valuation and delay sensitivity ranges are set to $v_{\min} = 0.1$ and $\kappa_{\min} = 0$, respectively, and we vary the maximum values, $v_{\max}$ and $\kappa_{\max}$, from 0.2 to 1 and 0.1 to 1, respectively, both with a step size of 0.1. The correlation between v and κ is varied from -1 to 1 using a Gaussian copula [182, Table 2.1]. Reliance on copulae to capture dependencies across multiple random variables is relatively standard, *e.g.,* see [77, 82] for recent uses in related applications, and a range of copulae are available that can accommodate different dependency structures. Because we are interested in varying the correlation coefficient ρ from -1 to $+1$, we opted for a Gaussian copula that can accommodate this requirement. Note that because copulae produce distributions with uniform marginals between 0 and 1, we use linear transformations on v and κ to map the marginals to uniform distributions in $[v_{\min}, v_{\max}]$ and $[\kappa_{\min}, \kappa_{\max}]$.

Those mapping do not affect correlation. Because job size does not affect a job's bidding strategy, experiments are limited to jobs of size $t = 1$.

Since, as mentioned earlier, Proposition 1 (OPT_BID) is unaffected by the use of continuous distributions for v and κ, we focus our investigation on Proposition 6 (CORR_THRESH).

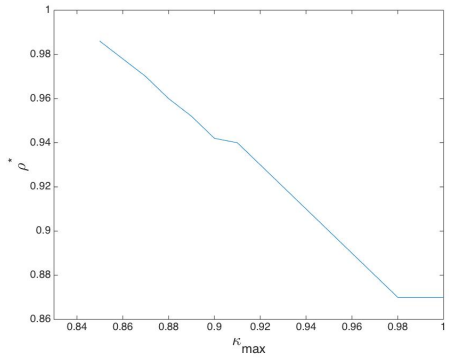


Figure 2.5: ρ^* as a function of $\kappa_{\max}$ for continuous distributions and $v_{\max} = 0.9$

Proposition 6 (CORR_THRESH)

Characterizing the optimal pricing strategy for continuous distributions is complex. For simplicity, we therefore limit pricing to one or two prices, and explore the validity of Proposition 6 (CORR_THRESH) by comparing the best one and two-price strategies and testing for the presence of a correlation threshold above which two-price strategies outperform the best one-price strategy. Specifically, we vary ρ from -1 to 1 with a step size of 0.02. For all $(v_{max}, \kappa_{max}, \rho)$ triplets, we then perform an exhaustive search for an optimal strategy across all possible combinations of p_1, p_2, and π, with a step size of 0.02. A two-price strategy is deemed optimal iff the search yields an optimal value for which $\pi \in (0, 1)$.

All experiments produced outcomes consistent with Proposition 6 (CORR_THRESH). Presenting the entirety of our experimental results calls for a three-dimensional graph (with

x and y axes associated with $v_{\max}$ and $\kappa_{\max}$, respectively, and ρ^* displayed on the z axis), which is difficult to visualize across all values of ρ^* . As a result, we instead report a representative outcome in the form of a two-dimensional projection for $v_{\max} = 0.9$. The results are shown in Fig. 2.5, which plots ρ^* as a function of $\kappa_{\max}$. The figure shows that ρ^* is a decreasing function of $\kappa_{\max}$. This is consistent with our earlier findings (and the discussion of Section A.6.6) that in a binary system ρ^* decreases with κ_2. Increasing $\kappa_{\max}$ plays a similar role, as it shifts the distribution of job's delay sensitivities towards higher values.

2.7 Summary

The chapter investigates the potential benefits (to users and cloud providers) of offering a combination of services that realize a different trade-off between cost and timeliness of job completion. Delays in a job's completion are caused by service interruptions that put the job on hold for a period of time. Spot services and preemptible instances are both examples of such services, even if they differ in their realization. Spot services rely on dynamic pricing[40] that adds a bidding dimension to the user's decision of when and how to use the service. In exchange for this added complexity, the user is offered some control on service interruptions, *i.e.,* higher bids are less likely to be interrupted. This aspect is absent in preemption based services that rely on fixed prices, with preemption decisions exogenous to the user and preemption odds available only through rough estimates, either made available by the provider or obtained from empirical data. In both cases though, the main question of interest is whether such a combination of services can help the provider increase revenue by better exploiting market segmentation.

[40] As mentioned before, the recent changes made by Amazon to their spot pricing offering has significantly reduced both the range and frequency of spot price changes. As stated in Section 3.1 and argued in [99], the current offering is now much closer to Google's and Microsoft's fixed price versions and in the process has lost some of the benefits that variable prices offered.

Towards investigating this question, the chapter makes the assumption that due to its scale the cloud rarely experiences capacity constraints. As a result, service prices are not responsive to demand, and used primarily towards improving revenue. Under this assumption and relying, for tractability, on a simple model for job profiles and how they differ in their willingness to pay and sensitivity to execution delays, the chapter establishes a number of interesting results.

In particular, we find that offering services that allow trading a lower cost for a potentially longer execution time (through service interruptions because of spot price variations or preemptions) is beneficial *only* if the correlation between job valuation and sensitivity to delay is high enough, *i.e.,* most high-value jobs require timely executions. In such a setting, a simple service offering with a set of properly chosen increasing spot prices that map to corresponding decreases in the probability of a job being interrupted (*i.e.,* because of a bid that falls below the spot price, can successfully extract most of the market value). The result also extends to services such as preemptible instances, even if they rely on fixed rather than dynamic prices. The common feature that links spot and preemptible instances together is that both offer a lower cost service option (compared to on-demand instances) in exchange for potential delays in a job's completion (in addition to requiring that the job be able to accommodate interruptions). Numerical investigations showed that the results appear to hold even in the presence of more complex job profiles than those used to derive them.

The chapter's other main result is specific to a spot service. It establishes that under certain assumptions regarding job profiles, *i.e.,* a linear sensitivity to execution delay and a policy to run jobs to completion once they start executing, a simple fixed bidding strategy can be optimal for users. Unlike the previous finding, the result is somewhat fragile, with, for example, allowing early termination of jobs and/or the use of non-linear delay sensitivity functions, both resulting in dynamic bidding policies outperforming a simple, fixed bidding policy.

Those differences notwithstanding, the gap between fixed and dynamic bidding policies was found to be relatively small across most of our experiments.

Chapter 3

Traffic Scheduling and Shaping for Inter-data Center Networks

3.1 Introduction

The networks that connect datacenters are both similar and yet very different from the public Internet. On the one hand, they rely on the same suite of protocols so that given sizes that often exceed those of major Internet Service Providers (ISPs) networks [128, 169, 109, 141, 181], they now make up a significant fraction of IP traffic worldwide [55, 106]. On the other hand, unlike the public Internet where providers have limited information and control on end-user traffic, datacenter operators have explicit contractual relationships with their users/customers. These are typically in the form of traffic contracts such as token buckets [189, Section 4.2], and Service Level Objectives/Agreements (SLOs/SLAs) expressing rate and latency targets. When combined with the centralized control that technologies

such as Software Defined Networking (SDN) enable [110, 186, 192], fine tuning of performance is now not only possible [42, 96, 160, 188], but also highly desirable to minimize cost [129, 193].

The work assumes such an environment with datacenters inter-connected by links under the purview of an operator with both knowledge and control of individual flows (or flow aggregates) traversing the network. The goal in this setting is then to identify the minimum network capacity (bandwidth of inter-datacenter links) required to meet the performance targets of individual flows (or flow aggregates). In particular, datacenter traffic commonly maps to a range of profiles and associated latency bounds that lend themselves to careful optimization [129, 193], with the consolidation and orchestration of those requests in a central controller offering the opportunity for significant improvements in efficiency [5]. Specifically, the work seeks to answer the following question: *"What is the minimum network bandwidth required to meet the latency targets of a given set of flows?"* The answer obviously depends on the type of scheduling mechanisms in use, and we consider options of varying complexity. We note that this question is the dual of the traditional *call admission* problem that seeks to assess whether performance goals can be met *given* the available network capacity. The added complexity in answering this dual problem is in the exploration of the space of possible configurations in handling individual flows.

This work starts with the most basic of network configurations, namely, one that involves a *single node and link* (hop), and then extends the investigation to the more general multi-hop case.

The work's contributions are in formulating optimal solutions for schedulers of different complexity in the single hop case, and in proposing algorithms to improve network efficiency on the multiple hop case. This work first characterizes the optimal solution (minimum

bandwidth for a given set of flows) and how to realize it with a dynamic priority (service curve-based) scheduler. The chapter then explores static priority and first-in-first-out (fifo) schedulers that are considerably easier to implement. Of interest is the "cost of simplicity" in terms of the additional bandwidth required. In addition, while an optimal scheduler by definition leaves no room for improvements, the same does not hold for simpler schedulers. In particular, both static priority and fifo schedulers may benefit from modifying flow profiles *prior* to offering them to the scheduler. Towards leveraging this insight, the chapter identifies optimal ingress traffic (re)shaping configurations that minimize the link bandwidth required to meet flows' deadlines (inclusive of shaping delays) for both such schedulers. The relative benefits of such an approach and the extent to which it helps those simpler schedulers approximate the performance of an optimal scheduler is then evaluated for a range of flow configurations. After that, this work then proposes reshaping mechanisms to improve network efficiency in the multiple hop case based on its single hop results.

The rest of this part is structured as follows. Section 3.2 offers a brief review of related works, both from the perspective of the problem under consideration and the techniques on which its relies. Section 3.3 focuses on the single hop case. It introduces our one-hop "network" mode, formulates and solves our targeted optimization under dynamic priority, static priority and fifo scheduler. Based on that, it proposes reshaping mechanisms for static priority and fifo, and then explores their performance through numerical experiments. Section 3.4 presents reshaping algorithms for the multi-hop case, and explores their performance through numerical experiments. Section 3.5 summarizes the chapter's findings and identifies extensions. Proofs and a few additional ancillary results are provided in an extensive set of appendices that complete the thesis.

3.2 Related Works

There is a vast literature aimed at traffic engineering and scheduling within data centers. However, while some of the mechanisms put forth in those works may be applicable to our setting, their scope and focus typically differ from ours. Our concern is with the wide area network (WAN) connecting datacenters as opposed to internal datacenter networks, and our primary goal is to minimize its cost (under performance constraints) rather than optimize performance. From that perspective, works such as [208, 5, 29] and [126] are closest conceptually to our goals.

Reducing costs while meeting latency performance requirements is a goal we share with [208], and so is our reliance on achieving this goal through careful assignment of workload priorities and reshaping options. The main difference is in the cost parameters under consideration, and consequently the criteria they give rise to. More specifically, [208] is concerned with minimizing the number of servers needed to accommodate a workload with given deadlines, whereas our focus is on minimizing network bandwidth given a set of inter-datacenter flows and deadlines. As a result, [208] selects token-bucket parameters to optimize workflow co-location across servers, while we set them based on their impact on end-to-end flow deadlines and their ability to constrain flow interactions in the network; hence minimizing bandwidth.

Lowering network cost while meeting performance targets is also a goal of [29], albeit in the form of availability rather than latency. Because latency is not a concern, it departs from our focus by not considering the possibility of reshaping traffic flows (through adjusting token bucket parameters). In contrast, how to best reshape flows is very much a concern of [126], as is minimizing (network) cost while meeting target deadlines. Specifically, [126] relies on a network/link bandwidth cost function (based on load percentile) that creates opportunity for periods of "free" bandwidth. Its goal is then to make shaping and scheduling decisions that

can maximize the amount of traffic sent during such free periods, and consequently lower network cost. Our goal is both simpler and more complex. It is simpler in that we only seek to minimize link capacity as a substitute for network cost. It is more complex because (ingress) reshaping decisions affect how flows interact, which impacts their deadlines. This aspect is absent from [126] that focuses on meeting long-term traffic volume targets on a single link (the ISP link whose cost is to be optimized).

On the modeling front, the theory of "network calculus" (NC) [120] is of most relevance. The NC framework involves deterministic traffic envelopes, as produced by token buckets controllers, and offers a powerful tool to analyze end-to-end network performance. In particular, a number of recent works have provided increasingly tight end-to-end delay bounds for different types of schedulers and network topologies, *e.g.,* feed-forward networks and topologies that include cycles, with the latter being typically significantly harder to handle (see [35, 39, 40, 36, 37] for a comprehensive review or relevant techniques, [32] for a detailed discussion of the two main families of solution techniques, including the more recent optimization-based framework, and [207] for a recent survey of tools based on them).

Also of note is the IEEE 802.1$*$ set of standards on "time sensitive networking" (TSN) [69] that builds on the NC framework and provides specifications and mechanisms to upper-bound the end-to-end latency for different traffic flows across a packet network. Within that framework, a number of results [119, 143, 202] have been derived on end-to-end bounds for specific types of rate-controllers, *e.g.,* interleaved (also called asynchronous traffic shaping) and credit based shapers, and schedulers, including first-come-first-served (FCFS) and class-based schedulers.

The relevance of these works notwithstanding, they all address the *dual* of the problem tackled in this chapter, which instead aims to determine the minimum amount of network

resources (bandwidth) required to meet given delay bounds for a set of flows. Besides, though much of NC's value manifests itself in the multiple-node setting where it offers a powerful tool to analyze end-to-end network performance, we only rely on NC to calculate the worst-case delay in the single-node setting. In the multiple-node setting, we characterize the end-to-end worst-case delay by simply adding up all the worst-case single hop delay. We acknowledge that this simple method, who does not benefits from the Pay Bursts Only Once phenomenon, will result in a looser delay bound, and we made this decision due to two reasons. In this work, we focus on dynamic priority, static priority, and fifo schedulers. For dynamic priority, no existing work provides close-form end-to-end bounds to our knowledge, and deriving such a bound, which is complicated by itself, is beyond the scope of this work. For static priority and fifo, existing results provide very complicated, if any, close-form bounds. Since our methods rely on an optimization whose constraints involving the end-to-end bounds, adopting such complicated bounds will make it impractical to solve the optimization.

3.3 Single-hop Case

In this section, we study the single hop case. Section 3.3.1 introduces our one-hop "network" model and the optimization we seek to solve, with the next three sections devoted to deriving optimal solutions for schedulers of different complexity. Section 3.3.2 relies on a general, dynamic priority scheduler, while Sections 3.3.3 and 3.3.4 consider simpler static priority and fifo schedulers. In the latter two cases, configurations both without and with (ingress) reshaping of flows prior to entering the network are considered. Section 3.3.5 quantifies the relative benefits of each approach, starting with a simple "two-flow" configuration that helps identify trends, before considering more general multi-flow scenarios.

3.3.1 Model Formulation

In this section, we formulate our problem as an optimization problem (**OPT**), which we proceed to solve in Sections 3.3.2 to 3.3.4 under different assumptions regarding the scheduling mechanism available in the "network."

Consider a network used to transport n flows. Flow $i, 1 \leq i \leq n$, is associated with an end-to-end[41] packet-level deadline d_i, where w.l.o.g. we assume $d_1 > d_2 > \ldots > d_n$ with $d_1 < \infty$. The traffic generated by flow i is rate-controlled using a two-parameter token-bucket (r_i, b_i), where r_i denotes the token rate and b_i the bucket size. The *profile* of flow i is then defined as (r_i, b_i, d_i). Our goal is to meet the latency requirements (deadlines) of all n flows at the lowest possible network "cost."

Network cost clearly involves many factors. Our focus is on link bandwidth, so that minimizing network cost maps to minimizing some function of link bandwidth. In a general network setting, network (bandwidth) cost depends on both the bandwidth of individual links and the number of links. The latter typically grows linearly[42], and while the former is often in practice a "step function" as bandwidth increases[43], it can be reasonably captured through a linear interpolation. In such scenarios, the sum of link bandwidths, *i.e.,* total network bandwidth, represents a reasonable optimization metric though others, *e.g.,* the maximum link bandwidth or a weighted sum of link bandwidths, are certainly possible.

In the simple case of one-hop (one link) networks considered in this chapter, this "sum" defaults to a single term. In this basic setting, denote as R the bandwidth of our single link,

[41] The deadline measures the time between transmission by the source end-system to reception by the destination end-system.

[42] Longer, more expensive links can be easily mapped to sequences of equal cost links connected in series.

[43] Retrieved 2020, September 27 from `https://enterprise.verizon.com/service_guide/reg/cp-gdl-rates-charges.pdf`.

and define $\boldsymbol{r} = (r_1, r_2, \ldots, r_n)$, $\boldsymbol{b} = (b_1, b_2, \ldots, b_n)$, and $\boldsymbol{d} = (d_1, d_2, \ldots, d_n)$ the vectors of the rates, burst sizes, and deadlines of the flows sharing the link. For notational simplicity we omit the scheduler type in the expression for flow i's *worst-case* end-to-end delay, $D_i^*(\boldsymbol{r}, \boldsymbol{b}, R)$. Our optimization constraint is then $D_i^*(\boldsymbol{r}, \boldsymbol{b}, R) \leq d_i, \forall i, 1 \leq i \leq n$, *i.e.,* all flows meet their deadline, and our optimization **OPT** is of the form:

$$\begin{aligned} \textbf{OPT} \quad & \min \parallel R \parallel_1 \\ \text{s.t} \quad & D_i^*(\boldsymbol{r}, \boldsymbol{b}, R) \leq d_i, \quad \forall i, 1 \leq i \leq n \end{aligned} \tag{3.1}$$

As propagation and processing delays are independent of link bandwidth, we focus on the contributions of queueing and transmission delays on a flow's end-to-end delay. As a result, in this simple scenario, a flow's deadline represents the time between a bit arriving at the node and being transmitted on the shared link.

The next three sections explore solutions to **OPT** under different combinations of schedulers in the simple one-hop (single link) network of Fig. 3.1. For further simplicity, the n flows sharing that network link are assumed connected through separate, dedicated access links of very high bandwidth. The (output) link bandwidth R is then the quantity **OPT** aims to minimize while meeting the n flows' deadlines. Further, the link is preceded by an infinite buffer and has capacity that exceeds the aggregate average arrival rate across all flows, so that the system is stable and lossless.

As alluded to earlier, of interest is the extent to which more sophisticated (expensive) schedulers translate into lower bandwidth values. For simplicity of exposition and analysis, results are presented using a fluid rather than packet-based model. Appendix B.6 derives a solution for a static priority scheduler under a packet-based model, but the results do not contribute additional insight.

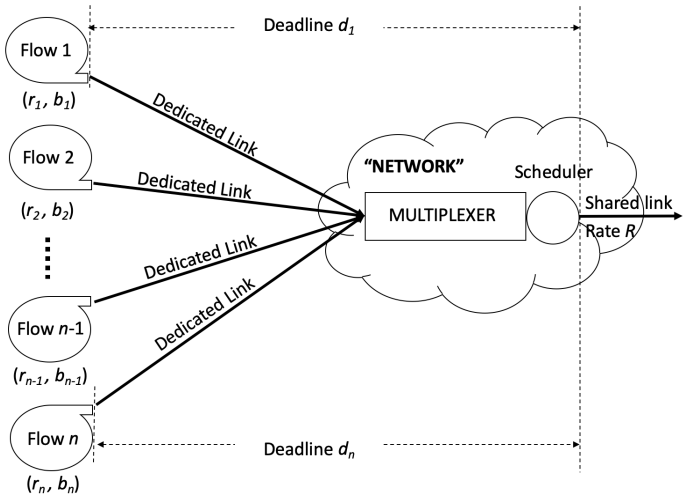


Figure 3.1: A typical one-hop configuration with n flows.

3.3.2 Dynamic Priorities

We start with the most powerful but most complex mechanism, dynamic priorities, where priorities are derived from general service curves assigned to flows as a function of their profile (deadline and traffic envelope). We then solve **OPT** to characterize the service curves that achieve the lowest bandwidth while meeting all deadlines.

Towards deriving this result, we first specify a service-curve assignment Γ_{sc} that satisfies all deadlines, identify the minimum link bandwidth R^* required to realize Γ_{sc}, and show that any scheduler requires at least R^*. We then show that an earliest deadline first (EDF) scheduler realizes Γ_{sc} and, therefore, meets all the flow deadlines under R^*.

Proposition 10. *Consider a one-hop network shared by n token-bucket controlled flows, where flow $i, 1 \leq i \leq n$, has a traffic contract of (r_i, b_i) and a deadline of d_i, with $d_1 > d_2 > ... > d_n$ and $d_1 < \infty$. Consider a service-curve assignment Γ_{sc} that allocates flow i a service*

curve of

$$SC_i(t) = \begin{cases} 0 & \text{when } t < d_i, \\ b_i + r_i(t - d_i) & \text{otherwise.} \end{cases} \tag{3.2}$$

Then

1. *For any flow $i, 1 \leq i \leq n$, $SC_i(t)$ ensures a worst-case end-to-end delay no larger than d_i.*

2. *Realizing Γ_{sc} requires a link bandwidth of at least*

$$R^* = \max_{1 \leq h \leq n} \left\{ \sum_{i=1}^{n} r_i, \frac{\sum_{i=h}^{n} b_i + r_i(d_h - d_i)}{d_h} \right\}. \tag{3.3}$$

3. *Any scheduling mechanism capable of meeting all the flows' deadlines requires a bandwidth of at least R^*.*

The proof of Proposition 10 is in Appendix B.2.1. The optimality of Γ_{sc} is intuitive. Recall that a service curve is a lower bound on the service received by a flow. Eq. (3.2) assigns service to a flow at a rate exactly equal to its input rate, but delayed by its deadline, *i.e.,* provided at the latest possible time. Conversely, any mechanism $\hat{\Gamma}$ that meets all flows' deadlines must by time t have provided flow i a cumulative service at least equal to the amount of data that flow i may have generated by time $t - d_i$, which is exactly $SC_i(t)$. Hence the mechanism must offer flow i a service curve $\widehat{SC}_i(t) \geq SC_i(t), \forall t$.

Next, we identify at least one mechanism capable of realizing the services curves of Eq. (3.2) under R^*, and consequently providing a solution to **OPT** for schedulers that support dynamic priorities.

Proposition 11. *Consider a one-hop network shared by n token-bucket controlled flows, where flow $i, 1 \leq i \leq n$, has a traffic contract of (r_i, b_i) and a deadline of d_i, with $d_1 > d_2 >$*

... > d_n and $d_1 < \infty$. The earliest deadline first (EDF) scheduler realizes Γ_{sc} under a link bandwidth of R^*.

The proof of Proposition 11 is in Appendix B.2.2. We note that the optimality of EDF is intuitive, as minimizing the required bandwidth is the dual problem to maximizing the schedulable region for which EDF's optimality is known [75].

Note that Γ_{sc} specifies a non-linear (piece-wise-linear) service curve for each flow. Given the popularity and simplicity of linear service curves, *i.e.,* rate-based schedulers, it is tempting to investigate whether such schedulers, *e.g.,* GPS [150], could be used instead. Unfortunately, it is easy to find scenarios where linear service curves perform worse.

Consider a one-hop network with only two flows with profiles $(r_1, b_1, d_1) = (1, 45, 10)$ and $(r_2, b_2, d_2) = (1, 5, 1)$. For a flow in isolation, a rate-based scheduler requires a bandwidth of $\max\left\{\frac{b}{d}, r\right\}$ to meet the deadline of a flow with profile (r, b, d). Applying this to flow 2 that has the tighter deadline calls for a bandwidth of 5 to meet its deadline. After 1.25 units of time (the time to clear the initial burst of 5 and the additional data that accumulated during its transmission), flow 2's bandwidth usage drops down to $r_2 = 1$. The remaining 4 units become then available to flow 1. This means that the initial dedicated bandwidth needed by flow 1 to meet its deadline of 10 given its initial burst of45 is equal to 1[44], for a total network bandwidth of 6 units. In contrast, Eq. (3.3) tells us that Γ_{sc}, only requires a bandwidth of $R^* = 5.9$. This difference, albeit relatively small in this toy example, illustrates the advantage of EDF over rate-proportional policies through the greater flexibility it affords in deciding how to allocate link bandwidth.

[44]The dedicated bandwidth c_1 of flow 1 must satisfy $45 - \frac{5}{4}x - (x+4)\left(10 - \frac{5}{4}\right) = 0$ and hence $c_1 = 1$ to ensure that the original burst of size 45 is cleared by the deadline $d_1 = 10$.

In the next section, we consider the use of simpler, static priority schedulers; first used alone (Section 3.3.3) and then combined with an ingress (re)shaper (Section 3.3.3). As discussed in Section 3.3.3, the introduction of ingress shapers is motivated by the need to limit the impact of high-priority flows on lower priority ones. As formalized in the next proposition, this potential benefit from ingress (re)shaping of flows is absent when dynamic priority (service curve-based) schedulers are used. This is intuitive given the optimality of such schedulers.

Proposition 12. *Consider a one-hop network shared by n token-bucket controlled flows, where flow* $i, 1 \leq i \leq n$*, has a traffic contract of* (r_i, b_i) *and a deadline of* d_i*, with* $d_1 > d_2 > ... > d_n$ *and* $d_1 < \infty$*. Adding ingress (re)shapers will not decrease the minimum bandwidth required to meet the flows' deadlines.*

The proof is in Appendix B.2.3.

3.3.3 Static Priorities

Though dynamic priorities are efficient and may be realizable [168, 163], they are expensive and not feasible in all environments. It is, therefore, or interest to explore simpler alternatives to offer service differentiation, and to quantify the resulting trade-off between efficacy and complexity. For that purpose, we consider next a static priority scheme with flows assigned a fixed priority as a function of their deadline.

As before, we consider a single-hop scenario with n flows with profiles $(r_i, b_i, d_i), 1 \leq i \leq n$, sharing a common network link. The question we first address is how to assign (static) priorities to each flow given their profile and the goal of **OPT** of minimizing the link bandwidth required to meet all deadlines? The next proposition offers a partial and somewhat intuitive

answer to this question by establishing that the minimum link bandwidth can be achieved by giving flows with shorter deadlines a higher priority. Formally,

Proposition 13. *Consider a one-hop network shared by n token-bucket controlled flows, where flow $i, 1 \leq i \leq n$, has a traffic contract of (r_i, b_i) and a deadline of d_i, with $d_1 > d_2 > ... > d_n$ and $d_1 < \infty$. Under a static-priority scheduler, there exists an assignment of flows to priorities that minimizes link bandwidth while meeting all flows deadlines such that flow i is assigned a priority strictly greater than that of flow j only if $d_i < d_j$.*

The proof is in Appendix B.3.1. We note that while Proposition 13 states that network link bandwidth can be minimized by assigning flows to priorities in the order of their deadline, it neither rules out other mappings nor does it imply that flows with different deadlines should always be mapped to distinct priorities. For example, when deadlines are large enough so that they can be met by assigning all flows their average rate, then the ordering of priority often does not matter. More generally, in some scenarios, grouping flows with different deadlines in the same priority class can result in a lower bandwidth than if they are mapped to distinct priority classes[45]. Nevertheless, motivated by Proposition 13, we propose a simple assignment rule that strictly maps lower deadline flows to higher priorities, and proceed to evaluate its performance.

Static Priorities without (re)Shaping

From [120, Proposition 1.3.4] we know that when n flows with traffic envelopes $(r_i, b_i), 1 \leq i \leq n$, share a network link of bandwidth $R \geq \sum_{i=1}^{n} r_i$ with flow i assigned to priority i, then, under a static-priority scheduler, the worst case delay of flow h is upper-bounded by $\frac{\sum_{i=h}^{n} b_i}{R - \sum_{i=h+1}^{n} r_i}$ (recall that under our notation, priority n is the highest). As a result, the

[45] We illustrate this in Appendix B.5 for the case of two flows sharing a static priority scheduler. Extending the result to a general scenario with n flows is left to future work.

minimum link bandwidth $\widetilde{R}^*$ to ensure that flow h's deadline d_h is met for all h is given by:

$$\widetilde{R}^* = \max_{1 \leq h \leq n} \left\{ \sum_{i=1}^{n} r_i, \frac{\sum_{i=h}^{n} b_i}{d_h} + \sum_{i=h+1}^{n} r_i \right\} \tag{3.4}$$

Towards evaluating the performance of a static priority scheduler compared to one that relies on dynamic priorities, we compare $\widetilde{R}^*$ with R^* through their relative difference, *i.e.,* $\frac{\widetilde{R}^* - R^*}{R^*}$. For ease of comparison, we rewrite R^* as

$$R^* = \max_{1 \leq h \leq n} \left\{ \sum_{i=1}^{n} r_i, \frac{\sum_{i=h}^{n} b_i}{d_h} + \sum_{i=h+1}^{n} r_i \left(1 - \frac{d_i}{d_h}\right) \right\}. \tag{3.5}$$

Comparing Eqs. (3.4) and (3.5) gives that $R^* = \widetilde{R}^*$ iff $\widetilde{R}^* = \sum_{i=1}^{n} r_i$, *i.e.,* $\frac{\sum_{i=h}^{n} b_i}{d_h} \leq \sum_{i=1}^{h} r_i, \forall\ 1 \leq h \leq n$. In other words, a static priority scheduler will perform as well as the optimal one (yield the same minimum bandwidth), whenever flow deadlines are relative large and flow bursts small. However, when $\widetilde{R}^* \neq \sum_{i=1}^{n} r_i$, the use of a static priority scheduler can translate into a need for a much larger bandwidth.

Consider a scenario where R^* is achieved at h^*, *i.e.,* $R^* = \frac{\sum_{i=h^*}^{n} b_i}{d_{h^*}} + \sum_{i=h^*+1}^{n} r_i \left(1 - \frac{d_i}{d_{h^*}}\right)$. Though $\widetilde{R}^*$ may not be realized at the same h^* value, this still provides a lower bound for $\widetilde{R}^*$, namely, $\widetilde{R}^* \geq \frac{\sum_{i=h^*}^{n} b_i}{d_{h^*}} + \sum_{i=h^*+1}^{n} r_i$. Thus, the relative difference between $\widetilde{R}^*$ and R^* is no less than

$$\frac{\frac{\sum_{i=h^*}^{n} b_i}{d_{h^*}} + \sum_{i=h^*+1}^{n} r_i}{\frac{\sum_{i=h^*}^{n} b_i}{d_{h^*}} + \sum_{i=h^*+1}^{n} r_i \left(1 - \frac{d_i}{d_{h^*}}\right)} - 1 = \frac{\sum_{i=h^*+1}^{n} d_i r_i}{\sum_{i=h^*}^{n} b_i \ + \sum_{i=h^*+1}^{n} r_i \left(d_{h^*} - d_i\right)}. \tag{3.6}$$

As the right-hand-side of Eq. (3.6) increases with d_i for all $i \geq h^*$, it is maximized for $d_i = d_{h^*} - \epsilon_i, \forall i > h^*$, for arbitrarily small $\epsilon_{h^*+1} < \ldots < \epsilon_n$, so that its supremum is equal to $\frac{\sum_{i=h^*+1}^{n} r_i d_{h^*}}{\sum_{i=h^*}^{n} b_i}$. Note that this is intuitive, as when flows have arbitrarily close deadlines, they

should receive mostly equal service shares, which is in direct conflict with a strict priority ordering.

Note that under certain flow profiles, this supremum can be large. Take a two-flow scenario as an example. Basic algebraic manipulations give a supremum of $\frac{r_2}{r_1+r_2}$, which is achieved at $d_2 = d_1 = \frac{b_2+b_1}{r_1+r_2}$. Note that $\frac{r_2}{r_1+r_2} \to 1$ as $\frac{r_1}{r_2} \to 0$. Thus, in the two-flow case, the optimal static priority scheduler could have bandwidth requirements twice as large as those of the optimal dynamic priority scheduler.

Static Priorities with (re)Shaping

As shown by the comparison of Eqs. (3.4) and (3.5), static priorities can result in a minimum required bandwidth significantly larger than R^*. This is largely because static priorities are a rather blunt instrument when it comes to fine-tuning how to allocate transmission opportunities as a function of packet deadlines. In particular, they often result in some packets experiencing a delay much lower than their deadline.

This limitation is intrinsic to the static structure of the scheduler's decision, but it can be mitigated by anticipating the extent to which a flow may experience better deadlines than necessary and offset that advantage by modifying how the flow's packets are delivered to the scheduler. This can be realized by (re)shaping higher-priority (smaller deadline) flows before they enter the network, *i.e.,* through ingress reshaping. This introduces an additional access (ingress) reshaping delay that must be accounted for (deducted) in the target end-to-end deadline for the flow, but limits its impact on lower priority flows.

Consider the trivial example of a one-hop network (link) shared by two flows with profiles $(r_1, b_1, d_1) = (1, 5, 1.4)$ and $(r_2, b_2, d_2) = (4, 5, 1.25)$. In this case, the strict static-priority mechanism gives $\widetilde{R}^* = 11.14$. Assume next that we first (re)shape flow 2 to $(r_2, b_2') = (4, 0)$

before it enters the shared link. This reduces its delay budget at the shared link down to 0, but also entirely eliminates its burst. As a result, the resulting (fluid) system only needs a bandwidth of 7.57 while still meeting both flows' deadlines (a bandwidth of $4 = r_2$ is still consumed by flow 2, but the remaining 3.57 is sufficient to allow flow 1 to meet its deadline). In other words, (re)shaping flow 2 yields a bandwidth decrease of more than 30%. This simple example illustrates the potential benefits of access/ingress reshaping of flows. Next we proceed to characterize optimal (re)shaping parameters, and the resulting bandwidth gains.

When considering reshaping a flow with profile (r_i, b_i, d_i), the goal is to identify reshaping parameters (r'_i, b'_i) that maximize bandwidth savings (function of other, lower priority flows) without violating the flow's deadline d_i. In the base configuration involving only two flows, it can be shown (see Appendix B.6) that is sufficient to consider reshaping profiles of the form (r_i, b'_i), *i.e.,* limited to the flow's burst[46]. For that reason and to simplify our investigation, we limit ourselves to such profiles, *i.e.,* $r'_i = r_i$ and $0 \leq b'_i \leq b_i$. In the next few propositions, we first characterize flow delays when reshaped under static priorities, before deriving the optimal reshaping parameters (burst sizes) and the resulting minimum link bandwidth $\widetilde{R}^*_s$ that solves the corresponding version of **OPT** under a static priority scheduler and ingress reshaping.

Specifically, Proposition 14 characterizes the worst case delays (ingress reshaping delay plus link scheduling delay) of flows with given token-bucket traffic envelopes when assigned to a link of capacity R and served according to a priority scheduler. The result is then used to formulate an optimization problem, **OPT_S**, that seeks to minimize the link bandwidth R required to meet individual flow's deadlines, when flows are assigned to a priority class based

[46]We note that while introducing a *peak rate* constraint won't help in this base scenario, this needs not be the case in more general settings, *e.g.,* multi-hop networks, so that investigating the benefits of such an option remains of interest.

on their deadline (shorter deadlines have higher priority). The variables of the optimization are the flows ingress reshaping parameters. Proposition 16 then characterizes the minimum bandwidth $\widetilde{R}_s^*$ that **OPT_S** can achieve, while Proposition 17 provides explicit expressions for the optimal reshaping parameters.

Recall that priority n is the highest priority and let $\boldsymbol{b'} = (b'_1, b'_2, b'_3, ..., b'_n)$ be the vector of (re)shaped flow bursts, with $\boldsymbol{b'^*} = (b'^*_1, b'^*_2, b'^*_3, ..., b'^*_n)$ denoting the optimal configuration. Further, let $B'_i = \sum_{j=i}^{n} b'_j$ and $R_i = \sum_{j=i}^{n} r_j$, *i.e.,* the sum of the (re)shaped bursts and rates of flows with priority greater than or equal to $i, 1 \leq i \leq n$, with $B'_i = 0$ and $R_i = 0$ for $i > n$. Then flow i's worst-case end-to-end delay is captured by

Proposition 14. *Consider a one-hop network shared by n token-bucket controlled flows, where flow $i, 1 \leq i \leq n$, has a traffic contract of (r_i, b_i). Assume a static priority scheduler that assigns flow i a priority of i, where priority n is the highest priority, and (re)shapes flow i to (r_i, b'_i), where $0 \leq b'_i \leq b_i$. Given a shared link bandwidth of $R \geq \sum_{j=1}^{n} r_j$, the worst-case delay for flow i is*

$$D_i^* = max \left\{ \frac{b_i + B'_{i+1}}{R - R_{i+1}},\ \frac{b_i - b'_i}{r_i} + \frac{B'_{i+1}}{R - R_{i+1}} \right\}. \tag{3.7}$$

The proof is in Appendix B.3.2. Note that Eq. (3.7) captures the worst-case delay for flow i's last bit in a burst of b_i. Specifically, its first (second) term considers the case when the last bit arrives before (after) flow i's last busy period at the shared link, which affects the extent to which it is affected by the reshaping delay.

Observe also that D_i^* is independent of b'_1 for $2 \leq i \leq n$, and decreases with b'_1 when $i = 1$. This is intuitive as flow 1 has the lowest priority so that (re)shaping it cannot decrease the worst-case end-to-end delay of other flows. Consequently, reshaping it will also not reduce the minimum link bandwidth required to meet specific deadlines for each flow. Formally,

Corollary 15. *Consider a one-hop network shared by n token-bucket controlled flows, where flow $i, 1 \leq i \leq n$, has a traffic contract of (r_i, b_i) and a deadline of d_i, with $d_1 > d_2 > ... > d_n$ and $d_1 < \infty$. Assume a static priority scheduler that assigns flow i a priority of i, where priority n is the highest priority, and (re)shapes flow i to (r_i, b'_i), where $0 \leq b'_i \leq b_i$. Given a shared link bandwidth of $R \geq \sum_{j=1}^{n} r_j$, reshaping flow 1 cannot reduce the minimum required bandwidth.*

Combining Proposition 14 and Corollary 15 with **OPT** gives the following optimization **OPT_S** for a one-hop network shared by n flows and relying on a static priority scheduler with reshaping. Note that since the minimum link bandwidth needs to satisfy $R \geq \sum_{i=1}^{n} r_i$, combining this condition with R_i's definition gives $\sum_{i=1}^{n} r_i = R_1 \leq R$.

$$
\begin{aligned}
\textbf{OPT\_S} \quad & \min_{\boldsymbol{b'}} R \\
\text{s.t} \quad & \max\left\{\frac{b_i + B'_{i+1}}{R - R_{i+1}},\ \frac{b_i - b'_i}{r_i} + \frac{B'_{i+1}}{R - R_{i+1}}\right\} \leq d_i, \quad \forall\ 1 \leq i \leq n, \qquad (3.8) \\
& R_1 \leq R, \quad b'_1 = b_1, \quad 0 \leq b'_i \leq b_i, \qquad \forall\ 2 \leq i \leq n.
\end{aligned}
$$

The solution of **OPT_S** is characterized in Propositions 16 and 17. Proposition 16 characterizes the optimal bandwidth $\widetilde{R}^*_s$ based only on flow profiles, and while it is too complex to yield a closed-form expression, it offers a feasible numerical procedure to compute $\widetilde{R}^*_s$.

Proposition 16. *For $1 \leq i \leq n$, denote $H_i = b_i - d_i r_i$, $\Pi_i(R) = \frac{r_i + R - R_{i+1}}{R - R_{i+1}}$ and $V_i(R) = d_i(R - R_{i+1}) - b_i$. Define $\mathbb{S}_1(R) = \{V_1(R)\}$, and $\mathbb{S}_i(R) = \mathbb{S}_{i-1}(R) \bigcup \{V_i(R)\} \bigcup \left\{\frac{s - H_i}{\Pi_i(R)} \mid s \in \mathbb{S}_{i-1}(R)\right\}$ for $2 \leq i \leq n$. Then we have $\widetilde{R}^*_s = \max\{R_1, \inf\{R \mid \forall s \in \mathbb{S}_n(R), s \geq 0\}\}$.*

Computing $\widetilde{R}^*_s$ requires solving polynomial inequalities of degree $(n-1)$, so that a closed-form expression is not feasible except for small n. However, as $\mathbb{S}_i(R)$ relies only on flow profiles and $\mathbb{S}_j(R)$, $\forall j < i$, we can recursively construct $\mathbb{S}_n(R)$ from $\mathbb{S}_1(R)$. Hence, since

$R_1 \leq \widetilde{R}_s^* \leq \widetilde{R}^*$, we can use a binary search to compute $\widetilde{R}_s^*$ from the relation $\widetilde{R}_s^* = \max\{R_1, \inf\{R \mid \forall s \in \mathbb{S}_n(R), s \geq 0\}\}$ in Proposition 16.

Next, Proposition 17 gives a constructive procedure to obtain the optimal reshaping parameters $\boldsymbol{b'}^*$ given $\widetilde{R}_s^*$ and flow profiles.

Proposition 17. *Optimal reshaping parameters* $\boldsymbol{b'}^*$ *satisfy*

$$b_i'^* = \begin{cases} \max\{0, b_n - r_n d_n\}, & \text{when } i = n; \\ \max\left\{0,\ b_i - r_i d_i + \dfrac{r_i B_{i+1}'^*}{\widetilde{R}_s^* - R_{i+1}}\right\}, & \text{when } 2 \leq i \leq n-1. \end{cases} \tag{3.9}$$

where we recall that $b_1'^* = b_1$.

Note that the reshaping parameter $b_i'^*$ of flow $1 < i < n$ relies only on the optimal link bandwidth $\widetilde{R}_s^*$ and the reshaping parameters of higher priority flows. Hence, we can recursively characterize $b_i'^*$ from $b_n'^*$ given $\widetilde{R}_s^*$.

3.3.4 Basic FIFO with (re)Shaping

In this section, we consider the simplest possible scheduler, namely, a first-in-first-out (fifo) scheduler that serves data in the order in which it arrives. For conciseness and given the benefits of reshaping demonstrated in Section 3.3.3, we directly assume that flows can be reshaped prior to entering the network. Considering again a one-hop network and a set of n flows with profiles $(r_i, b_i, d_i), 1 \leq i \leq n$, our goal is to find reshaping parameters (r_i', b_i') to minimize the link bandwidth required to meet all the flows' deadlines. Again as in Section 3.3.3, we assume that $r_i = r_i'$ and focus on identifying the best b_i' values.

Towards answering this question, we first proceed to characterize the worst case delay across n flows sharing a link of bandwidth R equipped with a fifo scheduler, when the flows have initial traffic envelopes of the form $(r_i, b_i), 1 \leq i \leq n$, and have been individually reshaped to $(r_i, b'_i), 1 \leq i \leq n$, prior to accessing the shared link. Using this result, we then identify the reshaping parameters $b'_i, 1 \leq i \leq n$, that minimize the link bandwidth required to ensure that all flows meet deadlines of the form $d_1 > d_2 > \ldots > d_n$, and $d_1 < \infty$. As with other configurations, we only state the results with proofs relegated to Appendix B.4.

Proposition 18. *Consider a system with n rate-controlled flows with traffic envelopes (r_i, b_i), where $1 \leq i \leq n$, which share a fifo link with bandwidth $R \geq R_1 = \sum_{j=1}^{n} r_j$. Assume that the system reshapes flow i's traffic envelope to (r_i, b'_i). Then the worst-case delay for flow i is*

$$\widehat{D}_i^* = \max\left\{\frac{b_i - b'_i}{r_i} + \frac{\sum_{j\neq i} b'_j}{R}, \frac{\sum_{j=1}^{n} b'_j}{R} + \frac{(b_i - b'_i)R_1}{r_i R}\right\}. \tag{3.10}$$

The proof of Proposition 18 is in Appendix B.4.1

With the result of Proposition 18 in hand, we can formulate a corresponding optimization problem, **OPT_F**, for computing the optimal reshaping parameters that minimize the link bandwidth required to meet the deadlines $d_1 > d_2 > \ldots > d_n$, and $d_1 < \infty$ of a set of n flows. Specifically, combining Proposition 18 with **OPT** gives the following optimization **OPT_F** for a one-hop network shared by n flows and relying on a fifo scheduler with reshaping. As before, $\sum_{i=1}^{n} r_i = R_1 \leq R$.

$$\begin{aligned}
\textbf{OPT_F} \quad & \min_{\boldsymbol{b'}} R \\
\text{s.t} \quad & \max\left\{\frac{b_i - b'_i}{r_i} + \frac{\sum_{j\neq i} b'_j}{R}, \frac{\sum_{j=1}^{n} b'_j}{R} + \frac{(b_i - b'_i)R_1}{r_i R}\right\} \leq d_i, \quad \forall\ 1 \leq i \leq n, \\
& R_1 \leq R, \quad 0 \leq b'_i \leq b_i, \qquad\qquad \forall\ 1 \leq i \leq n.
\end{aligned} \tag{3.11}$$

The solution of **OPT_F** is characterized in Propositions 19 and 20. As in the case of a static priority scheduler, Proposition 19 describes a numerical procedure to compute the optimal bandwidth $\widehat{R}_s^*$ given the flow profiles, while Proposition 20 specifies the optimal reshaping parameters $\widehat{\boldsymbol{b}}'^*$ given $\widehat{R}_s^*$ and the flow profiles.

Proposition 19. *For* $1 \le i \le n$, *define* $H_i = b_i - d_i r_i$, $\widehat{B}_i = \sum_{j=1}^{i} b_j$, *and* $\mathbb{Z}_i = \{1 \le j \le i \mid j \in \mathbb{Z}\}$. *Denote*

$$X_F(R) = \max_{P_1, P_2 \subseteq \mathbb{Z}_n, P_2 \neq \mathbb{Z}_n, P_1 \bigcap P_2 = \emptyset} \frac{\sum_{i \in P_1} \frac{RH_i}{R+r_i} + \sum_{i \in P_2} \left(b_i - \frac{r_i d_i R}{R_1} \right)}{1 - \sum_{i \in P_1} \frac{r_i}{R+r_i} - \sum_{i \in P_2} \frac{r_i}{R_1}}$$

and

$$Y_F(R) = \min_{1 \le i \le n-1} \left\{ \widehat{B}_n, Rd_n, \min_{P_1, P_2 \subseteq \mathbb{Z}_i, P_1 \bigcap P_2 = \emptyset, P_1 \bigcup P_2 \neq \emptyset} \left\{ \frac{\widehat{B}_i - \sum_{j \in P_1} \frac{RH_j}{R+r_j} - \sum_{j \in P_2} \left(b_j - \frac{r_j d_j R}{R_1} \right)}{\sum_{j \in P_1} \frac{r_j}{R+r_j} + \sum_{j \in P_2} \frac{r_j}{R_1}} \right\} \right\}.$$

Then the optimal solution for ***OPT_F*** *is*

$$\widehat{R}_s^* = \max \left\{ R_1, \frac{\widehat{B}_n R_1}{\sum_{i=1}^{n} r_i d_i}, \min\{R \mid X_F(R) \le Y_F(R)\} \right\}.$$

Since $\max\left\{R_1, \frac{\widehat{B}_n R_1}{\sum_{i=1}^{n} r_i d_i}\right\} \le \widehat{R}_s^* \le \widehat{R}^* = \max\left\{R_1, \frac{\widehat{B}_n}{d_n}\right\}$, where $\widehat{R}^*$ is the minimum required bandwidth achieved by a pure fifo system, we can use a binary search to compute $\widehat{R}_s^*$ based on Proposition 19. Once $\widehat{R}_s^*$ is known, the optimal reshaping parameters can be obtained as stated in Proposition 20

Proposition 20. *For* $1 \le i \le n$, *define* $T_i(\widehat{B}'_n, R) = \max\left\{0, \frac{R}{R+r_i}\left(H_i + \frac{r_i}{R}\widehat{B}'_n\right), b_i + \frac{r_i(\widehat{B}'_n - Rd_i)}{R_1}\right\}$. ***OPT_F****'s optimal reshaping parameters* $\widehat{\boldsymbol{b}}'^*$ *satisfy* $\widehat{b}'^*_1 = \widehat{B}'^*_1$, *and* $\widehat{b}'^*_i = \widehat{B}'^*_i - \widehat{B}'^*_{i-1}$ *for*

$2 \leq i \leq n$, *where* $\widehat{\boldsymbol{B}}'^{*}$ *satisfy*

$$\begin{cases} \widehat{B}'^{*}_{n} = X_F(\widehat{R}^{*}_{s}), \\ \widehat{B}'_{i} = \max\left\{ \sum_{j=1}^{i} T_j(\widehat{B}'^{*}_{n}, \widehat{R}^{*}_{s}), \widehat{B}'^{*}_{i+1} - b_{i+1} \right\}, \ \textit{when } 1 \leq i \leq n-1. \end{cases} \tag{3.12}$$

Note that $\widehat{B}'^{*}_{n}$ relies only on $\widehat{R}^{*}_{s}$ and flow profiles. Whereas when $1 \leq i \leq n-1$, $\widehat{B}'^{*}_{i}$ relies only on $\widehat{R}^{*}_{s}$, $\widehat{B}'^{*}_{n}$, $\widehat{B}'^{*}_{i+1}$ and flow profiles. Hence, we can recursively characterize $\widehat{B}'^{*}_{i}$ from $\widehat{B}'^{*}_{n}$ given $\widehat{R}^{*}_{s}$.

3.3.5 Evaluation

In this section, we explore the relative benefits of the solutions presented in the previous three sections. Of interest is assessing the "cost of simplicity," namely the amount of additional bandwidth required when relying on simpler schedulers such as static priority or fifo compared to an edf-based dynamic priority scheduler. Also of interest is the magnitude of the improvements that (ingress) reshaping of flows can afford with static priority and fifo schedulers. To that end, the evaluation proceeds with a number of pairwise comparisons to quantify the relative cost (in bandwidth) of each alternative.

The evaluation initially focuses (Section 3.3.5) on scenarios involving only two flows. In this base setting, explicit expressions are available for the minimum bandwidth under each configuration, so that formal comparisons are possible. This is then extended (Section 3.3.5) to more "general" scenarios involving multiple flows with different combinations of deadlines and traffic envelopes.

Basic Two-Flow Configurations

Recalling our earlier notation for the minimum bandwidth required in each configuration, *i.e.,* R^* (dynamic priority); $\widetilde{R}^*$ (static priority); $\widetilde{R}^*_s$ (static priority w/ reshaping); $\widehat{R}^*$ (fifo); and $\widehat{R}^*_s$ (fifo w/ reshaping), and specializing Eq. (3.3) to a configuration with only two flows, (r_1, b_1, d_1) and (r_2, b_2, d_2), the minimum bandwidth to meet the flows' deadlines is given by

$$R^* = \max\left\{r_1 + r_2,\ \frac{b_2}{d_2},\ \frac{b_1 + b_2 - r_2 d_2}{d_1} + r_2\right\}, \tag{3.13}$$

which is, therefore, also the bandwidth required by the dynamic priority scheduler.

Conversely, if we consider a static priority scheduler, from Eq. (3.4), its bandwidth requirement $\widetilde{R}^*$ (in the absence of any reshaping) for the same two-flow configuration is of the form

$$\widetilde{R}^* = \max\left\{r_1 + r_2,\ \frac{b_2}{d_2},\ \frac{b_1 + b_2}{d_1} + r_2\right\}; \tag{3.14}$$

If (optimal) reshaping is introduced, specializing Proposition 16 to two flows, the minimum bandwidth $\widetilde{R}^*_s$ reduces to

$$\widetilde{R}^*_s = \begin{cases} \max\left\{r_1 + r_2,\ \frac{b_2}{d_2},\ \frac{b_1 + b_2 - r_2 d_2}{d_1} + r_2\right\}, & \text{when } \frac{b_2}{r_2} \geq \frac{b_1}{r_1} \\ \max\left\{r_1 + r_2,\ \frac{b_2}{d_2},\ \frac{b_1 + \max\{b_2 - r_2 d_2, 0\}}{d_1} + r_2\right\}, & \text{otherwise;} \end{cases} \tag{3.15}$$

Finally, similarly specializing the results of Propositions 19 and 20 to two flows, we find that the minimum required bandwidth $\widehat{R}^*$ under fifo without reshaping is

$$\widehat{R}^* = \max\left\{r_1 + r_2,\ \frac{b_1 + b_2}{d_2}\right\}; \tag{3.16}$$

and that when (optimal) reshaping is used, $\widehat{R}_s^*$ is given by

$$\widehat{R}_s^* = \max\left\{ r_1 + r_2, \frac{b_2}{d_2}, \frac{(b_1+b_2)(r_2+r_2)}{d_1 r_1 + d_2 r_2}, \frac{b_1 + b_2 - d_1 r_1 + \sqrt{(b_1+b_2-d_1 r_1)^2 + 4 r_1 d_2 b_2}}{2 d_2} \right\}. \tag{3.17}$$

With these expressions in hand, we can now proceed to assess the relative benefits of each option in the basic two-flow scenario.

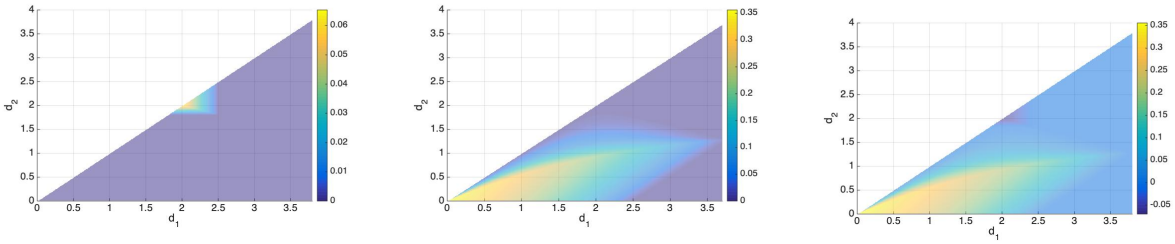
The Impact of Scheduler Complexity We first evaluate the impact of relying on schedulers of decreasing complexity, when those schedulers are coupled with an optimal reshaping solution. In other words, we compare the bandwidth requirements of a dynamic priority scheduler to those of static priority and fifo schedulers combined with an optimal reshaper. The comparison is in the form of relative differences (improvements realizable from more complex schedulers), *i.e.,* $\frac{\widetilde{R}_s^* - R^*}{\widetilde{R}_s^*}$, $\frac{\widehat{R}_s^* - R^*}{\widehat{R}_s^*}$, and $\frac{\widehat{R}_s^* - \widetilde{R}_s^*}{\widehat{R}^*}$.

Dynamic priority vs. static priority w/ optimal reshaping.

Starting with comparing a dynamic priority scheduler with a static priority one with optimal reshaping, we know from Eqs. (3.13) and (3.15) that $R^* < \widetilde{R}_s^*$ iff $\frac{b_2}{r_2} < d_2 \leq d_1 < \frac{b_1}{r_1}$.

To illustrate this difference, Fig. 3.2a uses a "heatmap" for a specific yet representative two-flow combination, $(r_1, b_1) = (4, 10)$ and $(r_2, b_2) = (10, 18)$, while varying their respective deadlines. As shown in the figure, the static priority scheduler, when combined with reshaping, performs as well as a dynamic priority scheduler, except for a relatively small (triangular) region where d_1 and d_2 are close to each other and both of intermediate values[47]. Towards better characterizing this range, *i.e.,* $d_2 > \frac{b_2}{r_2}$ and $d_1 < \frac{b_1}{r_1}$, we see that the

[47]When d_2 and d_1 are close but small, an edf dynamic priority scheduler behaves like a static priority one as the very large bandwidth called for by small deadlines ensures that data from either class is transmitted before dynamic priorities can affect transmissions order. In other words, if a burst of low-priority (larger



(a) Dyn. prio. vs. stat. prio. + reshaping

(b) Dyn. prio. vs. fifo + reshaping

(c) fifo + reshaping vs. stat. prio. + reshaping

Figure 3.2: Relative bandwidth increases for $(r_1, b_1) = (4, 10)$ and $(r_2, b_2) = (10, 18)$, as a function of d_1 and $d_2 < d_1$. The figure is in the form of a "heatmap." Darker colors (purple) correspond to smaller increases than lighter ones (yellow).

supremum of $\frac{\widetilde{R}_s^* - R^*}{\widetilde{R}_s^*}$ is achieved at $d_1 = d_2 = \frac{b_1+b_2}{r_1+r_2}$, with $\widetilde{R}_s^* = \frac{b_1}{d_1} + r_2$, and $R^* = r_1 + r_2$. The relative difference in bandwidth between the two schemes is then of the form

$$\frac{\widetilde{R}_s^* - R^*}{\widetilde{R}_s^*} = 1 - \frac{1}{\frac{b_1}{b_1+b_2} + \frac{r_2}{r_1+r_2}},$$

which can be shown to be upper-bounded by 0.5. In other words, in the two-flow case, the (optimal) dynamic scheduler can result in a bandwidth saving of at most 50% when compared to a static priority scheduler with (optimal) reshaping, and this happens when the deadlines of the two flows are very close to each other. This is unlikely in practice, and we expand on this in Section 3.3.5.

Dynamic priority vs. fifo w/ optimal reshaping

deadline) arrives $(d_1 - d_2)$ prior to a high-priority (smaller deadline) burst so that its remainder would have "higher" priority under edf, the speed of the link ensures there is no remainder, and therefore no difference between edf and a static priority scheduler. Similarly, when both the high and the low priority flows generate simultaneous bursts, the high speed of the link ensures that any additional data contributed by the high-priority flow while its burst is being transmitted under static priority is cleared before the low-priority burst would have become eligible for transmission under an edf scheduler. Conversely, when d_2 and d_1 are close but large, both schedulers meet the deadlines with a bandwidth equal to the sum of the flows' average rates.

Next, we study the performance difference between the dynamic priority scheduler and a fifo scheduler that, as with the static priority scheduler, has been combined with optimal reshaping. Comparing $\widehat{R}_s^*$ with R^* from Eqs. (3.17) and (3.13) gives that $\widehat{R}_s^* > R^*$ iff $d_1 - \frac{b_1}{r_1} < d_2 < \frac{b_1+b_2-d_1r_1}{r_2}$. We illustrate the corresponding relative difference in Fig. 3.2b using again the same two-flow combination as before. From the figure, we see that fifo + shaping performs poorly relative to a dynamic priority scheduler when neither d_1 nor d_2 are large. As with static priorities, such configurations may not be common in practice, and we explore this further below and in Section 3.3.5.

To explore the source and possible magnitude of this difference, we note that the supremum of $\frac{\widehat{R}_s^*-R^*}{\widehat{R}_s^*}$ is achieved when $0 < d_2 < \frac{b_1+b_2+r_2d_1-\sqrt{(b_1+b_2+r_2d_1)^2-4r_2b_2d_1}}{2r_2}$, with Eq. (3.13) defaulting to $R^* = \frac{b_2}{d_2}$ and Eq. (3.17) to $\widehat{R}_s^* = \frac{b_1+b_2-d_1r_1+\sqrt{(b_1+b_2-d_1r_1)^2+4r_1d_2b_2}}{2d_2}$. Hence, the relative difference becomes

$$\frac{\widehat{R}_s^* - R^*}{\widehat{R}_s^*} = 1 - \frac{2b_2}{b_1 + b_2 - d_1r_1 + \sqrt{(b_1 + b_2 - d_1r_1)^2 + 4r_1d_2b_2}},$$

which increases with d_2. Thus, its supremum is achieved at $d_2 \to d_1$. From basic algebraic manipulation, $1 - \frac{2b_2}{b_1+b_2-d_1r_1+\sqrt{(b_1+b_2-d_1r_1)^2+4r_1d_2b_2}}$ also decreases with d_1. Hence, the supremum of the relative difference is achieved at $d_1 \to 0$, and is of the form $\frac{b_1}{b_1+b_2}$, which goes to 1 as $\frac{b_1}{b_2} \to \infty$. In other words a dynamic priority scheduler can realize up to a 100% improvement over a basic fifo scheduler that reshapes flows optimally.

Fifo vs. static priority both w/ optimal reshaping

Finally, we compare fifo and static priority schedulers when both rely on optimal reshaping. Comparing Eqs. (3.17) and (3.15) gives that $\widehat{R}_s^* > \widetilde{R}_s^*$ iff $\max\left\{\frac{b_2}{r_2}, \frac{(b_1+b_2)(r_1+r_2)}{r_2(b_1/d_1+r_2)}\right\} < d_1 <$

$\frac{b_1}{r_1}$. Fig. 3.2c illustrates their difference, again relying on a heatmap for the same two-flow combination used for the two previous scenarios.

The figure shows that the benefits of priority are maximum when d_2 is small and d_1 is not too large. This is intuitive in that a small d_2 calls for affording maximum protection to flow 2, which a priority structure offers more readily than a fifo. Conversely, when d_1 is large, flow 1 can be reshaped to eliminate all burstiness, which limits its impact on flow 2 even when both flows compete in a fifo scheduler.

The figure also reveals that a small region exists (when d_1 and d_2 are close to each other and both are of intermediate value) where fifo outperforms static priority. As alluded to in the discussion following Proposition 13 and as expanded in Appendix B.5, this is because a *strict* priority ordering of flows as a function of their deadlines needs not always be optimal. For instance, it is easy to see that two otherwise identical flows that only differ infinitesimally in their deadlines should be treated "identically." This is more readily accomplished by having them share a common fifo queue than having them assigned to two distinct priorities.

To better understand differences in performance between the two schemes, we characterize the supremum and the infimum of $\frac{\widehat{R}_s^* - \widetilde{R}_s^*}{\widehat{R}_s^*}$.

Basic algebraic manipulations show that the supremum is achieved at $d_1 = d_2 \to 0$, where Eq. (3.17) defaults to $\widehat{R}_s^* = \frac{b_1+b_2-d_1r_1+\sqrt{(b_1+b_2-d_1r_1)^2+4r_1d_2b_2}}{2d_2}$ and Eq. (3.15) to $\widetilde{R}_s^* = \frac{b_2}{d_2}$, so that their relative difference is ultimately of the form

$$\frac{\widehat{R}_s^* - \widetilde{R}_s^*}{\widehat{R}_s^*} = \frac{b_1}{b_1 + b_2},$$

which goes to 1 when $\frac{b_1}{b_2} \to \infty$, *i.e.,* a maximum penalty of 100% for fifo with reshaping over static priorities with reshaping.

Conversely, the infimum is achieved at $d_1 = d_2 = \frac{b_1+b_2}{r_1+r_2}$, with Eqs. (3.15) and (3.17) then defaulting to $\widetilde{R}_s^* = \frac{b_1}{d_1} + r_2$ and $\widehat{R}_s^* = r_1 + r_2$, respectively. Their relative difference is then of the form

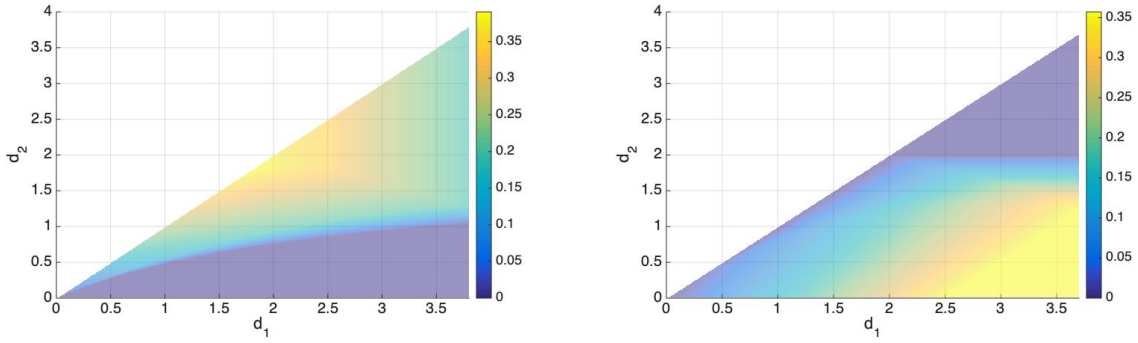
$$\frac{\widehat{R}_s^* - \widetilde{R}_s^*}{\widehat{R}_s^*} = \frac{r_1}{r_1 + r_2} - \frac{b_1}{b_1 + b_2},$$

which increases with $\frac{r_1}{r_2}$ and decreases with $\frac{b_1}{b_2}$. When $\frac{r_1}{r_2} \to 0$ and $\frac{b_1}{b_2} \to \infty$, it achieves an infimum of -1, *i.e.,* a maximum penalty of 100% but now for static priorities with reshaping over fifo with reshaping. In other words, when used with reshaping, both fifo and static priority can end-up requiring twice as much bandwidth as the other. Addressing this issue calls for determining when flows should be grouped in the same priority class rather than assigned to separate classes, and while the optimal grouping can be identified in simple scenarios with two or three flows, *e.g.,* see Appendix B.5, a general solution remains elusive. However, as we shall see in Section 3.3.5, the simple strict priority assignment on which we rely appears to perform reasonably well across a broad range of flow configurations.

The Benefits of Reshaping In this section, we evaluate the benefits afforded by (optimally) reshaping flows when using static priority and fifo schedulers. This is done by computing the minimum bandwidth required to meet flow deadlines under both schedulers without and with reshaping, and evaluating the resulting relative differences, *i.e.,* $\frac{\widetilde{R}^* - \widetilde{R}_s^*}{\widetilde{R}^*}$ and $\frac{\widehat{R}^* - \widehat{R}_s^*}{\widehat{R}^*}$.

Starting with a static priority scheduler, Eqs. (3.14) and (3.15) indicate that $\widetilde{R}_s^* < \widetilde{R}^*$ iff $\widetilde{R}^* = \frac{b_1+b_2}{d_1} + r_2 > \max\left\{r_1 + r_2, \frac{b_2}{d_2}\right\}$, *i.e.,* for a static priority scheduler, (re)shaping[48] decreases the required bandwidth only when d_1, the larger deadline, is not too large and d_2, the smaller deadline, is not too small. This is intuitive. When d_1 is large, the low-priority flow 1 can meet its deadline even without any mitigation of the impact of flow 2.

[48]Recall from Corollary 15 that the flow with the largest deadline, flow 1 in the two-flow case, is never reshaped.



(a) Stat. prio. without vs. with shaping

(b) fifo without vs. with shaping

Figure 3.3: Relative bandwidth increases for $(r_1, b_1) = (4, 10)$ and $(r_2, b_2) = (10, 18)$ as a function of d_1 and $d_2 < d_1$. The figure is in the form of a "heatmap." Darker colors (purple) correspond to smaller increases than lighter ones (yellow).

Conversely, a small d_2 offers little to no opportunity for reshaping flow 2 as the added delay it would introduce would need to be compensated by an even higher link bandwidth. This is illustrated in Fig. 3.3a for the same two-flow combination as in Fig. 3.2, *i.e.,* $(r_1, b_1) = (4, 10)$ and $(r_2, b_2) = (10, 18)$. The intermediate region where "d_1 is not too large and d_2 is not too small" corresponds to the yellow triangular region where the benefits of reshaping can reach 40%.

Similarly, Eqs. (3.16) and (3.17) indicate that $\widehat{R}_s^* < \widehat{R}^*$ iff $d_2 < \frac{b_1+b_2}{r_1+r_2}$, *i.e.,* for a fifo scheduler, (re)shaping decreases the required bandwidth only when d_2, the smaller deadline, is small. This is again intuitive as a large d_2 means that the small deadline flow 2 can meet its deadline even without any reshaping of flow 1[49]. Fig. 3.3b presents the relative gain in link bandwidth for again the same 2-flow combination. As seen in the figure, the benefits of reshaping can, as in the static priority case, again reach reach close to 40% for a fifo scheduler, at least in the example under consideration. The next section explores more complex scenarios involving more than two flows and different combinations of flow profiles, and from those results it

[49]Note that unlike the static priority scenario where the smaller deadline (higher priority) flow is reshaped, the opposite holds in the fifo case, where the larger deadline flow is reshaped to minimize its impact on the one with the tighter deadline.

appears that, in general, a fifo scheduler stands to benefit more from reshaping than a static priority one.

Relative Performance – Multiple Flows In this section, we extend the investigation of Section 3.3.5 to more general configurations. Specifically, we consider scenarios with more than two flows, where flows are assigned to *ten* different deadline classes. We set the dynamic range of the deadline classes to 10, *i.e.,* with minimum and maximum deadlines of 0.1 and 1, respectively, and consider different assignments in that range for the 10 deadlines across classes. Specifically, we select three different possible patterns for assigning the deadline of each class, namely,

- *Even* deadline assignment:
  1. $\boldsymbol{d}_{11} = (1, 0.9, 0.8, 0.7, 0.6, 0.5, 0.4, 0.3, 0.2, 0.1)$;
- *Bi-modal* deadline assignment:
  2. $\boldsymbol{d}_{21} = (1, 0.95, 0.9, 0.85, 0.8, \quad 0.3, 0.25, 0.2, 0.15, 0.1)$,
  3. $\boldsymbol{d}_{22} = (1, 0.96, 0.93, 0.9, 0.86, 0.83, 0.8, \quad 0.2, 0.15, 0.1)$,
  4. $\boldsymbol{d}_{23} = (1, 0.95, 0.9, \quad 0.3, 0.26, 0.23, 0.2, 0.16, 0.13, 0.1)$;
- *Tri-modal* deadline assignment:
  5. $\boldsymbol{d}_{31} = (1, 0.95, 0.9, \quad 0.6, 0.55, 0.5, 0.45, \quad 0.2, 0.15, 0.1)$,
  6. $\boldsymbol{d}_{32} = (1, \quad 0.68, 0.65, 0.62, 0.6, 0.57, 0.55, 0.53, 0.5, \quad 0.1)$,
  7. $\boldsymbol{d}_{33} = (1, \quad 0.6, \quad 0.28, 0.25, 0.23, 0.2, 0.17, 0.15, 0.12, 0.1)$,
  8. $\boldsymbol{d}_{34} = (1, 0.97, 0.95, 0.93, 0.9, 0.88, 0.85, 0.82, \quad 0.6, \quad 0.1)$.

Those three assignments translate into different patterns in how close to each others deadlines are, as well as how they are spread across the full dynamic range, *e.g.,* with some scenarios grouping a number of deadlines within a sub-range.

Each of the above eight deadline assignments is then used to perform a total of $1,000$ experiments, where an experiment consists of randomly selecting a "flow's" traffic envelope for each of the ten deadlines. Note that what we denote by a flow in practice maps to the aggregate of all individual flows assigned to the corresponding deadline class (individual flow envelopes simply add up). Specifically, ten (aggregate) flow burst sizes b_1 to b_{10} are drawn independently from $U(1, 10)$, and ten (aggregate) rates r_1 to r_{10} are drawn independently from $U\left(0, \sum_{i=1}^{10} b_i\right)$. The upper bound $\sum_{i=1}^{10} b_i$ of the rate range corresponds to a rate value beyond which even a fifo scheduler without reshaping always performs as well as the optimal solution[50], so that there are then no differences across mechanisms.

The results of the experiments are summarized in Table 3.1, which gives the mean, standard deviation, and the mean's 95% confidence interval for the relative savings in required link bandwidth, first from using dynamic priority over static priority + shaping, followed by fifo + shaping, and then between static priority + shaping and fifo + shaping.

The first conclusion from the data in Table 3.1 is that while a dynamic priority scheduler affords some benefits, they are on average smaller than the maximum values that Section 3.3.5 indicated as possible. In particular, average improvements over static priority with reshaping were often around 1% and did not exceed slightly more than 6% across all configurations. Those values were a little higher when considering fifo with reshaping, where they reached 12%, but those again remain significantly less than the worst case scenarios of Section 3.3.5.

Table 3.1 also reveals that, somewhat surprisingly, static priority and fifo perform similarly when both are afforded the benefit of reshaping. Static priority holds a slight edge on average, but as discussed in Section 3.3.5, this is not consistent across configurations, and a few scenarios exist where a fifo scheduler outperforms static priority when combined with

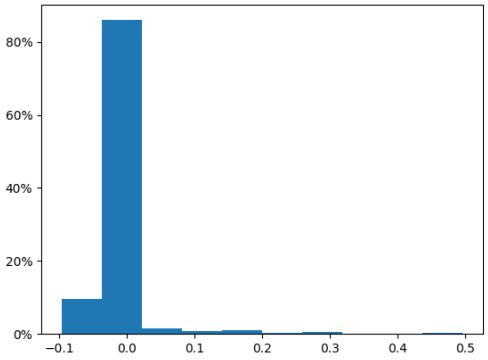
[50]This happens when the sum of the rates is large enough to alone clear the aggregate burst before the smallest deadline.

Table 3.1: Relative bandwidth savings.

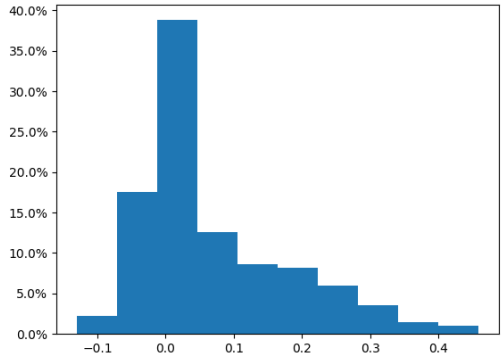
Comparisons	**Scenario**	**Mean**	**Std. Dev.**	95% **Conf. Intvl.**
R^* vs. $\widetilde{R}_s^*$ $\left(\frac{\widetilde{R}_s^* - R^*}{\widetilde{R}_s^*}\right)$	$\boldsymbol{d}_{11}$	0.012	0.023	[0.0102, 0.0131]
	$\boldsymbol{d}_{21}$	0.015	0.027	[0.0135, 0.0169]
	$\boldsymbol{d}_{22}$	0.011	0.022	[0.0101, 0.0128]
	$\boldsymbol{d}_{23}$	0.029	0.042	[0.0259, 0.0312]
	$\boldsymbol{d}_{31}$	0.014	0.025	[0.012, 0.0151]
	$\boldsymbol{d}_{32}$	0.010	0.021	[0.0084, 0.011]
	$\boldsymbol{d}_{33}$	0.062	0.065	[0.0576, 0.0658]
	$\boldsymbol{d}_{34}$	0.007	0.017	[0.006, 0.0081]
R^* vs. $\widehat{R}_s^*$ $\left(\frac{\widehat{R}_s^* - R^*}{\widehat{R}_s^*}\right)$	$\boldsymbol{d}_{11}$	0.017	0.065	[0.013, 0.0211]
	$\boldsymbol{d}_{21}$	0.032	0.087	[0.0268, 0.0376]
	$\boldsymbol{d}_{22}$	0.017	0.062	[0.0126, 0.0203]
	$\boldsymbol{d}_{23}$	0.080	0.128	[0.0724, 0.0882]
	$\boldsymbol{d}_{31}$	0.025	0.078	[0.0206, 0.0303]
	$\boldsymbol{d}_{32}$	0.008	0.046	[0.0054, 0.0111]
	$\boldsymbol{d}_{33}$	0.120	0.141	[0.1115, 0.129]
	$\boldsymbol{d}_{34}$	0.004	0.032	[0.002, 0.006]
$\widetilde{R}_s^*$ vs. $\widehat{R}_s^*$ $\left(\frac{\widehat{R}_s^* - \widetilde{R}_s^*}{\widehat{R}_s^*}\right)$	$\boldsymbol{d}_{11}$	0.006	0.065	[0.0016, 0.0095]
	$\boldsymbol{d}_{21}$	0.018	0.083	[0.0126, 0.0228]
	$\boldsymbol{d}_{22}$	0.005	0.061	[0.0012, 0.0088]
	$\boldsymbol{d}_{23}$	0.055	0.113	[0.0484, 0.0624]
	$\boldsymbol{d}_{31}$	0.012	0.075	[0.0076, 0.0169]
	$\boldsymbol{d}_{32}$	-0.002	0.045	[-0.0043, 0.0013]
	$\boldsymbol{d}_{33}$	0.066	0.112	[0.0592, 0.073]
	$\boldsymbol{d}_{34}$	-0.003	0.033	[-0.0053, -0.0012]

reshaping of flows. Fig. 3.4 illustrates when and why this might be the case by plotting the relative "penalty" of fifo + reshaping over static priority + reshaping for two of the tri-modal deadline assignments, d_{32} and d_{33}.

The differences between the two selected deadline assignments are in the relative magnitudes of their three modes. Assignment d_{32} boasts a relatively large middle mode with six intermediate deadlines, and two extreme modes (small and large deadlines), each with only one deadline. In contrast, assignment d_{33} has two small upper modes (large and intermediate),



(a) Tri-modal deadline assignment $\boldsymbol{d}_{32}$



(b) Tri-modal deadline assignment $\boldsymbol{d}_{33}$

Figure 3.4: Relative bandwidth difference between fifo + shaping and static priority + shaping.

each with a single deadline, and a large lower mode consisting of six relatively low value deadlines.

Recall that the introduction of priorities enforces strict differentiation between flows as a function of the their deadline (smaller deadlines have higher priority). As a result, even if reshaping mitigates the impact of priority, a relatively large number of closely grouped deadlines is a poor fit for a priority scheme, especially when the number of other flows for which it can be beneficial is small. This is the scenario of deadline assignment d_{32} (there is only one small deadline flow that benefits from being assigned to the highest priority, and conversely only one large deadline flow from which other flows are protected by assigning it to the lowest priority), which explains the relatively poor performance of static priority over fifo. In contrast, deadline assignment d_{33} boasts a large number of small deadlines that all stand to benefit from being shielded of the impact of the two larger deadline flows, even if the introduction of strict differentiation among those six small deadline flows needs not be very useful (reshaping again mitigates its negative impact). Nevertheless, this offers some insight into the better performance of static priority over fifo in this particular scenario.

Table 3.2: Relative benefits of reshaping flows.

Comparisons	Scenario	Mean	Std. Dev.	95% Conf. Intvl.
$\widetilde{R}_s^*$ vs. $\widetilde{R}^*$ $\left(\frac{\widetilde{R}^*-\widetilde{R}_s^*}{\widetilde{R}^*}\right)$	$\boldsymbol{d}_1$	0.0843	0.0450	[0.0815, 0.0871]
	$\boldsymbol{d}_{21}$	0.0811	0.0419	[0.0785, 0.0837]
	$\boldsymbol{d}_{22}$	0.0842	0.0452	[0.0814, 0.0871]
	$\boldsymbol{d}_{23}$	0.0938	0.0480	[0.0908, 0.0967]
	$\boldsymbol{d}_{31}$	0.0824	0.0433	[0.0797, 0.0851]
	$\boldsymbol{d}_{32}$	0.0949	0.0507	[0.0918, 0.0981]
	$\boldsymbol{d}_{33}$	0.1597	0.0478	[0.1567, 0.1627]
	$\boldsymbol{d}_{34}$	0.0883	0.0494	[0.0853, 0.0914]
$\widehat{R}_s^*$ vs. $\widehat{R}^*$ $\left(\frac{\widehat{R}^*-\widehat{R}_s^*}{\widehat{R}^*}\right)$	$\boldsymbol{d}_1$	0.4952	0.0817	[0.4901, 0.5003]
	$\boldsymbol{d}_{21}$	0.4871	0.0762	[0.4824, 0.4918]
	$\boldsymbol{d}_{22}$	0.4953	0.0827	[0.4902, 0.5005]
	$\boldsymbol{d}_{23}$	0.4578	0.0652	[0.4537, 0.4618]
	$\boldsymbol{d}_{31}$	0.4908	0.0788	[0.4859, 0.4957]
	$\boldsymbol{d}_{32}$	0.4995	0.0859	[0.4942, 0.5049]
	$\boldsymbol{d}_{33}$	0.4247	0.0619	[0.4208, 0.4285]
	$\boldsymbol{d}_{34}$	0.5013	0.0884	[0.4959, 0.5068]

Towards gaining a better understanding of reshaping and the extent to which it may be behind the somewhat unexpected good performance of fifo, Table 3.2 reports its impact for both static priority and fifo. Specifically, as Table 3.1, it gives the mean, standard deviation, and the mean's 95% confidence interval of the relative gains in bandwidth that reshaping affords for both schedulers.

The data from Table 3.2 highlights that while both static priority and fifo benefit from reshaping, the magnitude of the improvements is significantly higher for fifo. Specifically, improvements from reshaping are systematically above 40% and often close to 50% for fifo, while they exceed 10% only once for static priority (at 15% for scenario d_{33}) and are typically around 8%. As alluded to earlier, this is not surprising given that static priority offers at least some, albeit blunt, ability to discriminate flows based on their deadlines, while fifo lacks any such ability. This difference is illustrated more explicitly in Fig. 3.5 through the full cumulative distribution function (cdf) of those benefits for both static priority and fifo

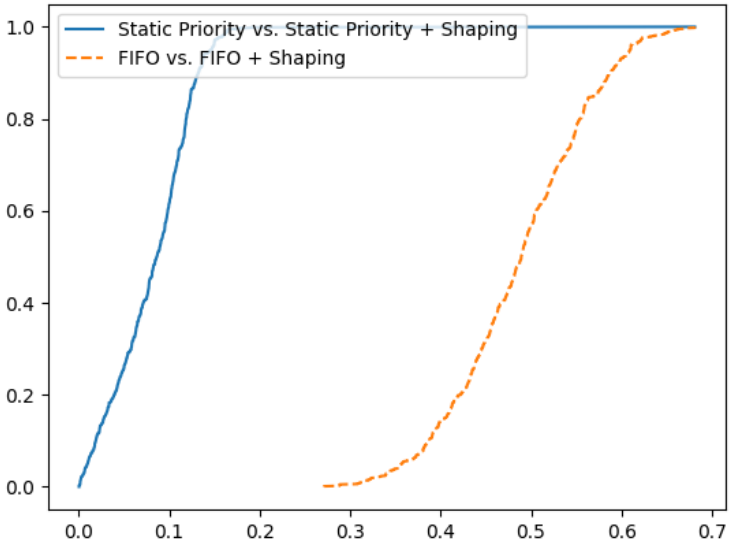


Figure 3.5: CDF of relative bandwidth reduction from reshaping under bi-modal deadline assignment $\boldsymbol{d}_{21}$.

under the deadline assignment $\boldsymbol{d}_{21}$ (bi-modal, with two similar modes of five deadlines at the two ends of the deadline range).

3.4 Multi-node Case

In this section, we proceed to extend our single-node results to multi-node networks. We present a first step in investigating this problem, and restrict ourselves to systems where the directed graph derived from flow routes can form a DAG, *i.e.,* all nodes in the network can be topologically ordered such that the ordering is consistent with all the flow routes.

There are mainly two challenges to build multi-hop procedures, to map the end-to-end deadline to per-node targets, and to account for decisions dependencies across nodes. Though local deadline assignment is an interesting yet complicated topic by itself, however, exploring different deadline splitting mechanisms is not the focus of this work. Therefore, we propose

to adopt the simple even deadline splitting mechanism, *i.e.,* split end-to-end deadline equally across nodes. For dependencies, we propose to use shapers as a means of mitigating error from ignoring dependencies under different schedulers. Note that the reshaping algorithms proposed in this part work not only for even deadline splitting, but for any valid local deadline assignment.

3.4.1 Base Algorithm

We first articulate an algorithm that applies to the three schedulers of Section 3.3 and for which we characterize an upper bound for the network-wise minimum required bandwidth. The algorithm essentially applies at each hop the solution derived in the one-hop setting after reshaping flows to their ingress arrival curve. According to network calculus, adding such a reshaper will not increase any flow's worst-case end-to-end delay.

Specifically, suppose flow i has an arrival curve of $\mathcal{AC}_i(t) = b_r + r_i t$, and the network has k nodes numbered from 1 to k. Denote the number of nodes traversed by flow i as k_i, and flow i's route as $M_i = [m_{i,1}, ..., m_{i,k_i}]$, where $m_{i,j} \in \{1, ..., k\}$ for all $j \in \{1, ..., k_i\}$. Suppose all nodes in M_i are unique, and for all $j \in \{1, ..., k_i - 1\}$ there exists a link from node $m_{i,j}$ to $m_{i,j+1}$ in the network. Suppose the system assigns a single-hop deadline of $\hat{d}_{i,j}$, where $\sum_{1 \leq j \leq k_i} \hat{d}_{i,j} \leq d_i$, for flow i's j^{th} hop along the route. Then we can sum up all the nodes' per-hop minimum required bandwidths (denote the procedure as $BandwidthOneHop$) to get an upper bound for the network-wise minimum required bandwidth.

ALGORITHM 1: Upper bound for the Network-wise Minimum Required Bandwidth

Function: $Bandwidth\,(\boldsymbol{AC}, \boldsymbol{M}, \boldsymbol{D}, k, n, mech) \rightarrow R$

Input: Flow arrival curves $\boldsymbol{AC}(t) = [r_1 t + b_1, ..., r_n t + b_n]$, paths $\boldsymbol{M} = [(m_{1,1}, ..., m_{1,k_1}), ..., (m_{n,1}, ..., m_{n,k_n})]$, per-hop deadlines along the path $\boldsymbol{D} = [(d_{1,1}, ..., d_{1,k_1}), ..., (d_{n,1}, ..., d_{n,k_n})]$ for each flow, the number of nodes inside the network k, and the number of flows inside the network n, network scheduling mechanism $mech$

Output: An upper bound R for the network-wise minimum required bandwidth under $mech$

// For each node, characterize all the flows traversing it;

for $i \leftarrow 1$ *to* n **do**

  for $j \leftarrow 1$ *to* k_i **do**

    $\Psi_i(\boldsymbol{M}_{i,j}) = \boldsymbol{D}_{i,j}$ // Ψ_i maps a hop number to flow i's per-hop deadline at this hop

  end

end

$R \leftarrow 0$;

for $i \in 1$ *to* k **do**

  $J \leftarrow \{j \mid i \in \boldsymbol{M}_j\}$;

  $\boldsymbol{P} \leftarrow \left[(r_{J_1}, b_{J_1}, \Psi_{J_1}(i)), ..., \left(r_{J_{|J|}}, b_{J_{|J|}}, \Psi_{J_{|J|}}(i)\right)\right]$ // $|J|$ the number of elements in J;

  $\boldsymbol{P} \leftarrow sort(\boldsymbol{P})$ // sort according to deadline decreasing order ;

  $R \leftarrow R + BandwidthOneHop(\boldsymbol{P}, n, mech)$;

end

3.4.2 Dynamic Priority Scheduler

In this part we explore the impact of reshapers given dynamic priority scheduler. We start with the base per-hop reshaping algorithm, and then identify how to further reshape some flows amd combine ingress and per-hop reshaping algorithm for better performance. Subsequently, motivated by rate-proportional scheduler, we further extend our ingress and per-hop reshaping algorithms by adopting two-slope reshapers.

Per-hop Reshaping Algorithm

From Proposition 10, SCED gives the optimal minimum required bandwidth for a single hop, *i.e.,* ingress reshaping cannot improve the current hop's performance. However, as reshaping reduces the burst size, it has the potential to benefit all downstream hops, and therefore improve the network's overall performance. From Eq 3.3, it is possible to reshape some flows without increasing the current hop's minimum required bandwidth. Specifically, flows that are not involved in setting the required link bandwidth, *i.e.,* they are not the bottleneck, experience a better delay bound than required, and this offers the opportunity to use the delay margin to further reshape them. Take Figure 3.6 as an example, since flow 2 is not the bottleneck, we can reshape it without increasing the current node's minimum required bandwidth.

For tractability, we again propose to reshape only the burst size, not the rate, of the flow. Ideally, we should consider all possible reshaping parameters, and pick the one that 1) keeps the local minimum required bandwidth, 2) does not violate any flow's per-hop deadline, and 3) reshapes each flow as much as possible. However, the reshaper values satisfying condition 1), 2) and 3) may not be unique. For example, consider a node with three flows

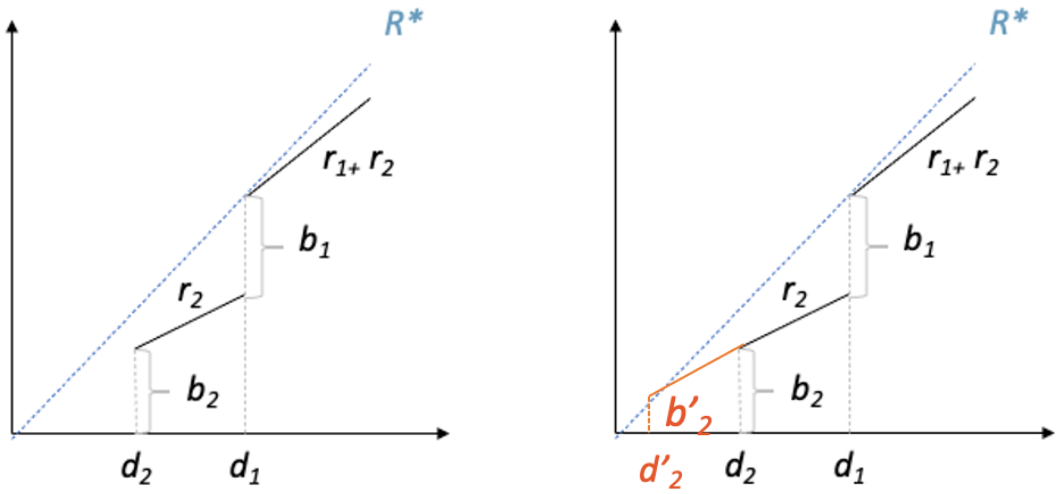


Figure 3.6: Example reshapers under dynamic priority scheduler

$(r_1, b_1, d_1) = (0.5, 0.7, 2)$, $(r_2, b_2, d_2) = (1, 0.4, 1.2)$, and $(r_3, b_3, d_3) = (1, 2.5, 1)$. In this case both $(b'_1, b'_2, b'_3) = (0.7, 0.2, 2.38)$ and $(b'_1, b'_2, b'_3) = (0.7, 0, 2.5)$ satisfy all three conditions.

To avoid this, we can either consider all valid reshaping combinations and select a globally optimal solution, or propose a mechanism to generate a unique local optimal. As the former is computationally very expensive, we adopt the latter. Specifically, we further add a restriction that the link deadline ordering must remains as before the addition of any reshaping. Formally, suppose there are n flows traversing a given hop with per-hop link deadlines $\hat{d}_1 \geq ... \geq \hat{d}_n$. We require the reshaped link deadlines, which equal the original per-hop link deadline minus the reshaping delay, to satisfy $\hat{d}'_1 \geq ... \geq \hat{d}'_n$.

With this restriction, we then propose to sequentially, from flow n to flow 1, characterize the smallest possible reshaper without violating the bandwidth or deadline constraints. Note that this procedure generates unique reshaping values. Thus we have Algorithm 2, and we can show that (see Appendix B.7 for details) that the reshapers produced by Algorithm 2 will not violate any aforementioned constraint.

ALGORITHM 2: Per-hop Reshaping Algorithm

Function: $ReshapingOneHop\left(\boldsymbol{P}, n\right) \rightarrow \left(\boldsymbol{b'}, \hat{\boldsymbol{d}}'\right)$

Input: Per-hop flow profiles $\boldsymbol{P} = \left[\left(r_1^{(1)}, r_1^{(2)}, b_1, \hat{d}_1\right), ..., \left(r_n^{(1)}, r_n^{(2)}, b_n, \hat{d}_n\right)\right]$; number of flows n;

Output: Reshaping parameter for each flow $\boldsymbol{b'} = (b'_1, ..., b'_n)$, and per-hop link deadlines after reshaping $\hat{\boldsymbol{d}}' = (\hat{d}'_1, ..., \hat{d}'_n)$

$R^* \leftarrow BandwidthOneHop\left(\boldsymbol{P}, n\right)$;

$\hat{d}'_{n+1} \leftarrow 0$;

for $i \leftarrow n$ *to* 1 **do**

$b'_i = \max\left\{0, b_i - r_i\left(\hat{d}_i - \hat{d}'_{i+1}\right)\right\}$;

if $R^* - \sum_{j \geq i} r_j > 0$ **then**

$b'_i = \max\left\{b'_i, \frac{r_i}{R^* - \sum_{j \geq i} r_j}\left[\frac{b_i - r_i \hat{d}_i}{r_i}\left(R^* - \sum_{j>i} r_j\right) + \sum_{j>i}\left(b'_j - \hat{d}'_j r_j\right)\right]\right\}$

end

$\hat{d}'_i \leftarrow \hat{d}_i - \frac{b_i - b'_i}{r_i^{(2)}}$;

end

Iterative Ingress Reshaping Algorithm

From Eq 3.3 a hop's minimum required bandwidth decreases with the arrival curves for all the flows traversing it under SCED, *i.e.,* a "smaller" arrival curve yields a lower bandwidth. Therefore, shifting all of a flow's per-hop reshaping and the corresponding reshaping delay to the first hop of its route ensures that every hop benefits from the smaller (reshaped) arrival curve, which results in a smaller overall bandwidth than that produced by the original per-hop reshaping solution. Formally,

Proposition 21. *Consider a network with k nodes and n flows, where flow $1 \leq i \leq n$ has a token-bucket arrival curve $\mathcal{AC}_i(t) = b_i + r_i t$, an end-to-end deadline of d_i, and a path of*

$\boldsymbol{M}_i = [m_{i,1}, ..., m_{i,k_i}]$, *where* M_i *contains unique nodes* $1 \leq m_{i,j} \leq k$ *for all* $1 \leq j \leq k_i$, *and there guarantees to be a link between* $m_{i,j}$ *and* $m_{i,j+1}$ *for all* $1 \leq j \leq k_i - 1$. *Suppose a per-hop reshaping mechanism reshapes flow* i *to* $b'_{i,j}$ *and assigns a link deadline of* $d_{i,j}$ *at node* $m_{i,j}$. *W.l.o.g, suppose* $b'_{i,j} \geq b'_{i,j+1}$ *for all* $1 \leq j \leq k_i - 1$. *Then under EDF, the mechanism that*

- *allocates flow* i *an ingress reshaper of* b'_{i,k_i} *at every hop along* $\boldsymbol{M}_i$*; and*
- *allocates a per-hop deadline of* $d_{i,j} - \frac{b'_{i,j-1} - b'_{i,j}}{r_i^{(2)}}$ *at node* $m_{i,j}$, *where* $1 \leq j \leq k_i$ *and for simplicity define* $b'_{i,0} = 0$,

performs no worse than the corresponding per-hop reshaping mechanism.

Remark: since reshaping delays are not the same at each hop, the ingress reshaping mechanism results in an uneven deadline splitting. This alludes to the fact that the even deadline splitting solution we start with is not the optimal deadline splitting mechanism.

Algorithm 3 specifies the procedure to construct ingress reshapers based on the proposed per-hop algorithm. Note that Algorithm 3 relies on a graph traversal order $\boldsymbol{O}$ that is consistent with all the flow routes, and enumerates all the nodes according to $\boldsymbol{O}$. As we restrict ourselves to DAG topology, such a traversal order always exists.

ALGORITHM 3: One-pass Ingress Reshaping Algorithm

Function: $IngressOnePass(\boldsymbol{AC}, \boldsymbol{M}, \boldsymbol{D}, \boldsymbol{O}, n, k) \rightarrow (\boldsymbol{b'}, \boldsymbol{D'})$

Input: Flow arrival curves $\boldsymbol{AC}(t) = [r_1 t + b_1, ..., r_n t + b_n]$, paths

$\boldsymbol{M} = [(m_{1,1}, ..., m_{1,k_1}), ..., (m_{n,1}, ..., m_{n,k_n})]$, and per-hop deadlines alone the path

$\boldsymbol{D} = [(d_{1,1}, ..., d_{1,k_1}), ..., (d_{n,1}, ..., d_{n,k_n})]$; graph traverse order $\boldsymbol{O} = [m_1, ..., m_k]$;

number of flows inside the network n; number of nodes inside the network k

Output: Flow reshaping parameters $\boldsymbol{b'} = (b'_1, ..., b'_n)$, and per-hop deadlines

$\boldsymbol{D'} = \left[(d'_{1,1}, ..., d'_{1,k_1}), ..., (d'_{n,1}, ..., d'_{n,k_n})\right]$.

$\boldsymbol{D'} \leftarrow \boldsymbol{D}$;

for $i \leftarrow 1$ *to* n **do**

  $b'_i \leftarrow b_i$;

  for $j \leftarrow 1$ *to* k_i **do**

    $\Psi(i, \boldsymbol{M}_{i,j}) = j$, // $\Psi(i, \boldsymbol{M}_{i,j})$ gives the index of node $\boldsymbol{M}_{i,j}$ in $\boldsymbol{M}_i$

  end

end

for $i \in \boldsymbol{O}$ **do**

  $J \leftarrow \{j \mid i \in \boldsymbol{M}_j\}$ // get all the flows traversing node i ;

  $\boldsymbol{P} \leftarrow \left[\left(r_{J_1}, b'_{J_1}, \boldsymbol{D}_{J_1,\Psi(J_1,i)}\right), ..., \left(r_{J_{|J|}}, b'_{J_{|J|}}, \boldsymbol{D}_{J_{|J|},\Psi(J_{|J|},i)}\right)\right]$;

  $\boldsymbol{P}, \boldsymbol{J} \leftarrow sort(\boldsymbol{P})$ // sort $\boldsymbol{P}$ by deadline decreasing order ;

  $\left(\boldsymbol{b'}, \hat{\boldsymbol{d}}'\right) \leftarrow ReshapingOneHop\left(\boldsymbol{P}, |\boldsymbol{J}|\right)$;

  for $l \leftarrow 1$ *to* $|\boldsymbol{J}|$ **do**

    $b'_{\boldsymbol{J}_l} \leftarrow \boldsymbol{b'}_l$, $\boldsymbol{D'}_{\boldsymbol{J}_l,\Psi(\boldsymbol{J}_l,i)} = \hat{\boldsymbol{d}}'_l$

  end

end

$\boldsymbol{b'} \leftarrow \{b'_1, ..., b'_n\}$

As moving all reshaping to the ingress updates flow arrival curves and per-hop deadlines at intermediate hops, it introduces the possibility of new per-hop improvements based on the updated flow profiles/ This suggests iteratively applying Algorithm 3 to the updated flow profiles. As Algorithm 3 only decreases the upper bound, and the upper bound is greater than 0, *i.e.,* has a lower bound, the proposed iterative algorithm is guaranteed to converge.

ALGORITHM 4: Iterative Ingress Reshaping Algorithm

Function: $IngressIterative(\boldsymbol{AC}, \boldsymbol{M}, \boldsymbol{D}, \boldsymbol{O}, n, k, \epsilon) \rightarrow (\boldsymbol{b'}, \boldsymbol{D'}, R)$

Input: Flow arrival curves $\boldsymbol{AC}(t) = [r_1 t + b_1, ..., r_n t + b_n]$, paths $\boldsymbol{M} = [(m_{1,1}, ..., m_{1,k_1}), ..., (m_{n,1}, ..., m_{n,k_n})]$, and per-hop deadlines alone the path $\boldsymbol{D} = [(d_{1,1}, ..., d_{1,k_1}), ..., (d_{n,1}, ..., d_{n,k_n})]$; graph traverse order $\boldsymbol{O} = [m_1, ..., m_k]$; number of flows inside the network n, number of nodes inside the network k, granularity ϵ

Output: Flow reshaping parameters $\boldsymbol{b'} = (b'_1, ..., b'_n)$, flow per-hop deadlines $\boldsymbol{D'} = \left[(d'_{1,1}, ..., d'_{1,k_1}), ..., (d'_{n,1}, ..., d'_{n,k_n})\right]$, and upper bound for the network-wise minimum required bandwidth according to $\boldsymbol{b'}$ and $\boldsymbol{D'}$ under EDF.

$R_{crt} \leftarrow Bandwidth(\boldsymbol{AC}, \boldsymbol{M}, \boldsymbol{D}, k, n)$;

$R_{prev} \leftarrow R_{crt} + 2\epsilon$;

$\boldsymbol{b'} \leftarrow [b_1, ..., b_n]$;

while $R_{prev} - R_{crt} > \epsilon$ **do**

  $R_{prev} \leftarrow R_{crt}$;

  $(\boldsymbol{b'}, \boldsymbol{D}) \leftarrow IngressOnePass(\boldsymbol{AC}, \boldsymbol{M}, \boldsymbol{D}, \boldsymbol{O}, n, k)$;

  $\boldsymbol{AC} \leftarrow [r_1 t + \boldsymbol{b'}_1, ..., r_n t + \boldsymbol{b'}_n]$;

  $R_{crt} \leftarrow Bandwidth(\boldsymbol{AC}, \boldsymbol{M}, \boldsymbol{D}, k, n)$

end

$R \leftarrow R_{crt}$

Remark: by graph decomposition, we can extend Algorithms 3 and 4 to general directed graphs. Specifically, after decomposing the graph into DAGs, we can apply our ingress reshaping algorithms and update the corresponding flow profiles for each DAG, and then iterate across DAGs.

Evaluation

In this part, we study the performance of the iterative ingress reshaping solution we have just introduced. We will not focus on per-hop and ingress reshaping algorithms, because we have proved that iterative ingress reshaping performs no worse than them.

For ease of evaluation, we rely on a parking lot topology for our experiments. A parking lot topology consists of N nodes, and there exists only one directed link from node i to $i+1$, where $1 \leq i \leq N-1$. Inside the system there are two types of traffic: main traffic and cross traffic. The main traffic traverses all the nodes inside the system. Whereas the number of nodes traversed by cross traffic is strictly less than N. For every node there is cross traffic entering from it. All the cross traffic traverses the same number of hops ($C < N$), and have the same traffic profiles regardless of the node they enter from. Figure 3.7 shows an example parking lot topology where $N = 5$ and $C = 2$.

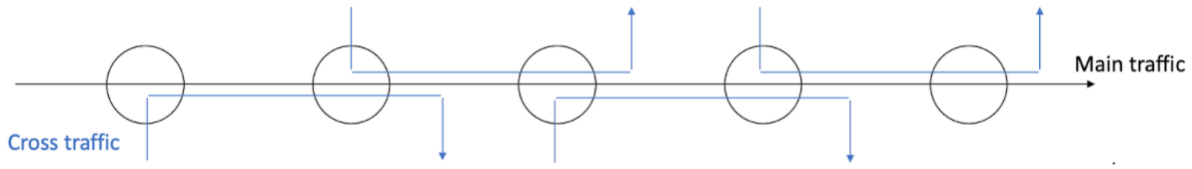
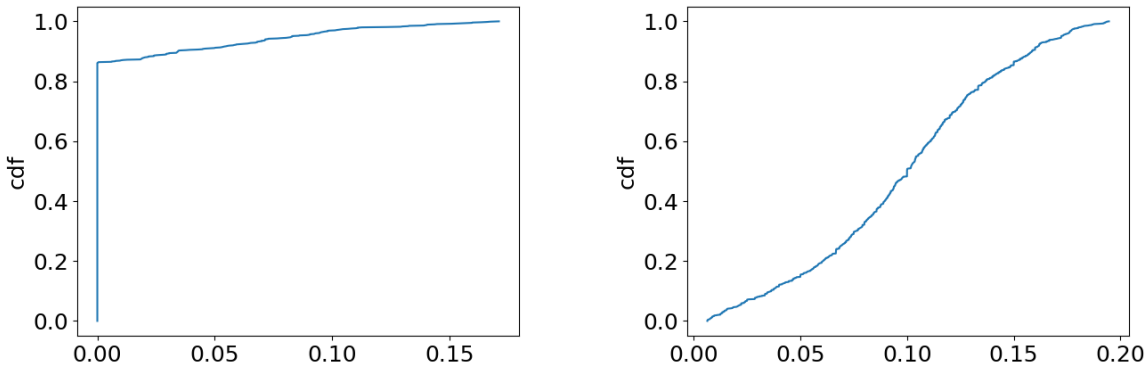


Figure 3.7: A parking lot topology with $N = 5$ and $C = 2$.

We assume that both main and cross traffic consist of three types of flows. For each flow, we draw its rate uniformly and independently from $U(1, 10)$, and draw its burst size uniformly



(a) Main Traffic Deadlines: $[1, 0.1, 0.01]$ (b) Main Traffic Deadlines: $[6, 0.6, 0.06]$

Figure 3.8: Relative bandwidth improvement of SCED + iterative ingress reshaping compared with SCED when $N = 5$ and $C = 1$

and independently from $U(1, 30)$. We consider two cases for flows' end-to-end deadlines. For the first case, both main and cross traffic's three types of flows have end-to-end deadlines of $[1, 0.1, 0.01]$. For the second case, cross traffic has end-to-end deadlines of $[1, 0.1, 0.01]$, and main traffic has end-to-end deadlines of $[6, 0.6, 0.06]$. We run 1000 cases for each comparison.

Figure 3.8 shows the relative improvement of iterative ingress reshaping under SCED compared with pure SCED when the main traffic has end-to-end deadlines of $[1, 0.1, 0.01]$ (Figure 3.10a) and $[6, 0.6, 0.06]$ (Figure 3.10). Iterative ingress reshaping gives better performance when the main traffic has larger end-to-end deadlines. This is intuitive as larger end-to-end deadlines leave more room for reshaping.

Two-slope Ingress Reshaper

We further compare iterative ingress reshaping algorithm with rate proportional mechanism that assigns flow (r, b, d) a rate of $\max\left\{r, \frac{b}{d}\right\}$ at every hop. Surprisingly, in a large fraction

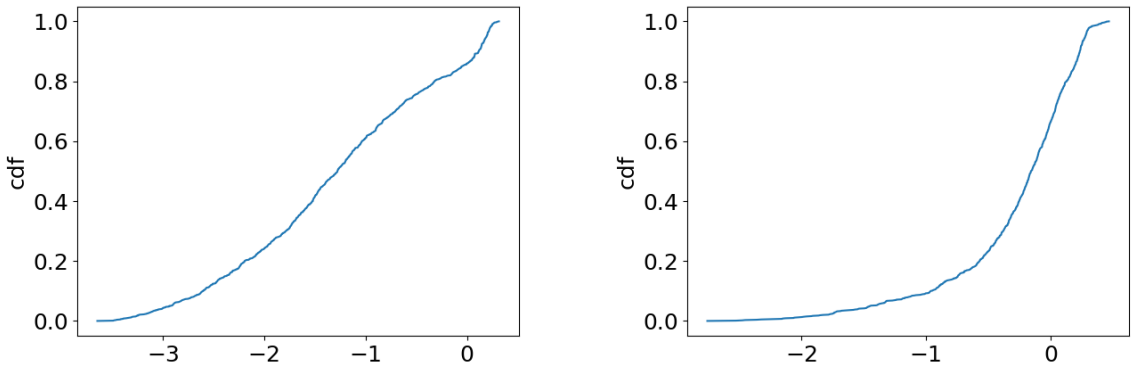


Figure 3.9: Relative improvement of iterative ingress reshaping + SCED compared with rate proportional when $N = 5$ and $C = 1$

of our numerical experiments reshaping + SCED perform worse than the simple rate proportional. As shown in Figure 3.9, rate proportional strictly outperforms[51] iterative ingress reshaping in more than 80% of the experiments when main traffic has end-to-end deadlines of $[1, 0.1, 0.01]$, and in more than 60% when main traffic has end-to-end deadlines of $[6, 0.6, 0.06]$.

Note that from the perspective of the trade-off between smoother burst and in-network flexibility, both SCED and rate proportional are extreme points. SCED relies on local deadline differences to allocate bandwidth to flows as efficiently as possible. Whereas rate proportional spreads bursts and accounts for benefit to downstream nodes as accurately as possible, but leaves little to no flexibility in dynamically allocating bandwidth to flows at intermediate hops.

To benefit from both smoother bursts and in-network flexibility, we propose to emulate rate proportional in our ingress reshaper to ensure that SCED can always outperform it while still exploring solutions where it can benefit from greater flexibility in local deadline

[51]The units of the x-axis are in relative bandwidth "improvement" of iterative reshaping + SCED over rate proportional, *i.e.,* a value of -1 means that rate proportional is 100% better.

assignments. Specifically, we propose to reshape flow (r, b, d) using a 2-slope token-bucket reshaper $\min\left\{r^{(1)}t, b' + r^{(2)}t\right\}$, where $r^{(1)} \geq \max\left\{r, \frac{b}{d}\right\}$, *i.e.,* the rate-proportional rate. Similar as before, we set $r^{(2)} = r$. Given both rates, we then rely on our previous iterative ingress reshaping algorithm to get b'. Note that when $r^{(1)} = \max\left\{r, \frac{b}{d}\right\}$, the reshaper defaults to rate proportional mechanism; whereas as when $r^{(1)} = \infty$, it defaults to 1-slope token-bucket reshaper. We search from $\max\left\{r, \frac{b}{d}\right\}$ to ∞ to find the best $r^{(1)}$.

By extending ingress reshaping from 1-slope to 2-slope, we generalize the per-hop arrival curves as well. We modify the per-hop reshaping algorithm to reflect this more general arrival curve and rely on a binary search to characterize the minimum reshaper.

ALGORITHM 5: Single-hop Reshaping Algorithm

Function: $ReshapingOneHop\,(\boldsymbol{P}, n) \rightarrow \left(\boldsymbol{b'}, \hat{\boldsymbol{d}}'\right)$

Input: Flow profiles $\boldsymbol{P} = \left[\left(r_1^{(1)}, r_1^{(2)}, b_1, \hat{d}_1\right), ..., \left(r_n^{(1)}, r_n^{(2)}, b_n, \hat{d}_n\right)\right]$; flow number n

Output: Reshaping parameter for each flow $\boldsymbol{b'}$, and updated per-hop link deadlines $\hat{\boldsymbol{d}}'$

$R^* \leftarrow BandwidthOneHop\,(\boldsymbol{P}, n)$;

$\hat{d}'_{n+1} \leftarrow 0, t_{n+1} \leftarrow 0$;

for $i \leftarrow 1$ *to* n **do**

  $\hat{d}'_i \leftarrow \hat{d}_i, b'_i \leftarrow b_i$, $t_i \leftarrow \frac{b_i}{r_i^{(1)} - r_i^{(2)}}$ when $r_i^{(1)} \neq r_i^{(2)}$; and 0 otherwise

end

for $i \leftarrow n$ *to* 1 **do**

  if $r_i^{(2)} = r_i^{(1)}$ **then**

    $b'_i \leftarrow b_i$;

    continue

  end

  $\mathbb{T} \leftarrow \left\{\hat{d}'_j + t_j \mid \hat{d}'_i \leq \hat{d}'_j + t_j < \hat{d}'_i + t_i\right\}$;

  for t *in* $\mathbb{T}$ **do**

    if $\sum_{1 \leq j \leq n} \mathcal{AC}_j(t - \hat{d}'_j \mid b'_j) = R^* t$ **then**

      goto next iteration $(i \leftarrow i + 1)$

    end

  end

  $left \leftarrow \left[b_i - r_i^{(2)}(\hat{d}_i - \hat{d}'_{i+1})\right]^+$;

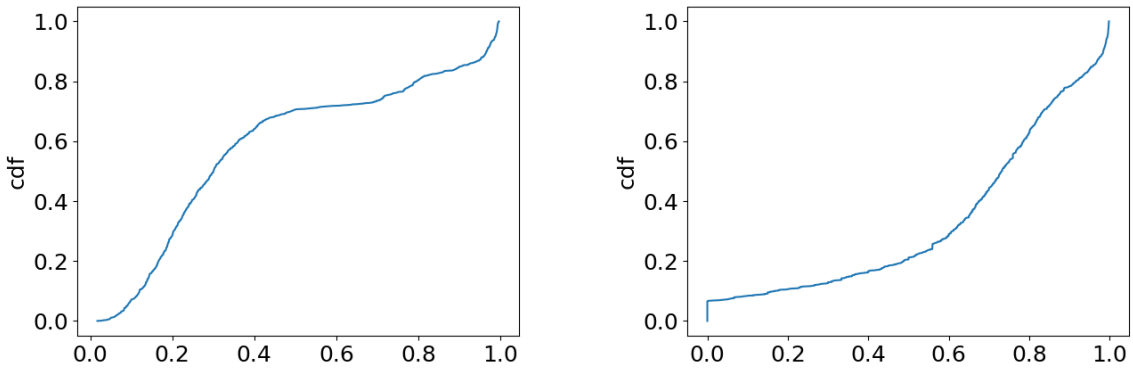
  $right \leftarrow b_i$;

  $b'_i = binary\_search(left, right, \boldsymbol{b'}, \boldsymbol{d'}, n, R^*)$;

  $\hat{d}'_i \leftarrow \hat{d}_i - \frac{b_i - b'_i}{r_i^{(2)}}$;

  $t_i = \frac{b'_i}{r_i^{(1)} - r_i^{(2)}}$

end



(a) Compared with rate proportional

(b) Compared with 1-slope SCED

Figure 3.10: Relative improvement of 2-slope ingress reshaping + iterative ingress reshaping + SCED with main traffic deadlines of $[1, 0.1, 0.01]$ when $N = 5$ and $C = 1$,

We then explore the performance of 2-slope ingress reshaping algorithm through numerical experiments. We again rely on the parking lot topology and random flow profiles. Specifically, we compare 2-slope ingress reshaping + Iterative ingress reshaping + SCED with rate proportional, and compare 2-slope ingress reshaper + SCED with 1-slope ingress reshape + SCED.

Numerical experiments show that adding 2-slope reshaping to iterative ingress reshaping brings considerable benefit. Take Figure 3.10 as an example. In this case, it brings an average relative improvement of around 43.93% compared with rate proportional, and that of 68.10% compared with 1-slope iterative ingress reshaping.

3.4.3 Static Priority and FIFO Scheduler

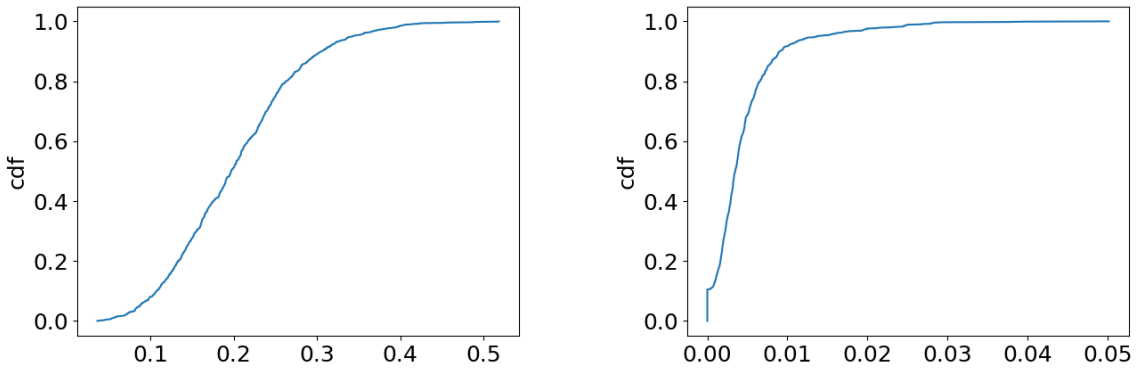
For static priority and FIFO scheduler, by running the single-hop reshaping algorithm at each hop, and then using the reshaped arrival curve as the arrival curve at the next hop along its route, we get a multi-hop reshaping procedure. Combining it with Propositions 16, 17, 19 and 20 gives per-hop greedy reshaping algorithm for static priority and fifo schedulers.

Note that the proposed (iterative) ingress reshaping algorithm for dynamic priority will not work for static priority or FIFO scheduler. This is because under these two schedulers, decoupling the reshaper and scheduling delays may increase the hop's minimum required bandwidth. From Propositions 14 and 18, the worst-case end-to-end delay is less than the sum of the worst-case delay inside reshaper and the worst-case delay at the link. Therefore, decreasing the burst and correspondingly the link delay does not guarantee to result in the same minimum required bandwidth.

Evaluation

In this part, we test the performance of greedy reshaping under static priority and FIFO, where we rely again on the parking lot topology and random flow profiles. Besides the per-hop reshaping algorithm, we also explore the potential benefit of ingress reshaping simply by adding an ingress reshaper and varying its configuration.

We first study the performance of our greedy per-hop reshaping algorithm. For both static priority and FIFO, we compare the bandwidth upper bound generated by our per-hop reshaping algorithm with that generated by the per-hop reshaping algorithm that reshapes each flow to the original arrival curve at each hop. Note that our assumption of the availability of per-hop reshaping, be it to the original arrival curve or using our one-hop algorithm, is intended to ensure reasonable end-to-end performance in spite of the schedulers' relative simplicity. In the absence of per-hop reshaping, arrival curves at each hop are determined based on the service curves of the previous hop, and it is well-known that the resulting worst-case burstiness increases rapidly (a cascade effect), which in turn forces the use of very large bandwidth to meet end-to-end bounds.



(a) FIFO: Greedy vs. base reshaping

(b) Static priority: Greedy vs. base reshaping

Figure 3.11: Relative bandwidth improvement of per-hop greedy reshaping when main traffic deadlines are $[6, 0.6, 0.06]$, $N = 5$ and $C = 1$.

Figure 3.11 gives representative examples of the per-hop reshaping Algorithm's performance under static priority and FIFO schedulers when the main traffic has end-to-end deadlines of $[6, 0.6, 0.06]$. Under FIFO scheduler, per-hop reshaping brings an average relative improvement of 20.28%. Whereas under static priority, it brings an average relative improvement of only 0.47%. As we shall see later, although reshaping generates a smaller improvement with static priority than with FIFO, the absolute performance of reshaping + static priority is usually better than reshaping + FIFO.

We then study the potential benefit of ingress reshaping by comparing Ingress reshaping + greedy algorithm with greedy algorithm. Numerical experiments show that when cross traffic and main traffic have the same end-to-end deadline, there is almost no benefit. Whereas when the main traffic has end-to-end deadlines of $[6, 0.6, 0.06]$, there is positive improvement in most of the cases for both static priority and FIFO. Nevertheless, the improvement is usually very small (less than 0.1%)

3.4.4 Comparing the Relative Benefits of Schedulers

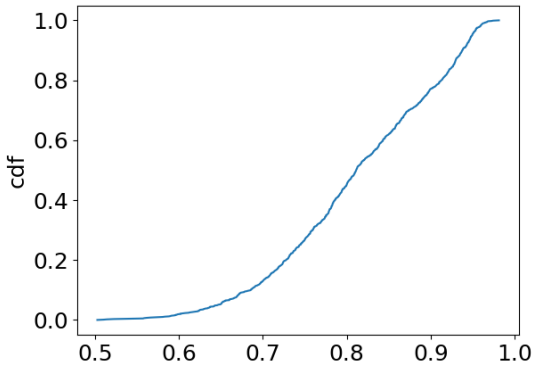
In this section, we compare the performance of dynamic priority, static priority and FIFO schedulers with reshaping. As before, we rely on the parking lot topology and random flow profiles.

We first compare reshaping + SCED with greedy per-hop reshaping + static priority and greedy per-hop reshaping + FIFO, where for SCED we consider both 1-slope and 2-slope reshapers. After that, we compare FIFO and static priority under greedy per-hop reshaping.

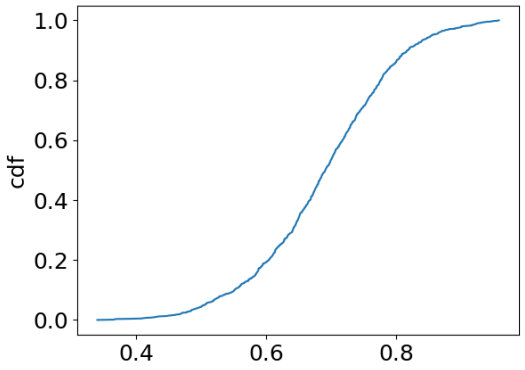
From our experiments, 1-slope iterative ingress reshaping + SCED usually outperforms per-hop reshaping + FIFO and per-hop reshaping + static priority. Figure 3.12 gives example comparisons between SCED and FIFO, and we can see that SCED usually outperforms FIFO by more than 50%. Figure 3.13 gives example comparisons between SCED and static priority. The improvements here are less than those under FIFO, and when main traffic has small deadlines, SCED brings little improvement.

Note that the benefits of SCED increases with main traffic deadlines compared with both FIFO and static priority. Remember that iterative ingress reshaping + SCED takes the whole network into consideration, whereas for FIFO and static priority we characterize the per-hop reshaping parameters greedily. Thus, though larger end-to-end deadlines make more room for reshaping for all three schedulers, intuitively iterative ingress reshaping + SCED exploits the additional room more efficiently.

Under 2-slope reshapers, SCED performs better than FIFO and static priority in all of our experiments. Figure 3.14 gives example improvements for 2-slope reshaper + iterative ingress reshaping + SCED. Comparing Figure 3.13a with Figure 3.14b, we see that by adding 2-slope reshapers, the performance gap between SCED and static priority enlarges significantly.

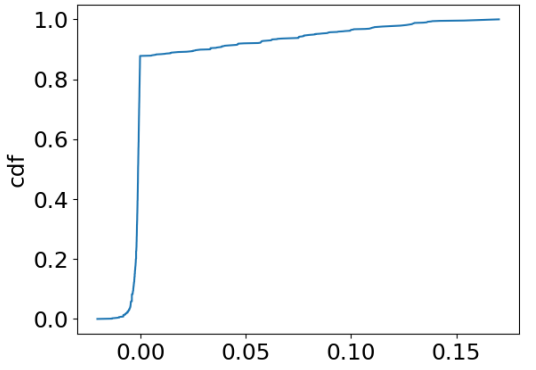


(a) main traffic deadline: $[1, 0.1, 0.01]$

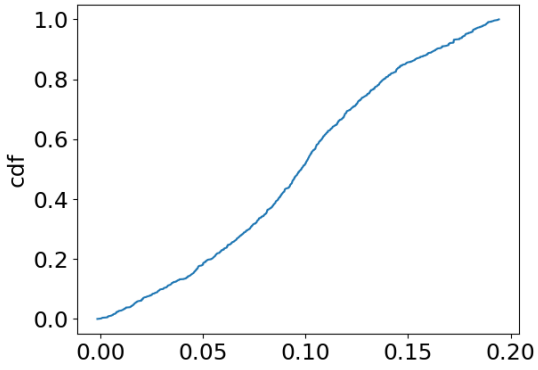


(b) main traffic deadline: $[6, 0.6, 0.06]$

Figure 3.12: Relative bandwidth improvement of 1-slope iterative ingress reshaping + SCED compared with greedy per-hop FIFO when $N = 5$ and $C = 1$.

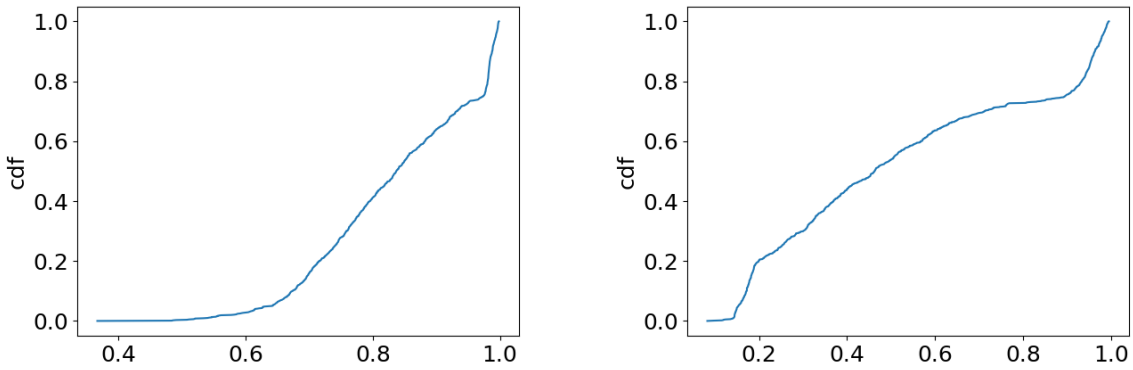


(a) main traffic deadline: $[1, 0.1, 0.01]$



(b) main traffic deadline: $[6, 0.6, 0.06]$

Figure 3.13: Relative bandwidth improvement of 1-slope iterative ingress reshaping + SCED compared with greedy per-hop static priority when $N = 5$ and $C = 1$.



(a) 2-slope iterative ingress reshaping + SCED vs. per-hop reshaping + FIFO

(b) 2-slope iterative ingress reshaping + SCED vs. per-hop reshaping + static priority

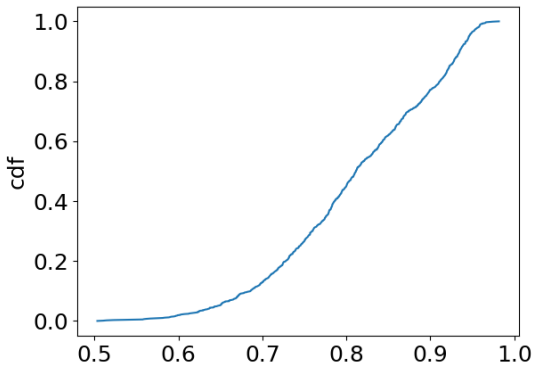
Figure 3.14: Relative bandwidth improvement of 2-slope iterative ingress reshaping + SCED when main traffic deadlines are $[1, 0.1, 0.01]$, $N = 5$ and $C = 1$.

In most of our experiments, given their corresponding greedy per-hop reshapers, static priority stricty outperforms FIFO scheduler. Figure 3.15 shows example comparisons between per-hop reshaping + static priority and per-hop reshaping + FIFO. Here greedy reshaping + static priority usually outperforms greedy reshaping + FIFO by 50%.

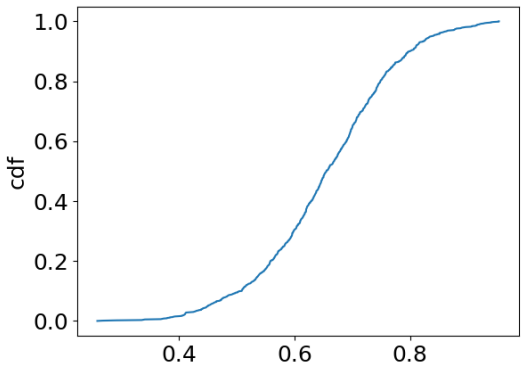
3.5 Summary

The work has investigated the question of minimizing the bandwidth required to meet worst case latency bounds for a set of rate-controlled (through a token bucket) flows in a basic one-hop setting. The investigation was carried for schedulers of different complexity.

We first consider the single-node case, and characterized the minimum required bandwidth independent of schedulers, and showed that an EDF scheduler could realize all flows' deadlines under such bandwidth. Motivated by the need for lower complexity solutions, we then proceeded to explore simpler static priority and fifo schedulers. It derived the minimum required bandwidth for both, but more interestingly established how to optimally reshape



(a) Main Traffic Deadlines $[1, 0.1, 0.01]$



(b) Main Traffic Deadlines $[6, 0.6, 0.06]$

Figure 3.15: Relative bandwidth improvement of static priority over FIFO under greedy per-hop reshaping when $N = 5$ and $C = 1$.

flows to reduce the bandwidth needed while still meeting all deadlines. The relative benefits of such an approach were illustrated numerically for a number of different flow combinations, which showed how "intelligent" reshaping could enable simpler schedulers to perform nearly as well a more complex ones across a range of different configurations.

Next we extended our single-node case results to networks with multiple nodes. For static priority and fifo schedulers, there exists a natural extension, *i.e.,* running the proposed single-hop reshaping mechanism at each node, and use the reshaped departure curve (reshaped by the optimal reshaper for the current node) as the arrival curve for the next hop along the flow's route. For EDF, where reshaping does not improve the single-node performance, we first proposed applying a single-hop algorithm at each hop that 1) keeps the current hop's minimum required bandwidth, and 2) decreases certain flows' bursts. We then extended this algorithm by showing that reshaping could be moved to the ingress, and then subsequently applying an iterative procedure that is guaranteed to converge and generate better results compared to the per-hop algorithm. Finally, towards ensuring that we could always outperform a solution based on a rate proportional algorithm, we extended our solution with EDF to support two-slope shapers and evaluated the benefits this could afford. From numerical

experiments, these "intelligent" reshaping algorithms bring considerable improvement in the multiple-node environment.

Chapter 4

Conclusion

4.1 Contributions

Customers inside the cloud computing market are heterogeneous in several aspects, e.g., willingness to pay and performance requirement. By taking advantage of trade-offs created by these heterogeneities, the service provider can realize a more efficient system. This thesis is concerned with the role of pricing in realizing those improvements and leveraging heterogeneity, and with methods to improve utilization of network resources through traffic engineering. Particularly,

- For the pricing part, we considered a cloud provider that seeks to maximize its revenue by offering services with different tradeoffs between cost and timeliness of job completion. Our focus was on exploiting heterogeneity across jobs in terms of value and sensitivity to execution delay, with a joint distribution that determines their relationship across the user population. We characterized optimal (revenue maximizing) pricing strategies and, in the case of spot instances, optimal bidding strategies as well

as identify conditions under which bidding at a fixed price is optimal.We showed that correlation between delay sensitivity and job value needs to exceed a certain threshold for a service offering that differentiates based on speed of execution to be beneficial to the provider. We further assessed the results' robustness under more general assumptions, and we offered guidelines for users and providers.

- For the network part, we investigated minimizing the bandwidth required to meet worst case latency bounds for rate-controlled (through a token bucket) flows. We start with a basic one-hop setting, characterized the minimum required bandwidth independent of schedulers, and showed that an EDF scheduler could realize all flows' deadlines under such bandwidth. Motivated by the need for lower complexity solutions, we then explored simpler static priority and fifo schedulers. It derived the minimum required bandwidth for both, but more interestingly established how to optimally reshape flows to reduce the bandwidth they needed to meet all deadlines. Based on the one-hop results, we then proposed reshaping mechanism to improve efficiency for networks with multiple nodes under different schedulers.

4.2 Future Work

There are many directions in which the work can be extended to better account for the many, often fast changing, facets of cloud offerings.

For the pricing part, one direction is to consider other forms of correlation in job profiled, *e.g.,* by allowing value or delay sensitivity to depend on a job's size. Another direction is to introduce some form of memory in the evolution of spot prices to capture connections to resource availability, *e.g.,* as in [199]. This could also be extended to include geographic

restrictions, instances heterogeneity and the ability to schedule resources in advance. Another potentially interesting extension is to consider sequences/bursts of jobs, as opposed to isolated jobs, possibly with some uncertainty in either their arrivals or sizes. In that context, incorporating services such as burstable instances available from Amazon[52], Google[53] and Microsoft[54] would also be of interest. Those services offer an intermediate option between a fast and a slow processor, namely, a slow processor that can burst at a higher speed for a limited amount of time. Nevertheless, the models developed in the chapter provide useful initial insight into when and why a spot service (or equivalent) can be useful to providers and users, and offer guidelines for how bidding and pricing strategies can be set to realize desirable outcomes.

For the network part, one direction is to consider more complicated deadline splitting mechanisms in the multiple-node network. In this work we adopt the simple evenly splitting mechanism, and from numerical experiments we know it is not optimal. Despite for EDF our algorithms will eventually generate uneven deadline splitting, they still rely on even deadline spitting as the starting point. Since our algorithms will never increase the per-hop deadline, this starting point set an upper bound for at each hop. Another direction is to consider systems with cycles. This extension entails significant added complexity with exact solutions likely intractable. However, it represents an essential next step to making the results more broadly applicable. In addition, incorporating the impact of routing decisions that affect the set of flows interacting at every hop, likely also represents an important extension. On a more punctual front, as illustrated in Section 3.3.5, generalizing the results of

[52]Retrieved 2022, Jan 2 from `https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/burstable-performance-instances.html`.

[53]Retrieved 2022, Jan 2 from `https://cloud.google.com/compute/docs/machine-types\#sharedcore`.

[54]Retrieved 2022, Jan 2 from `https://docs.microsoft.com/en-us/azure/virtual-machines/windows/b-series-burstable`.

Appendix B.5 to multiple flows to determine how to best group flows with different deadlines when relying on a static priority scheduler remains of practical interest. Finally, while initial evidence (see Corollary 46 of Appendix B.6) indicates that more complex, two-slopes shapers do not help in the simple case involving just two flows, exploring different types of reshapers, *i.e.,* based on more general traffic envelopes as well as extending to interleaved shapers in the multi-hop case, is also of interest.

References

[1] V. Abhishek, I. A. Kash, and P. Key. Fixed and market pricing for cloud services. In *Proc. NetEcon Workshop*, Orlando, FL, March 2012.

[2] V. Abhishek, I. A. Kash, and P. Key. Fixed and market pricing for cloud services. *CoRR*, abs/1201.5621, 2017. Available at `http://arxiv.org/abs/1201.5621`.

[3] P. Afèche. Incentive-compatible revenue management in queueing systems: Optimal strategic delay. *Management & Service Operations Management*, 15(3):423–443, Summer 2013.

[4] A. Aguiar and J. Gross. Wireless channel models. Tech. Rep. TKN-03-007, Telecom. Netw. Group, Technische Universität Berlin, 2003.

[5] A.Kumar, S.Jain, U.Naik, and A.Raghuraman. BwE: Flexible, hierarchical bandwidth allocation for WAN distributed computing. In *Proc. ACM SIGCOMM'15*, London, United Kingdom, August 2015. ACM.

[6] J.N. Al-Karaki and A.E. Kamal. Routing techniques in wireless sensor networks: A survey. *IEEE Wireless Comm.*, 11(6), December 2004.

[7] M. Alizadeh, T. Edsall, S. Dharmapurikar, R. Vaidyanathan, K. Chu, A. Fingerhut, V. T. Lam, F. Matus, R. Pan, N. Yadav, and G. Varghese. CONGA: distributed congestion-aware load balancing for datacenters. In *Proc. ACM SIGCOMM'14*, Chicago, IL, August 2014.

[8] A. Andrzejak, D. Kondo, and S. Yi. Decision model for cloud computing under sla constraints. Technical report, Inria Research Report, 2010.

[9] D. Anick, D. Mitra, and M.M. Sondhi. Stochastic theory of a data handling systems with multiple sources. *The Bell Sys. Tech. J.*, 61(8), October 1982.

[10] J.G. Apostolopoulos and M.D. Trott. Path diversity for enhanced media streaming. *IEEE Comm. Mag.*, 42(8), August 2004.

[11] B Arzani, R Guerin, and A Ribeiro. A distributed routing protocol for predictable rates in wireless mesh networks. In *Proc. IEEE ICNP*, Austin, TX, October 2012.

[12] Association for Computing Machinery. *ACM Visual Identity Standards*, 2007. `http://identitystandards.acm.org`.

[13] AWS remains dominant despite Microsoft and Google growth surges. `https://www.srgresearch.com/articles/aws-remains-dominant-despite-microsoft-and-google-growth-surges`, February 2016.

[14] Leveraging amazon EC2 spot instances at scale. White paper, Amazon Web Services (AWS), March 2018. Retrieved March 4, 2019 at `https://d1.awsstatic.com/whitepapers/cost-optimization-leveraging-ec2-spot-instances.pdf`.

[15] A. Aziz. *Bringing stability to wireless mesh networks*. PhD thesis, École Polytechnique Fédérale de Lausanne, March 2011.

[16] T. Babbitt, C. Morrell, and B. Szymanski. Self-selecting reliable path routing in diverse wireless sensor network environments. In *Proc. IEEE ISCC*, Sousse, Tunisia, July 2009.

[17] Martin J Bailey. The demand revealing process: To distribute the surplus. *Public Choice*, 91(2):107–126, 1997.

[18] Ron Banner and Ariel Orda. Multipath routing algorithms for congestion minimization. *IEEE/ACM Trans. Netw.*, 15(2), April 2007.

[19] J. Barr. Amazon EC2 update – streamlined access to spot capacity, smooth price changes, instance hibernation. AWS News Blog, November 2017. Retrieved 2019, July 25 from `https://aws.amazon.com/blogs/aws/amazon-ec2-update-streamlined-access-to-spot-capacity-smooth-price-changes-instance-hibernation/`.

[20] D. Bartoletti. Hybrid cloud management solutions, Q1 2016 – Tools and technology: The cloud computing playbook. The Forrester Wave™, Forrester Research, January 2016.

[21] BBC video factory and iPlayer. Case study, AWS. Retrieved March 4, 2019 at `https://www.elemental.com/resources/case-studies/bbc-video-factory-iplayer`.

[22] M. Ben-Akiva, D. McFadden, and K. Train. Foundations of stated preference elicitation: Consumer behavior and choice-based conjoint analysis. *Foundations and Trends® in Econometrics*, 10(1-2):1–144, 2019.

[23] M. Ben-Yehuda, O.Agmon Ben-Yehuda, and D. Tsafrir. The nom profit-maximizing operating system. In *Proc. ACM International Conference on Virtual Execution Environments(VEE)*, Melbourne, VIC, Australia, April 2016.

[24] O. Agmon Ben-Yehuda, M. Ben-Yehuda, A. Schuster, and D. Tsafrir. Deconstructing amazon EC2 spot instance pricing. *ACM Trans. Econ. Comput.*, 1(3), September 2013.

[25] O. Agmon Ben-Yehuda, M. Ben-Yehuda, A. Schuster, and D. Tsafrir. The rise of raas: The resource-as-a-service cloud. *Communications of the ACM*, 2014.

[26] T. Benson, A. Anand, A. Akella, and M. Zhang. MicroTE: Fine grained traffic engineering for data centers. In *Proc. ACM CoNEXT*, Tokyo, Japan, December 2011.

[27] K. P. Birman. *Reliable distributed systems – Technologies, web services and applications*. Springer, 2005.

[28] G. Blakowski and R. Steinmetz. A media synchronization survey: Reference model, specification, and case studies. *IEEE J. Select. Areas Comm.*, 14(1), January 1996.

[29] J. Bogle, N. Bhatia, M. Ghobadi, I. Menache, A. Valadarsky Bjørner, and M. Schapira. TeaVar: Striking the right utilization-availability balance in WAN traffic engineering. In *Proc. ACM SIGCOMM*, Beijing, China, August 2019.

[30] R. Bolla, R. Bruschi, F. Davoli, and F. Cucchietti. Energy efficiency in the future internet: A survey of existing approaches and trends in energy-aware fixed network infrastructures. *IEEE Communications Surveys Tutorials*, 13(2):223–244, Second 2011.

[31] T. Bonald. The erlang model with non-poisson call arrivals. In *Proc. SIGMetrics/Performance*, Saint Malo, France, June 2006.

[32] S. Bondorf, P. Nikolaus, and J. B. Schmitt. Quality and cost of deterministic network calculus: Design and evaluation of an accurate and fast analysis. *Proc. ACM Meas. Anal. Comput. Syst.*, 1(1), June 2017.

[33] J. Bort. Amazon Web Services is bigger than its next 4 competitors combined as cloud became a $70 billion market last year. *Business Insider*, February 6 2019. Available at `https://www.businessinsider.com/aws-market-share-dominates-70-billion-cloud-market-2019-2`.

[34] R. Bossche, K. Vanmechelen, and J. Broeckhove. Online cost-efficient scheduling of deadline-constrained workloads on hybrid clouds. *Future Generation Computer Systems*, 29, 2013.

[35] A. Bouillard. *Algorithms and efficiency of Network calculus*. Habilitation à diriger des recherches, Ecole Normale Supérieure (Paris), April 2014.

[36] A. Bouillard. Stability and performance guarantees in networks with cyclic dependencies, 2018.

[37] A. Bouillard. Trade-off between accuracy and tractability of network calculus in FIFO networks, 2020.

[38] A. Bouillard, L. Jouhet, and E. Thierry. Tight performance bounds in the worst-case analysis of feed-forward networks. In *Proc. IEEE INFOCOM*, pages 1–9, San Diego, CA, March 2010.

[39] A. Bouillard and G. Stea. Exact worst-case delay in FIFO-multiplexing feed-forward networks. *IEEE/ACM Transactions on Networking*, 23(5):1387–1400, 2015.

[40] A. Bouillard and G. Stea. Worst-case analysis of tandem queueing systems using network calculus, 2016.

[41] I. Brace. *Questionnaire design: How to Plan, Structure and Write Survey Material for Effective Market Research*. Kogan Page Publishers, 2008.

[42] M. Branscombe. Here's how the Software-Defined Network that powers Azure works, September 2017. Accessed on 9/11/19, available at `https://www.datacenterknowledge.com/design/here-s-how-software-defined-network-powers-azure-works`.

[43] Rogério Brito. *The algorithms bundle*, August 2009. `http://www.ctan.org/pkg/algorithms`.

[44] T. Burger. How fast is realtime? human perception and technology. PubNub, February 2015. See `https://www.pubnub.com/blog/2015-02-09-how-fast-is-realtime-human-perception-and-technology/`.

[45] G.C. Buttazzo. HARTIK: a real-time kernel for robotics applications. In *Proc. IEEE RTSS'93*, Raleigh-Durham, NC, December 1993.

[46] Jin Cao, William S Cleveland, Dong Lin, and Don X Sun. On the nonstationarity of Internet traffic. In *SIGMETRICS '01: Proceedings of the 2001 ACM SIGMETRICS international conference on Measurement and modeling of computer systems*. ACM Request Permissions, June 2001.

[47] David Carlisle. *The textcase package*, October 2004. `http://www.ctan.org/pkg/textcase`.

[48] Ruggiero Cavallo. Optimal decision-making with minimal waste: Strategyproof redistribution of VCG payments. In *Proceedings of the 5th International Conference on Autonomous Agents and Multiagent Systems*, pages 882–889, 2006.

[49] J. Chen, S.H.G. Chan, and V.O.K. Li. Multipath routing for video delivery over bandwidth-limited networks. *IEEE J. Select. Areas Commun*, 22(10), 2004.

[50] X. Chen, M. Chamania, A. Jukan., A.C. Drummond, and N.L.S Da Fonseca. On the benefits of multipath routing for distributed data-intensive applications with high bandwidth requirements and multidomain reach. In *Proc. Comm. Netw. & Serv. Research Conf. (CNSR'09)*, Moncton, NB, May 2009.

[51] Y. Chen and C. Shi. Joint pricing and inventory management with strategic customers. In *Proc. ACM Conference Econ. & Comp. (EC)*, Cambridge, MA, June 2017. See `https://papers.ssrn.com/sol3/papers.cfm?abstract_id=2770242` for the full version.

[52] N. Chohan, C. Castillo, M. Spreitzer, M. Steinder, A. Tantawi, and C. Krintz. See spot run: Using spot instances for map reduce workflows. In *Proc. USENIX HotCloud Workshop*, Boston, MA, June 2010.

[53] V. Cholvi, J. Echagüe, and J.-Y. Le Boudec. Worst case burstiness increase due to FIFO multiplexing. *Performance Evaluation*, 49(1):491–506, 2002. Performance 2002.

[54] I Cidon, R Guerin, A Khamisy, and M Sidi. Analysis of a correlated queue in a communication system. *IEEE Trans. Infor. Theo*, 39(2):456–465, 1993.

[55] *Cisco Global Cloud Index: Forecast and Methodology, 2016-2021*, 2018. Accessed on 8/31/19, available at `https://www.cisco.com/c/en/us/solutions/collateral/service-provider/global-cloud-index-gci/white-paper-c11-738085.pdf`.

[56] M. Claypool and K. Claypool. Latency and player actions in online games. *Commun. ACM*, 49(11):40–45, November 2006.

[57] M. Claypool and J. Tanner. The effects of jitter on the peceptual quality of video. In *Proc. ACM MULTIMEDIA'99*, pages 115–118, Orlando, Florida, USA, 1999.

[58] S. S. Craciunas, R. S. Oliver, Martin M. Chmelík, and W. Steiner. Scheduling real-time communication in IEEE 802.1Qbv time sensitive networks. In *Proceedings of the 24th International Conference on Real-Time Networks and Systems*, RTNS '16, page 183–192, New York, NY, USA, 2016. Association for Computing Machinery.

[59] T. Cucinotta, A. Mancina, G.F. Anastasi, G. Lipari, L. Mangeruca, R. Checcozzo, and F. Rusina. A real-time service-oriented architecture for industrial automation. *IEEE Trans. Indus. Informatics*, 5(3), 2009.

[60] B. Deb, S. Bhatnagar, and B. Nath. ReInForM: Reliable information forwarding using multiple paths in sensor networks. In *Proc. IEEE LCN*, Bonn, Germany, October 2003.

[61] R. J. Deneckere and R. P. McAfee. Damaged goods. *Journal of Economics & Management Strategy*, 5(2):149–174, Summer 1996.

[62] D. S. Diamond and L. L. Selwyn. Considerations for computer utility pricing policies. In *Proc. 23rd ACM National Conference*, pages 189–200, Las Vegas, NV, August 1968.

[63] M. Dick, O. Wellnitz, and L. Wolf. Analysis of factors affecting players' performance and perception in multiplayer games. In *Proc. NetGames'05*, Hawthorne, NY, October 2005.

[64] L. Dierks and S. Seuken. Cloud pricing: The spot marlet strikes back. In *Proc. ACM EC Workshop on Economics of Cloud Computing*, Maastricht, the Netherlands, July 2016.

[65] S. Domanal and G. Reddy. An efficient cost optimized scheduling for spot instances in heterogeneous cloud environment. *Future Generation Computer Systems*, 84, 2018.

[66] Michael Downes and Barbara Beeton. *The amsart, amsproc, and amsbook document classes*. American Mathematical Society, August 2004. `http://www.ctan.org/pkg/amslatex`.

[67] E.O. Elliott. Estimates of error rates for codes on burst-noise channels. *Bell System Technical Journal*, 42(5), September 1963.

[68] T. Erl, R. Puttini, and Z. Mahmood. *Cloud Computing: Concepts, Technology & Architecture*. Pearson Education, 2016.

[69] J. Farkas, L. L. Bello, and C. Gunther. Time-sensitive networking standards. *IEEE Communications Standards Magazine*, 2(2):20–21, June 2018.

[70] Simon Fear. *Publication quality tables in LaTeX*, April 2005. `http://www.ctan.org/pkg/booktabs`.

[71] Cristophe Fiorio. *algorithm2e.sty—package for algorithms*, October 2015. `http://www.ctan.org/pkg/algorithm2e`.

[72] A. Ford, C. Raiciu, M. Handley, S. Barre, and J. Iyengar. Architectural guidelines for multipath TCP development. Informational RFC 6182, Internet Engineering Task Force, March 2011.

[73] Deepak Ganesan, Ramesh Govindan, Scott Shenker, and Deborah Estrin. Highly-resilient, energy-efficient multipath routing in wireless sensor networks. *ACM SIGMOBILE Mobile Computing and Communications Review*, 5(4), 2001.

[74] Y Ganjali and A Keshavarzian. Load balancing in ad hoc networks: Single-path routing vs. multi-path routing. In *Proc. IEEE INFOCOM*, Hong Kong, March 2004.

[75] L. Georgiadis, R. Guerin, and A. Parekh. Optimal multiplexing on a single link: delay and buffer requirements. *IEEE Transactions on Information Theory*, 43(5):1518–1535, September 1997.

[76] L. Georgiadis, R. Guérin, V. Peris, and K. N. Sivarajan. Efficient network QoS provisioning based on per node traffic shaping. *IEEE/ACM Transactions on Networking*, 4(4), August 1996.

[77] M. Ghasemi and B. Lubin. Modeling multi-attribute demand for sustainable cloud computing with copulae. In *Proc. IJCAI'15*, Buenos Aires, Argentina, July 2015.

[78] E.N. Gilbert. Capacity of a burst-noise channel. *Bell System Technical Journal*, 39(5), September 1960.

[79] L. Golubchik, J.C.S. Lui, T.F. Tung, A.L.H. Chow, W.-J. Lee, G. Franceschinis, and C. Anglano. Multi-path continuous media streaming: what are the benefits? *Performance Evaluation*, 49(1-–4), September 2002.

[80] Y. Gong, B. He, and A. Zhou. Monetary cost optimizations for mpi-based hpc applications on amazon clouds: Checkpoints and replicated execution. In *Proc. ACM SC'15*, Austin, TX, November 2015.

[81] P. Green, A. Krieger, and Y. Wind. Thirty years of conjoint analysis: Reflections and prospects. *INFORMS Journal on Applied Analytics*, 31, 2001.

[82] R. Guérin, J. C. de Oliveira, and S. Weber. Adoption of bundled services with network externalities and correlated affinities. *ACM Trans. Internet Technol.*, 14(2–3):13:1–13:32, October 2014. DOI:http://dx.doi.org/10.1145/2663493.

[83] R. Gummadi, K. Jung, D. Shah, and R. Sreenivas. Computing the capacity region of a wireless network. In *Proc. IEEE INFOCOM*, Rio de Janeiro, Brazil, April 2009.

[84] Mingyu Guo and Vincent Conitzer. Worst-case optimal redistribution of VCG payments in multi-unit auctions. *Games and Economic Behavior*, 67(1):69–98, 2009.

[85] Weichao Guo, Kang Chen, Yongwei Wu, and Weimin Zheng. Bidding for highly available services with low price in spot instance market. In *Proc. ACM HPDC*, Portland, OR, June 2015.

[86] J.-H. Hahn. Damaged durable goods. *The RAND Journal of Economics*, 37(1):121–133, Spring 2006.

[87] A. Harel. Sharp bounds and simple approximations for the Erlang delay and loss formulas. *Management Science*, 34(8), August 1988.

[88] Jason D. Hartline and Tim Roughgarden. Optimal mechanism design and money burning. In *Proceedings of the Fortieth Annual ACM Symposium on Theory of Computing*, pages 75–84, 2008.

[89] Jason D Hartline and Tim Roughgarden. Simple versus optimal mechanisms. In *Proceedings of the 10th ACM Conference on Electronic Commerce*, pages 225–234. ACM, 2009.

[90] A. Hassidim, D. Raz, M. Segalov, and A. Shaqed. Network utilization: The flow view. In *Proc. IEEE INFOCOM*, Turin, Italy, April 2013.

[91] J. Heinanen and R. Guerin. A single rate three color marker. RFC 2697, Internet Request for Comments (Informational), September 1999.

[92] J. Heinanen and R. Guerin. A two rate three color marker. RFC 2698, Internet Request for Comments (Informational), September 1999.

[93] Carsten Heinz, Brooks Moses, and Jobst Hoffmann. *The Listings Package*, June 2015. `http://www.ctan.org/pkg/listings`.

[94] Health insurance portability and accountability act of 1996. Act of the 104^{th} Congress, `https://aspe.hhs.gov/report/health-insurance-portability-and-accountability-act-1996`. Public Law 104-191.

[95] S. Hong, T. Chantem, and X. S. Hu. Meeting end-to-end deadlines through distributed local deadline assignments. In *Prof. IEEE RTSS*, pages 183–192, Vienna, Austria, November 2011.

[96] M. Howard. Survey: SDN deployed by 78 percent of global service providers by the end of 2018, January 2019. Accessed on 9/11/19, available at `https://technology.ihs.com/610557/survey-sdn-deployed-by-78-percent-of-global-service-providers-by-the-end-of-2018`.

[97] D. Hoy, N. Immorlica, and B. Lucier. On-demand or spot? selling the cloud to risk-averse customers. In *Proc. WINE 2016*, pages 73–86, Montreal, Canada, December 2016.

[98] Xiaoxia Huang and Yuguang Fang. Multiconstrained QoS multipath routing in wireless sensor networks. *Wireless Networks*, 14(4), 2008.

[99] D. Irwin, P. Shenoy, P. Ambati, P. Sharma, and S. Shastri. The price is (not) right: Reflections on pricing for transient cloud servers. In *Proc. ICCCN*, Valencia, Spain, July 2019.

[100] G. W. Irwin, editor. *The engineering of complex real-time computer control systems*, volume 11(3) of *Special Issue of Real-Time Systems*. Kluwer Academic Publishers, 1996.

[101] Y. Ishibashi and S. Tasaka. A comparative survey of synchronization algorithms for continuous media in network environments. In *Proc. IEEE LCN'00*, Tampa, FL, November 2000.

[102] ITU–D SG 2/16 & ITC. *Teletraffic Engineering Handbook*, draft 2001-06-20 edition, 2001. Contact: V. B. Iversen.

[103] I. Jangjaimon and N. Tzeng. Effective cost reduction for elastic clouds under spot instance pricing through adaptive checkpointing. *IEEE Transactions on Computers*, 64, 2015.

[104] U. Javed, M. Suchara, J. he, and J. Rexford. Multipath protocol for delay-sensitive traffic. In *Proc. COMSNETS'09*, Bangalore, India, January 2009.

[105] Y. Jiang, M. Sharad, D. Wentzlaff, D. H.K. Tsang, and C. Joe-Wong. Burstable instances for clouds: Performance modeling, equilibrium analysis, and revenue maximization. In *Proc. IEEE INFOCOM*, Paris, France, April 2019.

[106] R. Johnson. Data centers are under pressure to meet Internet demand, November 2019. Accessed on 11/11/19. Available at `https://cryptodaily.co.uk/2019/11/data-centers-are-under-pressure-to-meet-internet-demand`.

[107] D. Jung, S. Chin, K. Chung, H. Yu, and J. Gil. An efficient checkpointing scheme using price history of spot instances in cloud computing environment. In *Network and Parallel Computing*, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.

[108] I. A. Kash and P. B. Key. Pricing the cloud. *IEEE Internet Computing*, 20(1):36–43, Jan.-Feb. 2016.

[109] M. Kean. Microsoft Azure has the best global WAN, November 2018. Accessed on 8/31/19, available at `https://marckean.com/2018/11/13/microsoft-azure-has-the-best-global-wan/`.

[110] Y. Khalidi. How Microsoft built its fast and reliable global network, March 2017. Accessed on 9/10/19, available at `https://azure.microsoft.com/fr-fr/blog/how-microsoft-builds-its-fast-and-reliable-global-network/`.

[111] A.E. Khandani, J. Abounadi, E. Modiano, and Lizhong Zheng. Reliability and route diversity in wireless networks. *IEEE Trans. Wireless Comm.*, 7(12), December 2008.

[112] M. Khodak, L. Zheng, A. Lan, C. Wong, and M. Chiang. Learning cloud dynamics to optimize spot instance bidding strategies. In *Proc. IEEE INFOCOM*, Honolulu, HI, April 2018.

[113] J. Kiefer. Sequential minimax search for a maximum. *Proc. American Mathematical Society*, 4:502–506, 1953.

[114] C. Kilcioglu and C. Maglaras. Revenue maximization for cloud computing services. *SIGMETRICS Perform. Eval. Rev.*, 43(3):76–76, November 2015. Full version available at `https://www0.gsb.columbia.edu/faculty/cmaglaras/papers/rm-cloud.pdf`.

[115] Turgay Korkmaz and Kamil Saraç. Characterizing link and path reliability in large-scale wireless sensor networks. In *Proc. IEEE WiMob*, Niagara Falls, Canada, October 2010.

[116] D. Kumar, G. Baranwal, Z. Raza, and D. Vidyarthi. A survey on spot pricing in cloud computing. *Journal of Network and Systems Management*, November 2017.

[117] D. Kusic, J. O. Kephart, J. E. Hanson, N. Kandasamy, and G. Jiang. Power and performance management of virtualized computing environments via lookahead control. *Cluster Computing*, 12(1), 2009.

[118] N. Laoutaris and I. Stavrakakis. Intrastream synchronization for continuous media streams: a survey of playout schedulers. *IEEE Network*, 16(3), May 2002.

[119] J.-Y. Le Boudec. A theory of traffic regulators for deterministic networks with application to interleaved regulators. *IEEE/ACM Transactions on Networking*, 26(6):2721–2733, December 2018.

[120] J.-Y. Le Boudec and P. Thiran, editors. *Network Calculus: A Theory of Deterministic Queuing Systems for the Internet*. Springer, 2001.

[121] Sung-Ju Lee and M Gerla. Split multipath routing with maximally disjoint paths in ad hoc networks. In *Proc. IEEE ICC*, Beijing, China, May 2001.

[122] P. Leitner and J. Scheuner. Bursting with possibilities – an empirical study of credit-based bursting cloud instances types. In *Proc. IEEE/ACM 8th International Conference on Utility and Cloud Computing*, Limassol, Cyprus, December 2015.

[123] Ka-Cheong Leung, V.O.K. Li, and Daiqin Yang. An overview of packet reordering in transmission control protocol (TCP): Problems, solutions, and challenges. *IEEE Trans. Parallel Dist. Sys.*, 18(4), April 2007.

[124] C. Li and E. W. Knightly. Coordinated multihop scheduling: A framework for end-to-end services. *IEEE/ACM Transactions on Networking*, 10(6):776–789, December 2002.

[125] W. Li, P. Sv/"ard, J. Tordsson, and E. Elmroth. Cost-optimal cloud service placement under dynamic pricing schemes. In *Proc. IEEE/ACM UCC*, Dresden, Germany, Dec 2013.

[126] W. Li, X. Zhou, K. Li, H. Qi, , and D. Guo. TrafficShaper: Shaping inter-datacenter traffic to reduce the transmission cost. *IEEE/ACM Transactions on Networking*, 26(3):1193–1206, June 2018.

[127] Junjie Liu and Roch Guérin. Multipath and rate stability. Tech. rep., Washington University in St. Louis, 2015. Available at `http://students.cec.wustl.edu/~junjie.liu/multipath_tech_report.pdf`.

[128] J. Loughridge. What AWS customer should know about the AWS global backbone, April 2019. Accessed on 8/31/19, available at `https://konekti.us/2019/04/29/aws-backbone.html`.

[129] L. Luo, H. Yu, K. T. Foerster, M. Noormohammadpour, and S. Schmid. Inter-datacenter bulk transfers: Trends and challenges. *IEEE Network*, 34(5):240–246, July 2020.

[130] N. C. Luong, P. Wang, D. Niyato, Y. Wen, and Z. Han. Resource management in cloud networking using economic analysis and pricing models: A survey. *IEEE Communications Surveys Tutorials*, 19(2):954–1001, Second Quarter 2017.

[131] D.J.C. MacKay. Fountain codes. *IEE Proceedings – Communications*, 152(6), December 2005.

[132] R. Mahfouzi, A. Aminifar, S. Samii, A. Rezine, P. Eles, and Z. Peng. Stability-aware integrated routing and scheduling for control applications in Ethernet networks. In *Proc. Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 682–687, Dresden, Germany, March 2018.

[133] M Maleki, K Dantu, and M Pedram. Lifetime prediction routing in mobile ad hoc networks. *Proc. IEEE WCNC*, March 2003.

[134] M. Mao and M. Humphrey. A performance study on the VM startup time in the cloud. In *Proc. IEEE CLOUD Conference*, Honolulu, HI, June 2012.

[135] M K Marina and S R Das. On-demand multipath distance vector routing in ad hoc networks. In *Proc. IEEE ICNP*, Riverside, CA, November 2001.

[136] D. Marinca, P. Minet, and L. George. Analysis of deadline assignment methods in distributed real-time systems. *Computer Communications*, 27(15):1412 – 1423, September 2004.

[137] M. Mattess, C. Vecchiola, and R. Buyya. Managing peak loads by leasing cloud infrastructure services from a spot market. In *Proc. IEEE 12th HPCC*, pages 145–160, Atlanta, GA, September 2010.

[138] M. Mazzucco and M. Dumas. Achieving performance and availability guarantees with spot instances. In *Proc. IEEE HPCC*, Banff, Canada, September 2011.

[139] N Meghanathan. Stability-energy consumption tradeoff among mobile ad hoc network routing protocols. In *Proc. IEEE ICWMC*, Guadeloupe, French Carribean, March 2007.

[140] H. Mendelson and S. Whang. Optimal incentive-compatible priority pricing for the M/M/1 queue. *Operations Research*, 38(5):870–883, September-October 1990.

[141] Microsoft global network, June 2019. Accessed on 8/31/19, available at `https://docs.microsoft.com/en-us/azure/networking/microsoft-global-network`.

[142] D. K. Mishra, P. Vankar, and M. P. Tahiliani. TCP evaluation suite for ns-3. In *Proc. Workshop on ns-3*, WNS3 '16, page 25–32, Seattle, WA, USA, June 2016. Association for Computing Machinery.

[143] E. Mohammadpour, E. Stai, M. Mohiuddin, and J-Y. Le Boudec. Latency and backlog bounds in time-sensitive networking with credit based shapers and asynchronous traffic shaping. In *Proc. 30th International Teletraffic Congress (ITC 30)*, Vienna, Austria, September 2018.

[144] E. Mooi, M. Sarstedt, and I. Mooi-Reci. *Market Research – The Process, Data, and Methods Using Stata*. Springer, Singapore, 2018.

[145] Hervé Moulin. Almost budget-balanced VCG mechanisms to assign multiple objects. *Journal of Economic Theory*, 144(1):96–119, 2009.

[146] Asis Nasipuri and Samir R Das. On-demand multipath routing for mobile ad hoc networks. In *Proc. IEEE ICCCN*, Boston, MA, October 1999.

[147] R.B. Nelsen. *An introduction to copulas*. Springer, second edition, 2009.

[148] T. Nguyen and A. Lebre. Virtual machine boot time model. In *Proc. Euromicro International Conference on Parallel, Distributed and Network-based Processing*, St. Peterbourg, Russia, May 2017.

[149] Scaling AWS for Black Friday: Lessons from MovieTickets.com Star Wars release. Case study, Onica, November 2018. Retrieved March 4, 2019 at `https://www.onica.com/blog/scaling-aws-black-friday-lessons-movietickets-com-star-wars-release/`.

[150] A. K. Parekh and R. G. Gallager. A generalized processor sharing approach to flow control in integrated services networks: the single-node case. *IEEE/ACM Transactions on Networking*, 1(3):344–357, June 1993.

[151] A. K. Parekh and R. G. Gallager. A generalized processor sharing approach to flow control in integrated services networks: the multiple node case. *IEEE/ACM Transactions on Networking*, 2(2):137–150, April 1994.

[152] R. Pary. New Amazon EC2 Spot pricing model: Simplified purchasing without bidding and fewer interruptions, November 2018. Retrieved March 4, 2019 at `https://aws.amazon.com/blogs/compute/new-amazon-ec2-spot-pricing/`.

[153] V. Paxson. End-to-end routing behavior in the Internet. In *Proc. ACM SIGCOMM*, Stanford, CA, August 1996.

[154] M. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, Inc., 1994.

[155] M. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, Inc., 1994.

[156] M. Rabin. Efficient dispersal of information for security, load-balancing, and fault tolerance. *Journal of the ACM*, 32(2), April 1989.

[157] Marjan Radi, Behnam Dezfouli, Kamalrulnizam Abu Bakar, and Malrey Lee. Multipath routing in wireless sensor networks: Survey and research challenges. *Sensors*, 12(1), January 2012.

[158] Krishna Ramachandran, Irfan Sheriff, Elizabeth Belding, and Kevin Almeroth. Routing stability in static wireless mesh networks. In *Proc. PAM'07*, Louvain-la-Neuve, Belgium, April 2007.

[159] K. Razavi, G. V. D. Kolk, and T. Kielmann. Prebaked μVMs: Scalable, instant VM startup for IaaS clouds. In *Proc. IEEE 35th International Conference on Distributed Computing Systems*, Columbus, OH, June 2015.

[160] D. Richman. Amazon Web Services' secret weapon: Its custom-made hardware and network, January 2017. Accessed on 9/11/19, available at `https://www.geekwire.com/2017/amazon-web-services-secret-weapon-custom-made-hardware-network/`.

[161] A. Roy, H. Zeng, J. Bagga, G. Porter, and A. C. Snoeren. Inside the social network's (datacenter) network. In *Proc. ACM SIGCOMM*, SIGCOMM '15, page 123–137, London, United Kingdom, August 2015. Association for Computing Machinery.

[162] J. B. Schmitt. On the allocation of network service curves for bandwidth/delay-decoupled scheduling disciplines. In *Proc. IEEE GLOBECOM*, volume 2, pages 1544–1548 vol.2, Taipei, Taiwan, November 2002.

[163] N. Kr. Sharma, C. Zhao, M. Liu, P. G. Kannan, C. Kim, A. Krishnamurthy, and A. Sivaraman. Programmable calendar queues for high-speed packet scheduling. In *Proc. USENIX NSDI*, pages 685–699, Santa Clara, CA, February 2020. USENIX Association.

[164] P. Sharma, D. Irwin, and P. Shenoy. How not to bid the cloud. In *Proc. USENIX HotCloud Workshop*, Denver, CO, June 2016.

[165] D. A. Shepherd and A. Zacharakis. *Conjoint Analysis: A Window of Opportunity for Entrepreneurship Research*, chapter 6, pages 149–183. 2018.

[166] A. Shieh, S. Kandula, A. Greenberg, C. Kim, and B. Saha. Sharing the data center network. In *Proc. USENIX NSDI*, Boston, MA, March 2011.

[167] J. Shneidman, C. Ng, D. C. Parkes, A. AuYoung, A. C. Snoeren, A. Vahdat, and B. N. Chun. Why markets could (but don't currently) solve resource allocation problems in systems. In *Proc. USENIX HotOS Workshop*, Santa Fe, NM, 2005.

[168] A. Sivaraman, S. Subramanian, M. Alizadeh, S. Chole, S.-T. Chuang, A. Agrawal, H. Balakrishnan, T. Edsall, S. Katti, and N. McKeown. Programmable packet scheduling at line rate. In *Proc. ACM SIGCOMM*, SIGCOMM '16, page 44–57, Florianopolis, Brazil, August 2016. Association for Computing Machinery.

[169] B. T. Sloss. Expanding our global infrastructure with new regions and subsea cables, January 2018. Accessed on 8/31/19, available at `https://www.blog.google/products/google-cloud/expanding-our-global-infrastructure-new-regions-and-subsea-cables/`.

[170] Axel Sommerfeldt. *The subcaption package*, April 2013. `http://www.ctan.org/pkg/subcaption`.

[171] J. Song and R. Guérin. Pricing and bidding strategies for cloud computing spot instances. Technical report, Washington University in St. Louis, January 2017.

[172] J. Song, R. Guérin, and H. Sariowan. Minimizing network bandwidth under latency constraints: The multiple nodes case, 2021. Work in progress.

[173] J. Song, R. Guérin, and H. Sariowan. Minimizing network bandwidth under latency constraints: The single node case. In *Proc. ITC 33 - Networked Systems and Services*, Avignon, France, Sep 2021. Extended version available at `https://arxiv.org/abs/2104.02222`.

[174] Jiayi Song and Roch Guérin. Pricing (and bidding) strategies for delay differentiated cloud services. *ACM Trans. Econ. Comput.*, 8(2), may 2020.

[175] Y. Song, M. Zafer, and K. Lee. Optimal bidding in spot instance market. In *Proc. IEEE INFOCOM*, Orlando, FL, May 2012.

[176] J. Specht and S. Samii. Urgency-based scheduler for time-sensitive switched Ethernet networks. In *Proc. Euromicro Conference on Real-Time Systems (ECRTS)*, pages 75–85, Toulouse, France, July 2016.

[177] I. E. Sutherland. A futures market in computer time. *Communications of the ACM*, 11(6):449–451, 1968.

[178] Nicola L. C. Talbot. *User Manual for glossaries.sty v4.44*, December 2019. `http://www.ctan.org/pkg/glossaries`.

[179] S. Tang, J. Yuan, and X. Li. Towards optimal bidding strategy for amazon ec2 cloud spot instance. In *Proc. IEEE CLOUD*, Honolulu, HI, June 2012.

[180] *Public Cloud Performance Benchmark Report*, 2018. Accessed on 8/31/19, available at `https://www.thousandeyes.com/resources/2018-public-cloud-performance-benchmark-report`.

[181] *Cloud Performance Benchmark – 2019-2020 Edition*, 2019. Accessed on 4/02/21, available at http://presse.hbi.de/pub/ThousandEyes/Cloud_Performance_Benchmark_Report/ThousandEyesCloudPerformanceBenchmarkReport.pdf.

[182] P. K. Trivedi and D. M. Zimmer. Copula modeling: An introduction for practitioners. *Foundations and Trends in Econometrics*, 1(1):1–111, 2005.

[183] J Tsai and T Moors. A review of multipath routing protocols: From wireless ad hoc to mesh networks. *... researcher workshop on Wireless Multihop Networking*, 2006.

[184] J. Tsai and T. Moors. A review of multipath routing protocols: From wireless ad hoc to mesh networks. In *Proc. ACoRN Early Career Researcher Workshop on Wireless Multihop Networking*, July 2006.

[185] A. Tsirigos and Z. Haas. Analysis of multipath routing, part 1: The effect on the packet delivery ratio. *IEEE Trans. Wireless Comm.*, 3(1), January 2004.

[186] J. Turner. SDN is the unsung hero of the cloud services evolution, December 2016. Network World, retrieved on 9/10/19, available at https://www.networkworld.com/article/3152415/sdn-is-the-unsung-hero-of-the-cloud-services-evolution.html.

[187] UK TEX Users Group. UK list of TEX frequently asked questions. https://texfaq.org, 2019.

[188] A. Vahdat. A look inside Google's data center networks, June 2015. Accessed on 9/11/19, available at https://cloudplatform.googleblog.com/2015/06/A-Look-Inside-Googles-Data-Center-Networks.html.

[189] A. Van Bemten and W. Kellerer. Network calculus: A comprehensive guide. Technical Report 201603, Technical University Munich, October 2016.

[190] E. Vergetis, R. Guérin, and S. Sarkar. Realizing the benefits of user-level channel diversity. *ACM Computer Communication Review*, 35(5), October 2005.

[191] Boris Veytsman, Bern Schandl, Lee Netherton, and C. V. Radhakrishnan. *A package to create a nomenclature*, September 2005. http://www.ctan.org/pkg/nomencl.

[192] J. Wanderer and A. Vahdat. Google Cloud using P4Runtime to build smart networks, April 2018. Accessed on 9/10/19, available at https://cloud.google.com/blog/products/gcp/google-cloud-using-p4runtime-to-build-smart-networks.

[193] C. Wang and H. Kim. Touchdown on the Cloud: The impact of the Super Bowl on Cloud. *arXiv e-prints*, page arXiv:1902.07363, February 2019.

[194] Lei Wang, Lianfang Zhang, Yantai Shu, and Miao Dong. Multipath source routing in wireless ad hoc networks. In *Proc. 2000 Canadian Conf. Elec. & Comp. Eng.*, Halifax, NS, Canada, May 2000.

[195] B. Xavier, T. Ferreto, and L. Jersak. Time provisioning evaluation of KVM, Docker and Unikernels in a cloud platform. In *Proc. 16th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid)*, Cartagena, Columbia, May 2016.

[196] H. Xu and B. Li. Dynamic cloud pricing for revenue maximization. *IEEE Transactions on Cloud Computing*, 1(2):158–171, July 2013.

[197] Yaling Yang, Jun Wang, and Robin Kravets. Designing routing metrics for mesh networks. In *Proc. IEEE WiMesh*, Santa Clara, CA, September 2005.

[198] S. Yi, A. Andrzejak, and D. Kondo. Monetary cost-aware checkpointing and migration on amazon cloud spot instances. *IEEE Transactions on Services Computing*, 5, 2012.

[199] M. Zafer, Y. Song, and K. Lee. Optimal bids for spot VMs in a cloud for deadline constrained jobs. In *Proc. IEEE CLOUD*, Honolulu, HI, June 2012.

[200] E. Zhang and L. Xu. Capacity and token rate estimation for networks with token bucket shapers. *Computer Networks*, 88:1–11, 2015.

[201] H. Zhang. Providing end-to-end performance guarantees using non-work-conserving disciplines. *Computer Communications*, 18(10):769 – 781, 1995.

[202] J. Zhang, L. Chen, T. Wang, and X. Wang. Analysis of TSN for industrial automation based on network calculus. In *Proc. 24th International Conference on Emerging Technologies and Factory Automation*, Zaragoza, Spain, September 2019.

[203] H. Zhao, M. Pan, X. Liu, X. Li, and Y. Fang. Optimal resource rental planning for elastic applications in cloud market. In *Proc. IEEE IPDPS*, Shanghai, China, May 2012.

[204] X. Zhao, V. Vusirikala, B. Koley, V. Kamalov, and T. Hofmeister. The prospect of inter-data-center optical networks. *IEEE Communications Magazine*, 51(9):32–38, 2013.

[205] L. Zheng, C. Joe-Wong, C. W. Tan, M. Chiang, and X. Wang. How to bid the cloud. In *Proc. 2015 ACM SIGCOMM*, London, UK, AUGUST 2015.

[206] A. Zhou, B. He, and C. Liu. Monetary cost optimizations for hosting workflow-as-a-service in iaas clouds. *IEEE Transactions on Cloud Computing*, 4, 2016.

[207] B. Zhou, I. Howenstine, S. Limprapaipong, and L. Cheng. A survey on network calculus tools for network infrastructure in real-time systems. *IEEE Access*, 8:223588–223605, 2020.

[208] T. Zhu, M. A. Kozuch, and M. Harchol-Balter. *WorkloadCompactor: Reducing Datacenter Cost While Providing Tail Latency SLO Guarantees*, page 598–610. Association for Computing Machinery, Santa Clara, CA, September 2017.

Appendix A

Pricing Strategies for Delay Differentiated Cloud Services

A.1 Glossary

Notation used in the chapter's main body and listed approximately in order of appearance in the chapter.

Notation	**Definition**
$\boldsymbol{p} = (p_1, p_2, ..., p_n)$	spot prices
$\boldsymbol{\pi} = (\pi_1, \pi_2, ...\pi_n)$	spot price distribution
(t, v, κ)	Job profile (length, value, delay sensitivity)
$f(t)$	density function of job lengths
$q(v, \kappa)$	joint density function of job value and delay sensitivity

$\Gamma_{\boldsymbol{p},\boldsymbol{\pi}}(t, v, \kappa)$	bidding strategy for job (t, v, κ) given $\boldsymbol{p}$ and $\boldsymbol{\pi}$ (simplified as $\Gamma(t, v, \kappa)$ or Γ when unambiguous)
$< \boldsymbol{p} >_{\boldsymbol{r}}$	realization of spot prices under distribution $\boldsymbol{p}$
$u_r(t, v, \kappa, \Gamma, < \boldsymbol{p} >_{\boldsymbol{r}})$	utility of job with profile (t, v, κ) under strategy Γ and spot price realization $< \boldsymbol{p} >_{\boldsymbol{r}}$
$p_r(t, v, \kappa, \Gamma, < \boldsymbol{p} >_{\boldsymbol{r}})$	total price paid by job with profile (t, v, κ) under strategy Γ and spot price realization $< \boldsymbol{p} >_{\boldsymbol{r}}$
$d_r(t, v, \kappa, \Gamma, < \boldsymbol{p} >_{\boldsymbol{r}})$	delay penalty incurred by job with profile (t, v, κ) under strategy Γ and spot price realization $< \boldsymbol{p} >_{\boldsymbol{r}}$
$t_r(t, v, \kappa, \Gamma, < \boldsymbol{p} >_{\boldsymbol{r}})$	execution delay experienced by job with profile (t, v, κ) under strategy Γ and spot price realization $< \boldsymbol{p} >_{\boldsymbol{r}}$
$U_{\boldsymbol{p},\boldsymbol{\pi}}(t, v, \kappa, \Gamma)$	expected utility for job with profile (t, v, κ) under strategy Γ
$P_{\boldsymbol{p},\boldsymbol{\pi}}(t, v, \kappa, \Gamma)$	expected execution cost for job with profile (t, v, κ) under strategy Γ
$T_{\boldsymbol{p},\boldsymbol{\pi}}(t, v, \kappa, \Gamma)$	expected execution delay for job with profile (t, v, κ) under strategy Γ (denoted simply as $T(t)$ when there is no possibility of ambiguity)
$\Gamma^*_{\boldsymbol{p},\boldsymbol{\pi}}(t, v, \kappa)$	optimal bidding strategy for job (t, v, κ) given $\boldsymbol{p}$ and $\boldsymbol{\pi}$
$\Gamma^*(\boldsymbol{p}, \boldsymbol{\pi})$	optimal bidding strategies across jobs, given spot prices and spot prices distribution $(\boldsymbol{p}, \boldsymbol{\pi})$
$R_{f,q}(\Gamma^*(\boldsymbol{p}, \boldsymbol{\pi}))$	expected per job revenue under spot prices and spot prices distribution $(\boldsymbol{p}, \boldsymbol{\pi})$ users with profiles distributed according to $f(\cdot)$ and $q(\cdot, \cdot)$, and using strategy $\Gamma^*(\boldsymbol{p}, \boldsymbol{\pi})$
$(\boldsymbol{p^*}, \boldsymbol{\pi^*})$	optimal (revenue maximizing) spot prices and spot prices distribution given users with profiles distributed according to $f(\cdot)$ and $q(\cdot, \cdot)$, and using strategy $\Gamma^*(\boldsymbol{p^*}, \boldsymbol{\pi^*})$

$C_{\boldsymbol{p},\boldsymbol{\pi}}(t, v, \kappa, \Gamma)$	total expected cost (price plus delay penalty) for job with profile (t, v, κ) under strategy Γ (denoted as $C(t, v, \kappa, b)$ when focusing on a given job and pricing strategy and a fixed bidding strategy with a bid value of b, or even as $C(\Gamma)$ when there is no possibility of ambiguity)
$b^*(v, \kappa)$	optimal (fixed) bid for job (v, κ) under linear delay penalty and without the possibility of early termination
$b^*(\kappa)$	optimal (fixed) bid for job with delay sensitivity κ and a value high enough to justify bidding under linear delay penalty and without the possibility of early termination
b	bid value
$\alpha(b)$	probability of winning a bid when bidding at b under given spot prices values and distribution
$\overline{b}$	expected payment per unit of execution time for a job bidding at b under given spot prices and distribution
$\widetilde{\kappa}_i$	threshold in a job's delay sensitivity associated with a change in the value of a job's optimal bid to p_i
$\Gamma(\infty)$	bidding strategy that involves bidding at p_1 until a success in a system with two prices, $p_1 < p_2$
$\Gamma(l), 0 < l < \infty$	bidding strategy that involves bidding at p_1 for up to l slots until a success, before switching to bidding at p_2 in a system with two prices, $p_1 < p_2$
$\Gamma(0)$	bidding strategy that involves bidding at p_2 in a system with two prices, $p_1 < p_2$
r	fraction of jobs with value v_1 in a binary system
s	fraction of jobs with delay sensitivity κ_1 in a binary system
q_{ij}	fraction of jobs with value v_i and delay sensitivity κ_j in a binary system

p^*	optimal price of the one-price strategy in a binary system
ρ	correlation coefficient between value, v, and delay sensitivity, κ, in a job profile
ρ^*	correlation threshold above which a two-price policy is optimal
$\Delta^*_{21}(v_2, \kappa_2)$	difference in expected cost between bidding at p^*_2 and bidding at p^*_1 for (v_2, κ_2) jobs in a binary system
θ	delay threshold when specifying piecewise-linear convex or concave delay sensitivity function
$D_1(t)$	piecewise-linear convex delay sensitivity function
$D_2(t)$	piecewise-linear convex delay sensitivity function
$\widehat{t}$	a job residual service
$\widehat{T}$	a job current cumulative execution delay
$v_{\min}$ and $v_{\max}$	minimum and maximum job values when job values can span a continuous range
$\kappa_{\min}$ and $\kappa_{\max}$	minimum and maximum delay sensisitivity values when delay sensitivities can span a continuous range

A.2 The Need for Heterogeneity in Delay Sensitivity

The purpose of this section is simply to establish that without heterogeneity along the dimensions of *both* job value and delay sensitivity, a spot service defaults to an on-demand (single price service). We first show this for a scenario where all jobs have the same value and different sensitivities to delay, and then proceed to show that it also holds when jobs have different values and a common delay sensitivity.

A.2.1 Homogeneous Value, Heterogeneous Delay Sensitivities

Proposition 22. *When jobs all have a common value v and arbitrary delay sensitivities, a one-price strategy with a price of v dominates all multi-price strategies.*

Proof. Consider a one-price strategy with price v, *i.e.,* the job value common to all users. All jobs that had a non-negative utility under the optimal multi-price strategy, also have a non-negative utility under this strategy, and therefore purchase the service at the price of v. This ensures that this one-price strategy generates a revenue at least as large as that of the optimal multi-price strategy. □

A.2.2 Heterogeneous Values, Homogeneous Delay Sensitivity

Proposition 23. *When jobs have a common delay sensitivity κ and arbitrary values, a one-price strategy maximizes revenue.*

Proof. The proof is a direct consequence of Proposition 2 that established that given a pricing strategy, the choice of a specific bidding value was independent of a job's value. □

A.3 Lemmas and Proofs leading to Propositions 1 and 2

The first lemma establishes that the set of prices to be considered by the bidding strategy can be reduced to the set of spot prices advertised by the pricing strategy.

Lemma 24. *Bidding at a price $b \in [p_i, p_{i+1})$, where $1 \leq i \leq n$, generates the same expected cost and the same winning probability as bidding at p_i.*

Proof. A bid $b \in [p_i, p_{i+1})$ wins if and only if the current spot price $s \leq p_i$, so that all bidding prices in $[p_i, p_{i+1})$ have the same winning probability. Furthermore, a winning bid is charged the current spot price s, independent of the bid value. Hence any winning bid in $[p_i, p_{i+1})$ is charged the same $s \leq p_i$. Consequently, all bidding prices in $[p_i, p_{i+1})$ experience the same expected cost. □

Towards establishing the next results, we formulate the bidding process for job $(1, v, \kappa)$ as a finite Markov Decision Process (MDP). Specifically, the MDP consists of 3 states: an initial state (S_0), a bid state (S_1) and a terminal state (S_2). In S_0, customers choose to adopt the service (bid) or to exit. A job gets a reward of v and goes to S_1 if it decides to adopt. Otherwise, it gets a reward of 0 and goes to S_2. In S_1, the customer chooses a bid $b \in \{p_1, \ldots, p_n\}$, which gives a winning probability of $\sum_{p_l \leq b} \pi_l$. If the bid wins with a spot price of $p_i \leq b$, the job gets a reward of $-p_i$ and moves to S_2; otherwise, it gets a reward of $-\kappa$ and stays in S_1. Formally:

Decision epochs:

$$T = \{1, 2, \ldots, N\}, \qquad N \leq \infty.$$

States:

$$S = \{S_0, S_1, S_2\}.$$

Actions (A_i denotes the action set for S_i):

$$A_0 = \{adopt, exit\},$$

$$A_1 = \{p_1, p_2, \ldots, p_n\},\ \textit{i.e.,}\ \text{the set of possible bidding prices.}$$

Expected rewards:

$$r(adopt, S_0) = v; \qquad r(exit, S_0) = 0;$$

$$r(p_i, S_1) = -\sum_{h \leq i} p_h \pi_h - (1 - \sum_{l \leq i} \pi_l)\kappa.$$

Transition probabilities (all omitted transitions have a probability of 0):

$$Prob(S_1|S_0, adopt) = 1; \qquad Prob(S_2|S_0, exit) = 1;$$

$$Prob(S_2|S_1, p_i) = \sum_{h \leq i} \pi_h, \qquad Prob(S_1|S_1, p_i) = 1 - \sum_{h \leq i} \pi_h.$$

Proposition 25, which establishes the existence of an optimal bidding strategy for jobs of length 1, is then an immediate consequence of Proposition 4.4.3.*a* of [154], which we restate below for the reader's convenience.

Proposition 4.4.3.a [154]: Assume the set of possible system states S is finite or countable, and that for every $s \in S$, the set of its allowable actions is finite. Then there exists a deterministic Markovian policy which is optimal.

Proposition 25. *An optimal fixed bidding strategy exists for jobs of type* $(1, v, \kappa)$.

As the MDP is finite, Proposition 4.4.3.a ensures the existence of an optimal policy, and its deterministic Markovian nature implies the invariance of the optimal bidding price.

A.4 Proofs of Propositions 1 and 2

For clarity, we restate the propositions before introducing their proofs.

Proposition 1. *Given a pricing strategy* $(\boldsymbol{p}, \boldsymbol{\pi})$ *and a linear penalty function for a job's execution delay, a strategy that bids at* $b^*(v, \kappa)$ *in every slot is an optimal bidding strategy for job* (t, v, κ)*, where* $b^*(v, \kappa)$ *is the optimal bid for job* $(1, v, \kappa)$*.*

Proof. We first formulate job (t, v, κ)'s MDP. From its deterministic Markovian nature, we know that the optimal bidding strategy bids at a fixed price for a given job's residual size, *i.e.,* the amount of work left to complete the job. We then show the Proposition through an induction on the job length.

The MDP for job (t, v, κ) is of the following form:

MDP characterization: We characterize job (t, v, κ)'s bidding process as an MDP with $(t + 2)$ states: initial state (S_0), bid states $(S_1, S_2, ..., S_t)$, and terminal state (S_{t+1}). In S_0, the customer chooses to adopt the service or to exit. The job gets a rewards of vt and goes to state S_1 if it decides to adopt. Otherwise, it gets a reward of 0 and goes to state S_{t+1}. In $S_i, i = 1, 2, \ldots, t$, the customer chooses a bid in $\{p_1, ..., p_n\}$. If the bid wins with a spot price of p_j, then the job gets a reward of $-p_j$ and moves to S_{i+1}; otherwise, the job gets a reward of $-\kappa$ and stays at S_i. As the optimal bidding strategy in a given state bids systematically at the same price regardless of the time epoch, we eliminate the time epoch in our notation. Formally:

Decision epochs:

$$T = \{1, 2, ..., N\}, \qquad N \leq \infty.$$

States:

$$S = \{S_0, S_1, \ldots, S_t, S_{t+1}\}.$$

Actions (A_i denotes the action set for S_i):

$$A_0 = \{adopt, exit\},$$

$$A_i = \{p_1, p_2, \ldots, p_n\}, \text{ for } i \in \{1, \ldots, t\}.$$

Expected rewards:

$$r(adopt, S_0) = vt, \qquad r(exit, S_0) = 0;$$

$$r(p_j, S_i) = -\sum_{h \le j} p_h \pi_h - \left(1 - \sum_{l \le j} \pi_l\right) \kappa, \text{ for } i \in \{1, \ldots, t\}.$$

Of note from the above expression is the fact that $r(p_j, S_i)$ is *independent* of $S_i, i \in \{1, \ldots, t\}$

Transition probabilities (all omitted transitions have a probability of 0):

$$Prob(S_1|S_0, adopt) = 1; \qquad Prob(S_{t+1}|S_0, exit) = 1;$$

$$Prob(S_{i+1}|S_i, p_j) = \sum_{h \le j} \pi_h, \text{ for } i \in \{1, \ldots, t\};$$

$$Prob(S_i|S_i, p_j) = 1 - \sum_{h \le j} \pi_h, \text{ for } i \in \{1, \ldots, t\}.$$

Induction: Denote the optimal bid for job $(1, t, \kappa)$ as $b^*(v, \kappa)$. We show that bidding at $b^*(v, \kappa)$ in every slot gives job (t, v, κ)'s optimal bidding strategy.

When $t = 1$, Proposition 25 directly gives the results. Assume that bidding at $b^*(v, \kappa)$ in every slot is optimal for jobs $(l, v, \kappa), 1 \le l \le t - 1$. We show that bidding at $b^*(v, \kappa)$ is also optimal for jobs (t, v, κ).

First note that when a job (t, v, κ) has a residual size of 1, *i.e.,* is in state S_t, it will bid at $b^*(v, \kappa)$. Given its Markovian nature, the optimal substrategy at S_t is independent of the job's bidding history, and simply seeks to minimize the expected cost of winning a single bid to complete the job. Hence, it is identical to the optimal bidding strategy for a job $(1, v, \kappa)$.

Next we show that the optimal strategy for job (t, v, κ) also bids at $b^*(v, \kappa)$ before entering state S_t. As the optimal substrategy in state S_t has a fixed expected cost for successfully completing the next bid regardless of the bidding history, the optimal strategy prior to reaching state S_t solely seeks to minimize the expected cost of winning $(t-1)$ slots. Hence, it behaves as the optimal bidding strategy for jobs $(t-1, v, \kappa)$, which from our induction assumption keeps bidding at $b^*(v, \kappa)$.

When combined with the fact that the optimal (sub)strategy at S_t also bids at $b^*(v, \kappa)$, this establishes the induction and completes the proof. □

Recalling next Proposition 2.

Proposition 2. *A job's optimal fixed bid $b^*(\kappa)$ is independent of v and t, and non-decreasing in κ. Specifically, a job with $\kappa \in (\widetilde{\kappa}_{i-1}, \widetilde{\kappa}_i]$ will bid at p_i if it bids, where $\widetilde{\kappa}_i = \sum_{j \le i}(p_{i+1} - p_j)\pi_j$ for $i \ge 1$, and $\widetilde{\kappa}_0 = -\epsilon < 0$.*

Proof. As mentioned before introducing the proposition, the independence of the optimal bidding price from t is a direct consequence of Proposition 1. The fact that it is non-decreasing in κ is because $\widetilde{\kappa}_i$ increases with i, as we establish next. Recall that

$$\widetilde{\kappa}_i = \sum_{0 < j \le i} (p_{i+1} - p_j)\pi_j.$$

Hence we have

$$\begin{aligned}\widetilde{\kappa}_{i+1} - \widetilde{\kappa}_i &= \sum_{j\le i}(p_{i+2} - p_j)\pi_j + (p_{i+2} - p_{i+1})\pi_{i+1} - \sum_{j\le i}(p_{i+1} - p_j)\pi_j \\ &= \sum_{j\le i}(p_{i+2} - p_{i+1})\pi_j + (p_{i+2} - p_{i+1})\pi_{i+1} \\ &= \sum_{j\le i+1}(p_{i+2} - p_{i+1})\pi_j > 0\end{aligned}$$

Next we turn to showing that a customer with $\kappa \in (\widetilde{\kappa}_{i-1}, \widetilde{\kappa}_i]$ bids at p_i, which also establishes that $b^*(\kappa)$ is independent of v.

Basic algebraic manipulations yield

$$U(t, v, \kappa, p_{i+1}) - U(t, v, \kappa, p_i) = \frac{\kappa - \sum_{j\le i}(p_{i+1} - p_j)\pi_j}{\alpha(p_i)\alpha(p_{i+1})},$$

which implies that $U(t, v, \kappa, p_{i+1}) - U(t, v, \kappa, p_i) \ge 0$ iff $\kappa \ge \widetilde{\kappa}_i$.

As $\widetilde{\kappa}_i$ increases with i, $U(t, v, \kappa, p_i) \ge U(t, v, \kappa, p_{i-1})$ implies $U(t, v, \kappa, p_i) \ge U(t, v, \kappa, p_j), \forall j < i$, and $U(t, v, \kappa, p_{i+1}) \le U(t, v, \kappa, p_i)$ implies $U(t, v, \kappa, p_{i+1}) \le U(t, v, \kappa, p_j), \forall j > i$. Therefore, p_i is the optimal bid for job (t, v, κ) if only it satisfies $U(t, v, \kappa, p_{i+1}) \le U(t, v, \kappa, p_i)$ and $U(t, v, \kappa, p_{i-1}) \le U(t, v, \kappa, p_i)$, which is equivalent to $\kappa \in (\widetilde{\kappa}_{i-1}, \widetilde{\kappa}_i]$.

□

A.5 Proof of Lemma 4

Recall that Lemma 4 states

Lemma 4. *For binary profiles with $0 \leq \kappa_1 < \kappa_2$, the optimal pricing system needs at most two prices.*

Instead of proving Lemma 4, we prove a more general result, namely, Proposition 27, whose proof calls first for a simple lemma.

Lemma 26. *Under the assumptions of Sections 2.4 and 2.5, in a system with n distinct delay sensitivities, customers will bid using at most n prices.*

Proof. The lemma is a direct consequence of the proof of Proposition 2 that established that a job's bidding choice is independent of its value and, therefore, only affected by its delay sensitivity. Hence, n distinct κ values map to at most n different prices. □

Proposition 27. *Under the assumptions of Sections 2.4 and 2.5, the optimal (revenue maximizing) pricing strategy in a system with n distinct delay sensitivities needs at most n prices.*

Proof. We show that for every $(n+1)$-price service, there exists an n-price service that generates no less expected revenue.

Denote the prices in the $(n+1)$-price service as $p_1 < p_2 < ... < p_{n+1}$, and the corresponding probabilities as $\pi_1, \pi_2, ..., \pi_{n+1}$ respectively, where $\pi_i > 0, \forall i$.

From Lemma 26 we know that at most n prices have a positive adoption in this system. Assume $p_h, 1 \leq h \leq n+1$, has no adoption.

If $h \neq n+1$, consider a new pricing system with n prices $p_1 < \ldots < p_{h-1} < \frac{\pi_h p_h + \pi_{h+1} p_{h+1}}{\pi_h + \pi_{h+1}} < p_{h+2} < ...p_{n+1}$, and with probabilities $\pi_1, \ldots, \pi_{h-1}, \pi_h + \pi_{h+1}, \pi_{h+2}, ...\pi_{n+1}$, respectively. As neither the utility nor the average payment for bidding at p_i $(i \neq h)$ change, all jobs continue

to bid at the same prices. Hence this new n-price service generates the same revenue as the earlier $(n+1)$-price service.

If $h = n+1$, consider a new pricing system with prices $p_1 < p_2 < ... < p_n$, and with probability $\frac{\pi_1}{1-\pi_{n+1}}, \frac{\pi_2}{1-\pi_{n+1}}, ..., \frac{\pi_n}{1-\pi_{n+1}}$, respectively. In this case again the average payment for a job bidding at any price p_i remains the same, while the expected utility for bidding at p_i increases as the expected job completion time decreases. As a result, all jobs that adopted (decided to bid) under the previous price system, still do under the new price system. Furthermore, because the κ values for which $U(t,\kappa,v,p_i) = U(t,\kappa,v,p_{i-1}), 1 \leq i \leq n$, *i.e.,* $\widetilde{\kappa}_i$, remain the same, all jobs continue to bid at the same price as that in the original $(n+1)$-price system. Hence, the n-price service generates no less revenue than the $(n+1)$-price service system. □

A.6 Proofs of Propositions 6 to 9

We start with a number of lemmas that characterize the system's revenue under different configurations. Recall the simplified *binary system* configuration of Section 2.5, namely, a system with only two job values, $0 < v_1 < v_2$, and two delay sensitivities, $0 \leq \kappa_1 < \kappa_2$. The fraction of (v_i, κ_j) jobs is denoted as q_{ij}, $i,j \in \{1,2\}$, the fraction of v_1 jobs as r, and the fraction of κ_1 jobs as s, where $r,s \in (0,1)$, *i.e., r* and s are the marginals for v_1 and κ_1, respectively, so that job profiles are as follows

	κ_1	κ_2	
v_1	q_{11}	q_{12}	r
v_2	q_{21}	q_{22}	$1-r$
	s	$1-s$	

The next lemma was introduced in Section 2.5 and characterized the structure of the best one-price strategy in a binary system. We restate it and establish it formally.

Lemma 5. *In a binary system, the per unit of work expected revenue of the best one-price strategy is of the form*

$$R = \max\{v_1, (1-r)v_2\} \tag{2.10}$$

where $1-r$ *is the fraction of* v_2 *jobs in the system.*

Proof. Under a one-price strategy, there is no delay penalty, so that jobs derive a non-negative utility as long as the price does not exceed their value. It is then easy to see that the optimal price is either v_1 or v_2. If the price is v_1, all jobs adopt the service and bid at p_1 for a total revenue of $p_1 = v_1$. Conversely, if the price is $p_2 = v_2$, only jobs with value v_2 adopt the service and bid at p_2 for a revenue of $(1-r)v_2$. □

The next lemma identifies a necessary condition on bidding profiles for a two-price strategy to maximize revenue in a binary system.

Lemma 28. *If in a binary system, a two-price strategy dominates any one-price strategy,* i.e., *generates a strictly larger expected unit revenue, then jobs with profiles* $(v_1, \kappa_1), (v_2, \kappa_1)$ *and* (v_2, κ_2) *opt to bid, while jobs with profile* (v_1, κ_2) *do not bid,* i.e., *do not adopt the service.*

Proof. We first show that the optimal bidding strategy excludes jobs with profile (v_1, κ_2), and then show that it must attract all other jobs.

We show the exclusion of (v_1, κ_2) jobs by contradiction, namely, that if jobs with profile (v_1, κ_2) decide to bid, then the pricing scheme is not optimal. From Proposition 2 and the fact that both p_1 and p_2 are active bidding values, we know that jobs with profile (v_1, κ_2)

will bid at p_2 if they decide to bid. Furthermore, (v_1, κ_2) jobs bid iff $\overline{p}_2 \leq v_1$, where $\overline{p}_2$ is the expected payment per unit of execution time when bidding at p_2. As $p_1 < \overline{p}_2 \leq v_1$, all jobs that bid pay a price no more than $\overline{p}_2$, and since some jobs bid at p_1, some pay strictly less. Consider next a one-price pricing strategy charging $\overline{p}_2$. As $\overline{p}_2 \leq v_1 < v_2$, all jobs still bid and the strategy generates a higher expected revenue than the current pricing scheme. Hence, a contradiction.

Next we show that the optimal two-price strategy must attract bids from all other job types. Assume first that (v_2, κ_2) jobs do not bid, the service then only attracts κ_1 jobs. From Proposition 2, all κ_1 jobs bid at the same price. This contradicts the assumption that a two-price strategy dominates any one-price strategy. Assume next that (v_1, κ_1) jobs do not bid. In this case, the service attracts only v_2 jobs. From Lemma 5, there exists then a one-price strategy that extracts almost all values from those jobs, which again contradicts the assumption that a two-price strategy dominates any one-price strategy. Finally, combining Proposition 2 with the fact that (v_1, κ_1) jobs bid, we also know that (v_2, κ_1) will bid as well and at the same value. □

We note that an immediate corollary of Lemma 28 is

Corollary 29. *When a two-price strategy (p_1, p_2, π) maximizes revenue, the revenue per unit of work is of the form*

$$sp_1 + q_{22}(\pi p_1 + (1 - \pi)p_2) \tag{A.1}$$

The next lemma identifies conditions for a two-price strategy to be effective in a binary system.

Lemma 30. *In a binary system, a pricing strategy (p_1, p_2, π), where $p_1 < p_2$ and $0 < \pi < 1$, sees bids at both prices with jobs with profiles (v_1, κ_1), (v_2, κ_1) and (v_2, κ_2) bidding, if and only if the following inequalities hold:*

1. $\kappa_1 \leq \pi(p_2 - p_1) < \kappa_2$;
2. $v_1 - p_1 - \kappa_1(\frac{1}{\pi} - 1) \geq 0$;
3. $v_2 - (\pi p_1 + (1 - \pi)p_2) \geq 0$.

Proof. The first set of inequalities comes directly from Proposition 2, which indicates that jobs with κ_1 prefer p_1 while jobs with κ_2 prefer p_2. The second inequality ensures that jobs with value v_1 can afford bidding at p_1, whereas the third ensures that jobs with value v_2 can afford bidding at p_2. □

Next we characterize the solution of an optimization problem that serves as a basis for proving Propositions 6 and 9.

Lemma 31. *Given $0 \leq a_1, a_2 \leq 1$, the optimal solution to problem* **R**

$$\begin{aligned} \mathbf{R}: \quad & \underset{p_1,p_2,\pi}{\textit{maximize}} && R = a_1 p_1 + a_2 \left(\pi p_1 + (1-\pi)p_2\right) \\ & \textit{subject to} && \kappa_1 \leq \pi(p_2 - p_1) \leq \kappa_2, \\ & && v_2 - (\pi p_1 + (1-\pi)p_2) \geq 0, \\ & && v_1 - p_1 - \kappa_1(\frac{1}{\pi} - 1) \geq 0, \\ & && 0 \leq \pi \leq 1, 0 \leq p_1 \leq p_2, \end{aligned} \tag{A.2}$$

is $R^* = \max\{(a_1 + a_2)v_1, a_1\left(\frac{v_1\kappa_2 - v_2\kappa_1}{\kappa_2 - \kappa_1}\right) + a_2 v_2\}$.

Specifically, when $R^* = (a_1 + a_2)v_1$, *then* $p_1^* = v_1$, $\pi^* = 1$, *and* $p_2^* = v_1 + \kappa_1$; *and when* $R^* = a_1\left(\frac{v_1\kappa_2 - v_2\kappa_1}{\kappa_2 - \kappa_1}\right) + a_2 v_2$, *then* $p_1^* = \frac{\kappa_2 v_1 - \kappa_1 v_2}{\kappa_2 - \kappa_1}$, $\pi^* = \frac{\kappa_2 - \kappa_1}{\kappa_2 - \kappa_1 + v_2 - v_1}$, *and* $p_2^* = \frac{v_2 - \pi^* p_1^*}{1 - \pi^*}$.

Proof. As R increases with p_2, the optimal solution must be such that either one of the first two constraints are met, *i.e.,* either $\pi(p_2 - p_1) = \kappa_2$ or $\pi p_1 + (1 - \pi)p_2 = v_2$. Considering each case in turn

- If $\pi(p_2 - p_1) = \kappa_2$, we can rewrite Eq. (A.2) as

$$\begin{aligned} \underset{p_1,p_2,\pi}{\text{maximize}} \quad & R = a_1 p_1 + a_2\left(\kappa_2\left(\frac{1}{\pi} - 1\right) + p_1\right) \\ \text{subject to} \quad & p_1 \le v_1 - \kappa_1\left(\frac{1}{\pi} - 1\right), \\ & p_1 \le v_2 - \kappa_2\left(\frac{1}{\pi} - 1\right), \\ & 0 \le \pi \le 1, 0 \le p_1 \le p_2. \end{aligned}$$

  As R increases with p_1, the optimal solution is realized by having p_1 as large as possible, *i.e.,* $p_1^* = \min\{v_2 - \kappa_2(\frac{1}{\pi} - 1), v_1 - \kappa_1(\frac{1}{\pi} - 1)\}$, which is maximized by setting $\pi^* = 1$. This yields the optimal solution $R^* = (a_1 + a_2)v_1$, with $p_1^* = v_1$ and $p_2^* = v_1 + \kappa_1$.

- If $\pi p_1 + (1 - \pi)p_2 = v_2$, we can rewrite Eq. (A.2) as

$$\begin{aligned} \underset{p_1,p_2,\pi}{\text{maximize}} \quad & R = a_1 p_1 + a_2 v_2 \\ \text{subject to} \quad & \kappa_1 \le \pi\frac{v_2 - p_1}{1 - \pi} \le \kappa_2, \\ & p_1 \le v_1 - \kappa_1\left(\frac{1}{\pi} - 1\right), \\ & 0 \le \pi \le 1, 0 \le p_1 \le p_2. \end{aligned}$$

As R increases with p_1, the optimal solution is again realized by having p_1 as large as possible, while satisfying the constraints. The first constraint yields

$$v_2 - \kappa_2 \left(\frac{1}{\pi} - 1\right) \leq p_1 \leq v_2 - \kappa_1 \left(\frac{1}{\pi} - 1\right)$$

When combined with the second constraint, this implies $v_2 - \kappa_2(\frac{1}{\pi} - 1) \leq p_1 \leq v_1 - \kappa_1(\frac{1}{\pi} - 1)$. Under those constraints, maximizing p_1 is realized by setting $v_2 - \kappa_2(\frac{1}{\pi} - 1) = v_1 - \kappa_1(\frac{1}{\pi} - 1)$, which gives $\pi^* = \frac{\kappa_2 - \kappa_1}{\kappa_2 - \kappa_1 + v_2 - v_1}$, $p_1^* = \frac{\kappa_2 v_1 - \kappa_1 v_2}{\kappa_2 - \kappa_1}$, and $p_2^* = \frac{v_2 - \pi^* p_1^*}{1 - \pi^*}$, and correspondingly $R^* = a_1 \left(\frac{v_1 \kappa_2 - v_2 \kappa_1}{\kappa_2 - \kappa_1}\right) + a_2 v_2$.

□

Lemma B.23 is a technical lemma we use in the proofs of Propositions 6 to 9. Note that while its constraints are similar to the conditions of Lemma 30, they extend the constraints $0 < \pi < 1$ and $\pi(p_2 - p_1) < \kappa_2$ from open sets to closed sets to ensure the existence of a solution. This difference is the reason for the departure from the condition that bids exist at both prices in Lemma 30. When the solution is $R^* = a_1 \left(\frac{v_1 \kappa_2 - v_2 \kappa_1}{\kappa_2 - \kappa_1}\right) + a_2 v_2$, basic algebraic manipulation yields $\pi^*(p_2^* - p_1^*) = \kappa_2$. According to Proposition 2, jobs with κ_2 bid at p_1. When $R^* = (a_1 + a_2)v_1$, then $\pi^* = 1$, *i.e.,* only p_1 sees positive adoption.

Therefore, for the optimization under the constraints of Lemma 30, we have

Corollary 32. *Given $0 \leq a_1, a_2 \leq 1$, the supremum to problem* $\mathbf{R}$

$$\begin{aligned} \mathbf{R}: \quad & \underset{p_1,p_2,\pi}{\textit{maximize}} \quad R = a_1 p_1 + a_2 \left(\pi p_1 + (1-\pi) p_2\right) \\ & \textit{subject to} \quad \kappa_1 \leq \pi(p_2 - p_1) < \kappa_2, \\ & \qquad\qquad v_1 - p_1 - \kappa_1 \left(\frac{1}{\pi} - 1\right) \geq 0, \\ & \qquad\qquad v_2 - (\pi p_1 + (1-\pi) p_2) \geq 0, \\ & \qquad\qquad 0 < \pi < 1, 0 < p_1 < p_2, \end{aligned} \tag{A.3}$$

is $R^* = \max\{(a_1 + a_2)v_1, a_1 \left(\frac{v_1\kappa_2 - v_2\kappa_1}{\kappa_2 - \kappa_1}\right) + a_2 v_2\}$.

Specifically, when $R^* = (a_1 + a_2)v_1$, *then* $p_1^* = v_1$, $\pi^* = 1$, *and* $p_2^* = v_1 + \kappa_1$; *and when* $R^* = a_1 \left(\frac{v_1\kappa_2 - v_2\kappa_1}{\kappa_2 - \kappa_1}\right) + a_2 v_2$, *then* $p_1^* = \frac{\kappa_2 v_1 - \kappa_1 v_2}{\kappa_2 - \kappa_1}$, $\pi^* = \frac{\kappa_2 - \kappa_1}{\kappa_2 - \kappa_1 + v_2 - v_1}$, *and* $p_2^* = \frac{v_2 - \pi^* p_1^*}{1 - \pi^*}$.

A.6.1 Proof of Proposition 6

Recalling Proposition 6

Proposition 6. *Given a binary system with fixed marginals for job value and delay sensitivity, either there exists a value ρ^* such that for $\rho \leq \rho^*$ a one-price spot service is optimal and for $\rho > rho^*$ a two-price service is optimal, or else either one-price or two-price is optimal independent of ρ.*

Proof. Recall our earlier notation where v_1 and κ_1 have fixed marginals $r, s \in (0,1)$, respectively, and where q_{ij} is the probability that a job is of type (v_i, κ_j), $i, j \in \{1, 2\}$, *i.e.*,

	κ_1	κ_2	
v_1	q_{11}	q_{12}	r
v_2	q_{21}	q_{22}	$1-r$
	s	$1-s$	

The proof relies on the following three properties:

1. ρ increases with q_{22}, and $q_{22} = 0$ implies $\rho < 0$;
2. if at ρ_1, a two-price service is optimal, then for all $\rho > \rho_1$, a two-price service is also optimal;
3. if at ρ_2, a one-price service is optimal, then for all $\rho < \rho_2$, a one-price service is also optimal.

Proof of property 1:

The correlation coefficient ρ between (v, κ) is of the form

$$\rho = \frac{E[v\kappa] - E[v]E[\kappa]}{\sqrt{\mathrm{Var}(v)\,\mathrm{Var}(\kappa)}}. \tag{A.4}$$

Since with fixed marginals, $E[v], E[\kappa], \mathrm{Var}(v)$, and $\mathrm{Var}(\kappa)$ are also fixed, ρ increasing is equivalent to $E[v\kappa]$ increasing.

Rewriting q_{11}, q_{12}, and q_{21} as functions of q_{22}, r and s, we have: $q_{11} = r + s + q_{22} - 1, q_{12} = 1 - s - q_{22}$, and $q_{21} = 1 - r - q_{22}$. Using these expressions in Eq. (A.4) yields after some algebraic manipulations:

$$\rho = \frac{q_{22} - (1-r)(1-s)}{\sqrt{rs(1-r)(1-s)}}, \tag{A.5}$$

which, since the marginals r and s are fixed, increases if and only if q_{22} increases. Furthermore, because $0 < (1-r)(1-s) < 1$, we also have that $q_{22} = 0 \Rightarrow \rho < 0$, and that $q_{22} = 1 \Rightarrow \rho > 0$. This establishes property 1).

Proof of property 2:

Recall first from Lemma 5 that if a one-price strategy is optimal for a given configuration with fixed marginals, then its expected optimal revenue per unit of work is $\max\{v_1, v_2(1-r)\}$, which is independent of q_{22}.

Conversely, in a scenario where a two-price strategy is optimal, combining Corollary 29 and Lemma 30 gives the following optimization (akin to that of Corollary 32) w.r.t the optimal expected revenue per unit of work:

$$\begin{aligned} \underset{p_1,p_2,\pi}{\text{maximize}} \quad & sp_1 + q_{22}[\pi p_1 + (1-\pi)p_2] \\ \text{subject to} \quad & \kappa_1 \leq \pi(p_2 - p_1) < \kappa_2, \\ & v_1 - p_1 - \kappa_1 \left(\frac{1}{\pi} - 1\right) \geq 0, \\ & v_2 - \pi p_1 - (1-\pi)p_2 \geq 0, \\ & 0 < \pi < 1, 0 < p_1 < p_2, \end{aligned} \tag{A.6}$$

where the constraints are independent of q_{22}, and the objective function increases with q_{22}. Note that the strict inequalities in the constraints $0 < \pi < 1, 0 < p_1 < p_2$, are needed for a two-price policy to be optimal.

Combining these three conditions (independence from q_{22} of the optimal one-price revenue, independence from q_{22} of the constraints for the optimality of a two-price strategy, and growth with q_{22} of revenue under an optimal two-price strategy), we know that if, for fixed marginals

and a given value of q_{22}, a two-price strategy (p_1, p_2, π) generates a greater expected revenue per unit of work than the best one-price strategy, then the result still holds as q_{22} increases with the marginals remaining the same. As a result, since for fixed marginals, property 1 states that a larger ρ is equivalent to a larger q_{22}, this establishes property 2.

Proof of property 3:

The proof of property 3 is similar to that of property 2, and is omitted for brevity.

What properties 2 and 3 tell us is that if for fixed marginals, there exists a correlation coefficient for which a two-price (one-price) strategy is optimal, then the same holds true for all correlation coefficients above (below) that value. Next we characterize the value of the correlation coefficient ρ^* for which a change in optimal pricing strategy (from one-price to two-price) can occur.

If a two-price system generates more revenue than that of the optimal one-price system, from Corollary 32 with $a_1 = s$ and $a_2 = q_{22}$ and Lemma 5, we have

$$s\left(\frac{v_1\kappa_2 - v_2\kappa_1}{\kappa_2 - \kappa_1}\right) + q_{22}v_2 > \max\{v_1, (1-r)v_2\},$$

which is equivalent to

$$\begin{cases} q_{11}(v_1\kappa_2 - v_2\kappa_1) > q_{21}\kappa_2(v_2 - v_1) \\ (v_2 - v_1)[q_{22} - \frac{s\kappa_1}{\kappa_2 - \kappa_1}] > q_{12}v_1 \end{cases} \tag{A.7}$$

As $[q_{11}, q_{12}, q_{21}] = [r + s + q_{22} - 1, 1 - s - q_{22}, 1 - r - q_{22}]$, Eq. (A.7) is equivalent to

$$q_{22} > \max\left\{1 - r - \frac{s(v_1\kappa_2 - v_2\kappa_1)}{v_2(\kappa_2 - \kappa_1)}, \frac{v_1}{v_2} - \frac{s(\kappa_2 v_1 - \kappa_1 v_2)}{v_2(\kappa_2 - \kappa_1)}\right\}. \tag{A.8}$$

Substituting it in Eq. (A.5) gives

$$\rho^* = \frac{\max\{1-r, \frac{v_1}{v_2}\} - \frac{s(v_1\kappa_2 - v_2\kappa_1)}{v_2(\kappa_2-\kappa_1)} - (1-r)(1-s)}{\sqrt{rs(1-r)(1-s)}}.$$

Recall that a two-price strategy is considered optimal iff it generates a strictly greater expected unit revenue than any one-price strategy. Hence, if $\rho^* \in [-1, 1)$, then the optimal pricing strategy switches from a one-price strategy to a two-price strategy as ρ exceeds ρ^*. Conversely, if ρ^* lies outside $[-1, 1)$, then the optimal pricing strategy remains unchanged for all (feasible) values of the correlation coefficient ρ. □

A.6.2 Proof of Corollary 7

Restating the corollary

Corollary 7. *Given a binary system with fixed marginals r and s for job value and delay sensitivity, and a given correlation coefficient ρ between job value and delay sensitivity, there exists a value $\rho^* \geq 0$ given by*

$$\rho^* = \frac{\max\{1-r, \frac{v_1}{v_2}\} - \frac{s(v_1\kappa_2 - v_2\kappa_1)}{v_2(\kappa_2-\kappa_1)} - (1-r)(1-s)}{\sqrt{rs(1-r)(1-s)}} \tag{2.11}$$

such that

- *when $\rho^* \in [0, 1)$, then for $\rho \leq \rho^*$ a one-price spot service is optimal, and for $\rho > \rho^*$ a two-price spot service is optimal;*
- *otherwise a one-price spot service is always optimal independent of ρ.*

Proof. If $\rho^* \geq 0$, then $\rho^* \notin [0,1)$ implies $\rho^* \geq 1$, in which case Property 3 in the proof of Proposition 6 implies that a one-price strategy is optimal for all values of ρ. Hence, establishing the corollary amounts to proving that $\rho^* \geq 0$, where the expression for ρ^* was derived in the proof of Proposition 6.

As the denominator of ρ^* is always non-negative, ρ^* and its numerator have the same sign. Next, we show that the numerator of ρ^* is also always non-negative. For that purpose, we consider separately the cases $1 - r \geq \frac{v_1}{v_2}$ and $1 - r < \frac{v_1}{v_2}$, and prove that the numerator is non-negative in both cases.

When $1 - r \geq \frac{v_1}{v_2}$, the numerator of ρ^* becomes

$$s(1-r) - \frac{s(v_1\kappa_2 - v_2\kappa_1)}{v_2(\kappa_2 - \kappa_1)} \geq s\left[\frac{v_1}{v_2} - \frac{(v_1\kappa_2 - v_2\kappa_1)}{v_2(\kappa_2 - \kappa_1)}\right] = s\frac{\kappa_1(v_2 - v_1)}{v_2(\kappa_2 - \kappa_1)} \geq 0. \tag{A.9}$$

When $1 - r < \frac{v_1}{v_2}$, the numerator of ρ^* becomes

$$\begin{aligned}\frac{v_1}{v_2} - \frac{s(v_1\kappa_2 - v_2\kappa_1)}{v_2(\kappa_2 - \kappa_1)} - (1-r)(1-s) &= s\left[\frac{v_1}{v_2} - \frac{(v_1\kappa_2 - v_2\kappa_1)}{v_2(\kappa_2 - \kappa_1)}\right] + (1-s)\left[\frac{v_1}{v_2} - (1-r)\right] \\ &> s\left[\frac{v_1}{v_2} - \frac{(v_1\kappa_2 - v_2\kappa_1)}{v_2(\kappa_2 - \kappa_1)}\right] = s\frac{\kappa_1(v_2 - v_1)}{v_2(\kappa_2 - \kappa_1)} \geq 0.\end{aligned} \tag{A.10}$$

□

A.6.3 Proof of Proposition 8

Restating the proposition.

Proposition 8. *In a binary system with given job values, delay sensitivities, and corresponding marginals, the prices used in the best one-price and two-price strategies are independent of the correlation coefficient ρ between job value and delay sensitivity.*

Proof. Lemma 5 directly establishes that the best one-price strategy is independent of ρ. Similarly, the expressions for p_1^*, p_2^*, and π^* in Corollary 32 provide a similar result for the best two-price strategy. □

A.6.4 Proof of Proposition 9

Proposition 9. *When in a binary system job value and delay sensitivity are perfectly positively correlated,* i.e., *the system only has* (v_1, κ_1) *and* (v_2, κ_2) *jobs, where* $0 < v_1 < v_2$ *and* $0 \leq \kappa_1 < \kappa_2$*, then using* s *to denote the fraction of* (v_1, κ_1) *jobs, we have*

- *When* $\kappa_2(1-s) - \kappa_1 > 0$ *and* $v_1\kappa_2 > v_2\kappa_1$*, a two-price spot service is optimal;*
- *Otherwise, a one-price spot service is optimal.*

Proof. As a two-price system can only be optimal if both job profiles bid, we will characterize the conditions under which this occurs.

Consider a two-price system (p_1, p_2, π), where both prices have positive adoption, the expected revenue is then

$$R = p_1 s + (\pi p_1 + (1-\pi)p_2)(1-s),$$

where we have again used the result of Proposition 2 and the fact that by assumption, the two job profiles bid at different prices. This yields the following optimization focused on

optimizing the expected revenue per unit of work:

$$\begin{aligned} \underset{p_1,p_2,\pi}{\text{maximize}} \quad & p_1 s + [\pi p_1 + (1-\pi)p_2](1-s) \\ \text{subject to} \quad & \kappa_1 \leq \pi(p_2 - p_1) < \kappa_2, \\ & v_1 - \kappa_1(\frac{1}{\pi} - 1) - p_1 \geq 0, \\ & v_2 - \pi p_1 - (1-\pi)p_2 \geq 0, \\ & 0 < \pi < 1, 0 < p_1 < p_2. \end{aligned} \tag{A.11}$$

As before, solving the above optimization calls for relaxing its constraints to use large rather than strict inequalities. The corresponding solution, R^*, is then again a supremum of the actual solution. We also know that the best one-price revenue per unit of work is given by $R^*(1) = \max\{v_1, v_2(1-s)\}$, so that for a two-price strategy to be optimal, we need $R^* > R^*(1)$.

From Corollary 32 with $a_1 = s$ and $a_2 = 1 - s$, the supremum of the optimization is $R^* = \max\{v_1, v_2 - \kappa_2 s\frac{v_2 - v_1}{\kappa_2 - \kappa_1}\}$. Basic algebraic manipulation show that $R^* > R^*(1)$ iff $\kappa_2(1-s) - \kappa_1 > 0$ and $v_1\kappa_2 > v_2\kappa_1$. □

A.6.5 Deterministic Environments

As mentioned earlier, the main result of the chapter, *i.e.,* Proposition 6, also holds under two different settings: 1) preemptible VMs, and 2) multiple processor speeds. In this appendix, we characterize both settings more precisely and formally establish that Proposition 6 still holds.

We assume the same utility function as in Section 3.3.1, and use p to denote the unit price of the selected service. A job's utility function is then as before:

$$U(t, v, \kappa, p) = vt - pt - \kappa T(t, v, \kappa, p), \tag{A.12}$$

where $T(t, v, \kappa, p)$ is the job's expected "delay." For preemptible VMs, the delay is defined as the expected duration of preemption periods. For processors with different speeds, the delay is the difference in execution times between running the same job on the selected processor and running it on the fastest available processor.

As in Section 2.5, for simplicity we limit ourselves to a binary system, *i.e.,* binary job profiles with only two job values ($0 < v_1 < v_2$) and delay sensitivities ($0 \leq \kappa_1 < \kappa_2$). Next, we first formulate the revenue maximization problem for both preemptible VMs and multiple processor speeds, and then show that under these new settings the expected revenue under the optimal one-price and two-price strategies have similar expressions as in the spot service setting. We then use this equivalence to establish that Proposition 6 still holds.

Model Formulation

Preemptible VMs Denote the unit price for a preemptible VM as $0 < p_1$, and $p_2 > p_1$ for a non-preemptible VM. Assume that for preemptible VMs the probability $(1 - \beta)$ of being preempted is known to the customer, where preemptions (and restarts) occur at discrete points in time (slot boundaries in relation to our spot service model). The expected utility of a job (t, v, κ) when choosing the preemptible VM service is

$$U_p(t, v, \kappa) = vt - p_1 t - \kappa t \left(\frac{1}{\beta} - 1 \right),$$

where $(1/\beta - 1)$ captures the expected increase in completion time caused by preemptions.

Alternatively, its utility when choosing a non-preemptible VM is

$$U_n(t, v, \kappa) = vt - p_2 t.$$

In keeping with our assumption that given equal utilities, customers prefer the cheaper service, we assume that customers prefer preemptible VMs given the same expected utility. Then a (t, v, κ) job prefers a non-preemptible VM over a preemptible VM iff

$$\kappa > \frac{p_2 - p_1}{1/\beta - 1}. \tag{A.13}$$

Note that we also still have the condition that a job's utility must be non-negative for it to adopt the service.

Multiple Processor Speeds Assume that the service provider has n processors with speeds $\boldsymbol{\phi} = (\phi_1, ..., \phi_n)$ and corresponding prices $\boldsymbol{p} = (p_1, ..., p_n)$, where $0 < \phi_1 < ... < \phi_n = 1$ and $0 < p_1 < ... < p_n$. If job (t, v, κ) chooses processor i, it receives a utility of the form:

$$U_i(t, v, \kappa) = vt - p_i t - \kappa t \left(\frac{1}{\phi_i} - 1 \right).$$

In other words, the utility decreases in proportion to the increase in execution time compared to using the fastest processor. Again, when two processor speeds generate the same utility, we assume that the customer prefers the cheaper one. A customer's optimal service is then given by[55]

$$\min \arg\min_i p_i + \kappa \left(\frac{1}{\phi_i} - 1 \right),$$

[55]Note that the min operator in the expression is to break ties in case multiple speeds yield the same utility.

which is independent of v and t. Hence, in a binary system, the service provider only needs to offer at most two different processor speeds.

Denote the two services and corresponding prices and processor speeds as (p_1, ϕ) and $(p_2, 1)$, where $0 < \phi < 1$. A job (t, v, κ) then prefers the faster processor to the slower one iff

$$\kappa > \frac{p_2 - p_1}{1/\phi - 1},$$

which is identical to the expression obtained for preemptible VMs. As a result, we focus on establishing that Proposition 6 still holds for preemptible VMs and denote the provider's strategy in that setting as (p_1, p_2, β), where $p_1 = p_2$ or $\beta = 1$ corresponds to a system where preemptible VMs are not offered.

Optimal Pricing Strategy

As before, we denote the fraction of (v_i, κ_j) jobs as $q_{ij}, i, j \in \{1, 2\}$, the fraction of v_1 jobs as r, and the fraction of κ_1 jobs as s, where $r, s \in (0, 1)$.

First note that under the preemptible VMs setting, Lemma 5, which characterizes the optimal one-price strategy, holds with an identical proof. As a result, the preemptible VMs and spot price settings generate the same optimal one-price revenue given the same distribution of job profiles.

Next we show that the preemptible VMs and spot price settings also generate the same expected revenue under the optimal two-price strategy.

In the preemptible VMs setting, the proof of Lemma 28, which states that only (v_1, κ_2) jobs do not adopt the service when a two-price system is optimal, holds after trivial modifications.

Hence, we omit repeating it. Based on Lemma 28, we then have a modified version of Corollary 29:

COROLLARY 29'. *When a two-price strategy (p_1, p_2, β) maximizes revenue, the revenue per unit of work is of the form*

$$sp_1 + q_{22}p_2 \tag{A.14}$$

The proofs for Corollary 29' and Corollary 29 follow identical steps that we briefly recall for the reader's convenience. From Lemma 28, we know that a two-price strategy is optimal iff only $(v_1, \kappa_1), (v_2, \kappa_1)$ and (v_2, κ_2) jobs adopt the service. Because 1) jobs with the same delay sensitivity prefer the same service and 2) a higher delay sensitivity implies a preference for a higher-price service, a two-price strategy is optimal iff κ_1 jobs choose p_1 and (v_2, κ_2) jobs choose p_2. Hence, the revenue per unit of work under an optimal two-price strategy is of the form $sp_1 + q_{22}p_2$.

Similarly, we can state a parallel version of Lemma 30 that identifies the conditions under which a two-price strategy is effective in a (binary) preemptible VMs settings:

Lemma 30'. *In a binary system, a pricing strategy (p_1, p_2, β), where $p_1 < p_2$ and $0 < \beta < 1$, sees adoptions at both prices with jobs with profiles (v_1, κ_1), (v_2, κ_1) and (v_2, κ_2) adopting, if and only if the following inequalities hold:*

1. $\kappa_1 \leq \frac{p_2 - p_1}{1/\beta - 1} < \kappa_2$;
2. $v_1 - p_1 - \kappa_1 \left(\frac{1}{\beta} - 1\right) \geq 0$;
3. $v_2 - p_2 \geq 0$.

Lemma 30' follows the same logic as Lemma 30, with the first constraint derived directly from Eq. (A.13) and enforcing that κ_i jobs prefer $p_i, i \in \{1, 2\}$; the second constraint guaranteeing that κ_1 jobs have a non-negative utility when selecting p_1; and the third constraint ensuring that (v_2, κ_2) jobs have a non-negative utility when selecting p_2.

As before, the formulation of the optimization to maximize revenue per unit of work under a two-price system follows directly from Corollary 29' and Lemma 30':

$$\begin{aligned} \underset{p_1,p_2,\beta}{\text{maximize}} \quad & sp_1 + q_{22}p_2 \\ \text{subject to} \quad & \kappa_1 \leq \frac{p_2 - p_1}{1/\beta - 1} < \kappa_2, \\ & v_1 - p_1 - \kappa_1 \left(\frac{1}{\beta} - 1\right) \geq 0, \\ & v_2 - p_2 \geq 0, \\ & 0 < \beta < 1, 0 < p_1 < p_2. \end{aligned} \tag{A.15}$$

Next we show that given the same binary system, optimization (A.15) generates the same expected revenue as optimization (A.6) of Proposition 6, which we recall next

$$\begin{aligned} \underset{p_1,p_2,\pi}{\text{maximize}} \quad & sp_1 + q_{22}[\pi p_1 + (1 - \pi)p_2] \\ \text{subject to} \quad & \kappa_1 \leq \pi(p_2 - p_1) < \kappa_2, \\ & v_1 - p_1 - \kappa_1 \left(\frac{1}{\pi} - 1\right) \geq 0, \\ & v_2 - \pi p_1 - (1 - \pi)p_2 \geq 0, \\ & 0 < \pi < 1, 0 < p_1 < p_2. \end{aligned} \tag{A.6}$$

By substitution, we can reduce optimization (A.15) to optimization (A.6). Specifically, we introduce a new variable $\tilde{p}_2$ defined as $\tilde{p}_2 = \frac{p_2 - \beta p_1}{1 - \beta}$ for $0 < \beta < 1$. This allows us to rewrite

p_2 as $\beta p_1 + (1-\beta)\tilde{p}_2$. By substituting p_2 with $\beta p_1 + (1-\beta)\tilde{p}_2$ in (A.15), we get a new optimization w.r.t $p_1, \tilde{p}_2$ and β. Note that in (A.15) we have the constraint $0 < \beta < 1$, therefore, the new optimization has the same optimality as the previous one.

After substitution, basic algebraic manipulations give

$$\begin{aligned} \underset{p_1,\tilde{p}_2,\beta}{\text{maximize}} \quad & sp_1 + q_{22}[\beta p_1 + (1-\beta)\tilde{p}_2] \\ \text{subject to} \quad & \kappa_1 \leq \beta(\tilde{p}_2 - p_1) < \kappa_2, \\ & v_1 - p_1 - \kappa_1 \left(\frac{1}{\beta} - 1\right) \geq 0, \\ & v_2 - \beta p_1 - (1-\beta)\tilde{p}_2 \geq 0, \\ & 0 < \beta < 1, 0 < p_1 < \tilde{p}_2, \end{aligned}$$

which is identical to (A.6). Therefore, both settings generate the same optimal two-price expected revenue.

Next, we show Proposition 6 based on the fact that both settings have the same optimal expected revenue under either one-price or two-price strategies. Recall that the proof of Proposition 6 involves three properties:

1. ρ (the correlation between v and κ) increases with q_{22}, and $q_{22} = 0$ implies $\rho < 0$;
2. if at ρ_1, a two-price service is optimal, then for all $\rho > \rho_1$, a two-price service is also optimal;
3. if at ρ_2, a one-price service is optimal, then for all $\rho < \rho_2$, a one-price service is optimal.

Property 1) depends only on job profiles and their distribution. Hence, it holds regardless of settings. Properties 2) and 3) essentially compare the expected revenue between the optimal

one-price and two-price strategies. Since both settings (preemptible VMs and spot instances) have the same optimal one-price and two-price revenues given the same distribution of job profiles, Properties 2) and 3) also hold in the preemptible VMs setting. This then establishes that Proposition 6 still holds.

A.6.6 Impact of System Parameters on ρ^*

From Eq. (2.11), we readily see that ρ^* increases with κ_1 and decreases with κ_2. This is intuitive given our analysis of the optimal pricing strategy. A larger κ_1 implies a smaller p_1^* and therefore a smaller revenue from (v_1, κ_1) jobs for a two-price strategy. Compensating for this decrease calls for more (v_2, κ_2) jobs and/or fewer (v_2, κ_1) jobs, *i.e.,* a larger ρ^*. Conversely, a larger κ_2 allows the service provider to use a larger p_2^* or π^* (and therefore p_1^* in the latter case). This helps improve revenue over the best one-price strategy, so that fewer (v_2, κ_2) jobs are needed (or more (v_2, κ_1) jobs can be tolerated), and correspondingly a smaller ρ^* value.

The relationship between ρ^* and v_1 and v_2 is more complex, and it is therefore harder to derive explicit insight from it. Specifically, we have

1. when $(1-r)v_2 \geq v_1$, ρ^* increases as v_1 decreases and decreases as v_2 decreases;
2. when $(1-r)v_2 < v_1$ and $\kappa_2(1-s) > \kappa_1$, ρ^* increases as v_1 increases and as v_2 decreases;
3. otherwise, ρ^* decreases with v_1 and increases with v_2.

Combining Propositions 6 and 9, we know that a two-price strategy is never optimal when $\kappa_2(1-s) \leq \kappa_1$. Hence, we focus only on cases 1) and 2).

Recall that as per Lemma 5, the best one-price strategy either has a price $p^* = v_1$, in which case all jobs bid at p^* for an expected unit revenue of v_1, or it has a price of $p^* = v_2$ that only v_2 jobs can afford, and correspondingly an expected unit revenue of $(1 - r)v_2$. Hence, under case 1), the optimal one-price strategy only targets v_2 jobs. In this case, the primary advantage of a two-price strategy comes from its ability to attract (v_1, κ_1) jobs (it yields a lower revenue from (v_2, κ_1) jobs and the same revenue from (v_2, κ_2) jobs). A decrease in v_1 lowers the additional revenue from (v_1, κ_1) jobs and increases the loss from (v_2, κ_1) jobs. Compensating for this calls for fewer (v_2, κ_1) jobs and consequently a larger ρ^* value. Conversely, a decrease in v_2 decreases the loss a two-price strategy incurs from (v_2, κ_1) jobs. Hence, the two-price strategy needs fewer (v_1, κ_1) jobs to outperform the one-price strategy, *i.e.,* a smaller ρ^* value.

Under case 2), the optimal one-price strategy has a price $p^* = v_1$ and a corresponding expected unit revenue of v_1. Recall that since $p_1^* < p^*$ (because of the added delay cost), a two-price strategy outperforms a one-price strategy only if its revenue loss from $(v_1, \kappa_1), (v_2, \kappa_1)$, and (v_1, κ_2) jobs is more than offset by the increased revenue from (v_2, κ_2) jobs. Computing the difference between gain and loss under the optimal two-price strategy when the condition $\kappa_2(1 - s) > \kappa_1$ holds, establishes that it decreases as v_1 increases, with the loss eventually exceeding the gain. Compensating for this calls for a greater fraction of (v_2, κ_2) jobs, *i.e.,* a larger ρ^*. Similarly, a decrease in v_2 implies a lesser gain from (v_2, κ_2) jobs, and therefore again a larger ρ^* value to compensate for the shortfall.

A.7 Dynamic Bidding Strategy

In this section, we construct optimal dynamic bidding strategies using backward induction. We consider dynamic bidding strategies under three scenarios: 1) jobs have a linear delay

penalty function but can terminate, 2) jobs have a piece-wise linear convex delay penalty function, and 3) jobs have a piece-wise linear concave delay penalty function. Note that in cases 2) and 3), we assume that jobs do not terminate once they start bidding.

Let $\widehat{T}$ be a job's current execution delay, *i.e.,* the number of losing bids it has had so far, and $\widehat{t}$ its residual size, *i.e.,* the amount of work left to complete the job. Under this notation, a new job has $\widehat{T} = 0$ and $\widehat{t} = t$. Denote a job's current state as $(\widehat{T}, \hat{t}, \widehat{P})$, where $\widehat{P}$ is its accumulated payment to-date, and its residual state as $(\widehat{T}, \hat{t})$.

For all three scenarios, we first characterize their MDPs, and then complete the proof using backward induction. In the backward induction, we rely on a job's expected residual utility to compute an optimal substrategy. To distinguish it from the job's standard utility, U, we use the notation $\widetilde{U}$ ($\widetilde{U}^*$) for a job's (optimal) expected residual utility.

For the reader's convenience, we repeat the backward induction algorithm here.

The Backward Induction Algorithm [154]

1. Set $z = N$ and
$$u_N^*(s_N) = r_N(s_N) \qquad \text{for all } s_N \in S,$$

2. Substitute $z - 1$ for z and compute $u_z^*(S_z)$ for each $s_z \in S$ by
$$u_z^*(s_z) = \max_{a \in A_{s_z}} \left\{ r_z(s_z, a) + \sum_{j \in S} p_t(j|s_z, a) u_{z+1}^*(j) \right\}.$$
Set
$$A_{s_z,d}^* = \arg\max_{a \in A_{s_z}} \left\{ r_z(s_z, a) + \sum_{j \in S} p_z(j|s_z, a) u_{z+1}^*(j) \right\}.$$

3. If $z = 1$, stop. Otherwise return to step 2.

Here z denotes the decision epoch[56], N is the largest possible decision epoch, S is the set of possible system states, and A_s is the set of allowable actions in state s. $r_z(s,z)$ denotes the expected reward (or cost) when the state of the system at decision epoch z is s, and action $a \in A_s$ is selected; $p_z(j|s,a)$ denotes the probability that the system is in state $j \in S$ at time $z+1$, when the decision maker chooses action $a \in A_s$ in state s at time z.

As in Section 2.6, for tractability we limit ourselves to binary systems, and therefore pricing strategies with at most two prices, *i.e.,* (p_1, p_2, π).

A.7.1 Case 1: Linear Delay Penalty with Termination

In this sub-section, we characterize the optimal bidding strategy under a linear delay penalty when jobs are allowed to stop bidding. Recall that jobs stop bidding once their expected residual utility reaches zero, where a job's expected residual utility represents the utility it can expect to generate going forward from completing its execution. In other words, the residual utility ignores the payments $\widehat{P}$ already made, but accounts for the delay penalty $\kappa\widehat{T}$ the job has incurred, *i.e.,* it assumes a residual value of $\widehat{V}(t) = vt - \kappa\widehat{T}$ for the job. The rationale for using it as a termination criterion is that it captures whether or not continuing to bid can be expected to improve the job's final utility. If not, it is then best to terminate the job, even if it means incurring a loss of $\widehat{P}$.

For example and for illustration purposes, under a fixed bidding strategy and without termination, a job's expected residual utility when in state $(\widehat{T}, \widehat{t}, \widehat{P})$ is independent of $\widehat{P}$ and easy to compute, *i.e.,*

[56] [154] uses t to denote decision epochs. Unfortunately, we already used the variable t to denote the job length. To avoid any confusion, we therefore replaced t with z in the algorithm.

$$\widetilde{U}(\widehat{T},\widehat{t}) = \begin{cases} (vt - \kappa\widehat{T}) - \widehat{t}p_1 - \kappa\widehat{t}\left(\frac{1-\pi}{\pi}\right), & \text{if bid} = p_1 \\ (vt - \kappa\widehat{T}) - \widehat{t}(\pi p_1 + (1-\pi)p_2), & \text{if bid} = p_2 \end{cases} \tag{A.16}$$

The situation is more complex under dynamic bidding strategies and when termination is allowed. In the rest of this section, we identify how to compute a job's expected residual utility under such conditions, and in the process characterize optimal dynamic bidding strategies with termination. Specifically, we first construct the MDP for the bidding process before characterizing base strategies for $\hat{t} = 1$ and $\widehat{T}$ large, and then formalize the backward induction recurrence for the general case.

MDP Construction We characterize job (t, v, κ)'s bidding strategy using four types of states: start state (S_0), bid states ($S_{\widehat{T},\widehat{t}}$), terminate state ($S_{ter}$) and finish state ($S_f$), where S_{ter} and S_f are absorbing states. A job starts in S_0 and goes to state $S_{0,t}$ with a reward of vt if it decides to adopt the service. Otherwise, it exits and goes to state S_f with a reward of 0. In state $S_{\widehat{T},\widehat{t}}$, a job can either bid with a value in $\{p_1, p_2\}$ or terminate. In particular, when $\widehat{T} = \frac{vt}{\kappa}$, the job's residual utility is smaller than $vt - \kappa\widehat{T} = 0$ whatever its bidding strategy up to that point. Hence, it terminates bidding at the latest when $\widehat{T} = \frac{vt}{\kappa}$. If the job bids at p_1, it goes to $S_{\widehat{T},\widehat{t}-1}$ with a reward of $-p_1$ with probability π, and to $S_{\widehat{T}+1,\widehat{t}}$ with a reward of $-\kappa$ with probability $(1-\pi)$. If the job bids at p_2, it goes to $S_{\widehat{T},\widehat{t}-1}$ with an expected reward of $-\pi p_1 - (1-\pi)p_2$. If the job terminates, it goes to S_{ter} with a reward of $-vt + \kappa\widehat{T}$ (the final utility of a terminated job is simply the bidding costs it has incurred prior to termination). Formally:

Decision epochs[57]:

$$T = \{1, 2, ..., \frac{vt}{\kappa} + t\}.$$

[57]At time epoch $\frac{vt}{\kappa} + t$, the job incurred a delay of at least $\frac{vt}{\kappa}$, so that its residual utility is no longer positive.

States:

$$S = \{S_0,\ S_{\widehat{T},\widehat{t}},\ S_{ter},\ S_f\}, \qquad \text{where } \widehat{t} < t, \text{ and } S_{\widehat{T},0} := S_f.$$

Actions:

$$A_0 = \{adopt, exit\}; \qquad A_{\widehat{T},\widehat{t}} = \{p_1, p_2, terminate\}; \qquad A_{\frac{vt}{\kappa},\widehat{t}} = \{terminate\}.$$

Expected rewards:

$$r(adopt, S_0) = vt; \qquad r(exit, S_0) = 0; \qquad r(terminate, S_{\widehat{T},\widehat{t}}) = -vt + \kappa\widehat{T};$$

$$r(p_1, S_{\widehat{T},\widehat{t}}) = -\pi p_1 - (1-\pi)\kappa; \qquad r(p_2, S_{\widehat{T},\widehat{t}}) = -\pi p_1 - (1-\pi)p_2.$$

Transition probabilities (all omitted transitions, except from absorbing states to themselves , have a probability of 0):

$$Prob(S_{0,t}|S_0, adopt) = 1; \qquad Prob(S_f|S_0, exit) = 1; \qquad Prob(S_{ter}|S_{\widehat{T},\widehat{t}}\,, terminate) = 1;$$

$$Prob(S_{\widehat{T},\widehat{t}-1}|S_{\widehat{T},\widehat{t}}\,,\ p_1) = \pi; \qquad Prob(S_{\widehat{T}+1,\widehat{t}}|S_{\widehat{T},\widehat{t}}\,,\ p_1) = 1-\pi; \qquad Prob(S_{\widehat{T},\widehat{t}-1}|S_{\widehat{T},\widehat{t}}\,,\ p_2) = 1.$$

Base strategies ($\widehat{t} = 1$ and $\widehat{T}$ large) As the corresponding expressions are useful in deriving base strategies, we first characterize the residual utility for a job of residual size $\widehat{t} = 1$ of strategies that involve bidding at a fixed price (p_1 or p_2) until completion or termination.

The expected residual utility $\widetilde{U}_2(\widehat{T}, 1)$ when bidding at p_2 is directly obtained from Eq. (A.16).

$$\widetilde{U}_2(\widehat{T}, 1) = (vt - \kappa\widehat{T}) - (\pi p_1 + (1-\pi)p_2). \tag{A.17}$$

Computing the expected residual utility when bidding at p_1 calls for first identifying when the strategy would decide to terminate bidding. Assume that bidding stops after k consecutive failed bids at p_1. The job's expected residual utility $\widetilde{U}_1^{(k)}(\widehat{T}, 1)$ is then given by

$$\widetilde{U}_1^{(k)}(\widehat{T}, 1) = \sum_{i=0}^{k-1} (1-\pi)^i \pi (vt - \kappa\widehat{T} - p_1 - \kappa i). \tag{A.18}$$

Eq. (A.18) immediately establishes when termination should occur, *i.e.,* after $\widehat{k}$ failed bids where $\widehat{k}$ is the smallest value such that

$$(vt - \kappa\widehat{T} - p_1 - \kappa\widehat{k}) \leq 0 \quad \Rightarrow \widehat{k} = \left\lfloor \frac{vt - p_1}{\kappa} - \widehat{T} \right\rfloor. \tag{A.19}$$

With expressions for fixed bidding strategies (with termination) in place for jobs in residual state $(\widehat{T}, 1)$, we are now ready to state a lemma that establishes that we only need to consider those fixed bidding strategies.

Lemma 33. *Under a two-price pricing strategy, a linear delay penalty, and with job termination allowed, the optimal bidding strategy for jobs whose residual size is $\widehat{t} = 1$ is to either terminate bidding, or bid at p_2, or bid at p_1 until termination or completion.*

Proof. In order to establish the lemma, we only need to rule out strategies that initially starts bidding at p_1 before switching to bidding at p_2 after a number of unsuccessful bids.

Denote as $\widetilde{U}^*(\widehat{T}, 1 \mid p_1)$ and $\widetilde{U}^*(\widehat{T}, 1 \mid p_2)$ the expected residual utility in residual state $(\widehat{T}, 1)$ generated by the optimal sub-strategy, when it starts with a bid at p_1 or p_2, respectively.

Next, we show that if $\widetilde{U}^*(\widehat{T}, 1 \mid p_1) > \widetilde{U}^*(\widehat{T}, 1 \mid p_2)$, then $\widetilde{U}^*(\widehat{T}+1, 1 \mid p_1) > \widetilde{U}^*(\widehat{T}+1, 1 \mid p_2)$ when $\widetilde{U}^*(\widehat{T}+1, 1 \mid p_2) > 0$. In other words, if bidding at p_1 is initially superior to bidding at p_2, then it remains so after an unsuccessful first bid as long as termination is not preferred over bidding at p_2. In this latter case, bidding at p_2 can never replace bidding at p_1, *i.e.,* the optimal sub-strategy will either bid at p_1 or terminate.

Note first that $\widetilde{U}^*(\widehat{T}+1, 1 \mid p_1) \geq \widetilde{U}^*(\widehat{T}, 1 \mid p_1) - \kappa$. In other words, the optimal sub-strategy that starts by bidding at p_1 in residual state $(\widehat{T}+1, 1)$ generates a residual utility that is at least as good as the residual utility generated by reusing the same strategy as in residual state $(\widehat{T}, 1)$ minus the utility lost to the delay from the additional lost bid. More formally, this yields a utility of the form

$$\begin{aligned}
&\geq \sum_{i=0}^{\widehat{k}-1} (1-\pi)^i \pi (vt - \kappa\widehat{T} - p_1 - \kappa(i+1)) \\
&= \sum_{i=0}^{\widehat{k}-1} (1-\pi)^i \pi (vt - \kappa\widehat{T} - p_1 - \kappa i) - \kappa \sum_{i=0}^{\widehat{k}-1} (1-\pi)^i \pi \geq \widetilde{U}^*(\widehat{T}, 1 \mid p_1) - \kappa
\end{aligned}$$

Note also that $\widetilde{U}^*(\widehat{T}+1, 1 \mid p_2) = \max\{\widetilde{U}^*(\widehat{T}, 1 \mid p_2) - \kappa, 0\}$. Therefore, when $\widetilde{U}^*(\widehat{T}+1, 1 \mid p_2) > 0$, we have $\widetilde{U}^*(\widehat{T}+1, 1 \mid p_2) = \widetilde{U}^*(\widehat{T}, 1 \mid p_2) - \kappa$.

If the optimal sub-strategy in residual state $(\widehat{T}, 1)$ starts with p_1, then by definition $\widetilde{U}^*(\widehat{T}, 1 \mid p_2) < \widetilde{U}^*(\widehat{T}, 1 \mid p_1)$, which implies $\widetilde{U}^*(\widehat{T}, 1 \mid p_2) - \kappa < \widetilde{U}^*(\widehat{T}, 1 \mid p_1) - \kappa$. Hence, when$\widetilde{U}^*(\widehat{T}+1, 1 \mid p_2) > 0$, we have $\widetilde{U}^*(\widehat{T}+1, 1 \mid p_2) < \widetilde{U}^*(\widehat{T}, 1 \mid p_1) - \kappa \leq \widetilde{U}^*(\widehat{T}+1, 1 \mid p_1)$. This establishes that if the optimal sub-strategy starts bidding at p_1, it will never switch to bidding at p_2. □

Based on Lemma 33, the optimal sub-strategy $\Gamma^*(1|\widehat{T},\widehat{P})$ at $(\widehat{T},1,\widehat{P})$ can be obtained using Eqs. (A.17), (A.18), and (A.19), namely

$$\Gamma^*(1|\widehat{T},\widehat{P}) = \begin{cases} \text{bid at } p_1 & \text{if } \widetilde{U}_1^{(\widehat{k})}(\widehat{T},1) > \widetilde{U}_2(\widehat{T},1) \\ & \text{and } \widetilde{U}_1^{(\widehat{k})}(\widehat{T},1) > 0 \\ \text{bid at } p_2 & \text{if } \widetilde{U}_2(\widehat{T},1) \geq \widetilde{U}_1^{(\widehat{k})}(\widehat{T},1) \\ & \text{and } \widetilde{U}_2(\widehat{T},1) > 0 \\ \text{terminate} & \text{otherwise} \end{cases}$$

Next, we turn to the other base strategy we need to identify to be in a position to formulate our recurrence, namely, the strategy when $\widehat{T}$ is "large enough." Specifically, in our case "large enough" means $\widehat{T} > \frac{vt}{\kappa}$ so that $\kappa\widehat{T} \geq vt$, *i.e.,* the delay penalty exceeds the job value. Under this condition, the expected residual utility cannot be positive, and the optimal strategy is always to terminate. Note that while $\frac{vt}{\kappa}$ can be larger than the optimal termination slot, it still gives a valid strategy on which to base our recurrence as we describe next.

Recurrence Our goal in this next step is to formulate a recurrence, *i.e.,* to characterize step 2 in the backward induction algorithm, starting with the two base strategies identified for $\widehat{t} = 1$ and $\widehat{T} > \frac{vt}{\kappa}$, which will allow us to compute the optimal (dynamic) bidding strategy at any job's state, where the goal is to maximize expected residual utility (and therefore expected utility) or terminate if it is not positive.

Recall that at each step users can either bid at p_1, or bid at p_2, or terminate if neither price generates a positive expected residual utility. Consider then a job with residual state $(\widehat{T},\hat{t})$. If the job bids at p_1, it will either enter residual state $(\widehat{T},\hat{t}-1)$ and pay p_1 if the bid wins, or enter residual state $(\widehat{T}+1,\hat{t})$ if the bid fails. Whereas if the job bids at p_2, the job will

enter residual state $(\widehat{T}, \hat{t}-1)$ and pay $(\pi p_1 + (1-\pi)p_2)$ on average. Hence we have

$$\widetilde{U}^*(\widehat{T}, \hat{t}) = \max\Big\{0, \pi(\widetilde{U}^*(\widehat{T}, \hat{t}-1) - p_1) + (1-\pi)\widetilde{U}^*(\widehat{T}+1, \hat{t}), \\ \widetilde{U}^*(\widehat{T}, \hat{t}-1) - (\pi p_1 + (1-\pi)p_2)\Big\}.$$

which establishes the desired recurrence to compute which of the three possible actions yields the best residual utility, and therefore allows us to identify the optimal strategy at any residual state $(\widehat{T}, \hat{t})$.

A.7.2 Case 2: Convex Delay Sensitivity

In this sub-section we characterize the optimal bidding strategy when jobs have a piece-wise linear, convex delay sensitivity function. Specifically, we split the bidding strategy into two parts: before and after $\widehat{T} = \theta$. We first show that under $D_1(\kappa, t)$, the optimal bidding strategy for job (t, v, κ) keeps bidding at $b^*(v, \kappa)$ after $\widehat{T} = \theta$, where $b^*(v, \kappa)$ is the optimal bid for $(1, v, \kappa)$ jobs under a linear delay penalty. We then construct the optimal bidding strategy before $\widehat{T} = \theta$ through backward induction and a corresponding MDP.

Optimal substrategy for $\widehat{T} \geq \theta$. When $\widehat{T} \geq \theta$, the expected residual utility of (t, v, κ) jobs in state $(\widehat{T}, \widehat{t}, \widehat{P})$ given sub-strategy Γ is of the form:

$$\widetilde{U}_\Gamma(\widehat{T}, \widehat{t}) = vt - P(\widehat{t}, \Gamma) - \kappa[(\widehat{T} - \theta) + T(\widehat{t}, \Gamma)],$$

where $P(\widehat{t}, \Gamma)$ is the residual cost of completing the job under strategy Γ, and $T(\widehat{t}, \Gamma)$ is the expected delay for a job of size $\widehat{t}$ under strategy Γ. This can be rewritten as

$$\widetilde{U}_\Gamma(\widehat{T}, \widehat{t}) = \left[\frac{vt - \kappa(\widehat{T} - \theta)}{\widehat{t}}\right]\widehat{t} - P(\widehat{t}, \Gamma) - \kappa T(\widehat{t}, \Gamma).$$

The above expression is of a similar form as Eq. (2.1), so that Proposition 1 implies that a fixed bidding strategy maximizes the (residual) expected utility. Similarly, from Proposition 2, the optimal bidding strategy bids at p_1 if $\pi(p_2 - p_1) > \kappa$, and bids at p_2 otherwise.

MDP Construction Given that the optimal bidding strategy after $\widehat{T} = \theta$ is known, we turn next to the bidding strategy before $\widehat{T} = \theta$. For each $\widehat{t} \leq t$ we construct an aggregating state $S_{\widehat{t}}$, with a job transitioning to $S_{\widehat{t}}$ whenever it reaches $\widehat{T} = \theta$ with a residual job size of $\widehat{t}$. If a job (t, v, κ) reaches $S_{\widehat{t}}$, it gets a reward of $-C^*(\widehat{t}, \kappa)$, the expected total cost (payment plus delay penalty) for a job $(\widehat{t}, v, \kappa)$ under a linear delay penalty.

We characterize job (t, v, κ)'s bidding strategy using four types of states: start state (S_0), bid states ($S_{\widehat{T},\widehat{t}}$) for $\widehat{T} < \theta$, aggregating states ($S_{\widehat{t}}$) for $1 \leq \widehat{t} \leq t$, and finish state ($S_f$), where $S_{\widehat{t}}$ and S_f are absorbing states. A job starts in S_0, and goes to state $S_{0,t}$ with a reward of vt if it decides to adopt the service. Otherwise, it exits and goes to state S_f with a reward of 0. In state $S_{\widehat{T},\widehat{t}}$, a job bids with a value in $\{p_1, p_2\}$. If the job bids at p_2, it will go to state $S_{\widehat{T},\widehat{t}-1}$ with an expected reward of $-\pi p_1 - (1-\pi)p_2$. If the job bids at p_1, it will go to state $S_{\widehat{T},\widehat{t}-1}$ with a reward of $-p_1$ with probability π, and will go to state $S_{\widehat{T}+1,\widehat{t}}$ (or state $S_{\widehat{t}}$ when $\widehat{T} + 1 = \theta$) with a reward of 0 (or $-C^*(\widehat{t}, \kappa)$ when $\widehat{T} + 1 = \theta$) with probability $(1 - \pi)$. Formally:

Decision epochs:

$$T = \{1, 2, ..., \theta - 1\}.$$

States:

$$S = \{S_0,\ S_{\widehat{T},\widehat{t}},\ S_{\widehat{t}},\ S_f\}, \qquad \text{where } \widehat{T} < \theta, \widehat{t} \leq t, \text{ and } S_{\widehat{T},0} := S_f.$$

Actions:

$$A_0 = \{adopt, exit\}; \qquad A_{\widehat{T},\widehat{t}} = \{p_1, p_2\}.$$

Expected rewards:

$$r(adopt, S_0) = vt; \qquad r(exit, S_0) = 0; \qquad r(p_2, S_{\widehat{T},\widehat{t}}) = -\pi p_1 - (1-\pi)p_2;$$

$$r(p_1, S_{\widehat{T},\widehat{t}}) = -\pi p_1, \text{ when } \widehat{T} < \theta - 1; \qquad r(p_1, S_{\theta-1,\widehat{t}}) = -\pi p_1 - (1-\pi)C^*(\widehat{t}, \kappa).$$

Transition probabilities (all omitted transitions, except from absorbing states to themselves, have a probability of 0):

$$Prob(S_{0,t}|S_0, adopt) = 1; \qquad Prob(S_f|S_0, exit) = 1;$$

$$Prob(S_{\widehat{T},\widehat{t}-1}|S_{\widehat{T},\widehat{t}}\ ,\ p_1) = \pi; \qquad Prob(S_{\widehat{T},\widehat{t}-1}|S_{\widehat{T},\widehat{t}}\ ,\ p_2) = 1;$$

$$Prob(S_{\widehat{T}+1,\widehat{t}}|S_{\widehat{T},\widehat{t}}\ ,\ p_1) = 1-\pi, \text{ when } \widehat{T} < \theta - 1; \qquad Prob(S_{\widehat{t}}|S_{\theta-1,\widehat{t}}\ ,\ p_1) = 1-\pi.$$

Base strategies ($\widehat{t} = 1$) As before, we introduce a lemma that establishes general properties of the optimal bidding strategy for jobs with $\widehat{t} = 1$. Although the optimal bidding strategy when $\widehat{T} \geq \theta$ is already known, we include it in Lemma 34 for completeness and consistency with our earlier formulation (Lemma 33).

Lemma 34. *Under a two-price pricing system with jobs (t, v, κ) having a delay sensitivity of the form*

$$D_1(\kappa, t) = \kappa\ \max\{0, T(t) - \theta\},$$

the optimal bidding strategy for a job in state $(\widehat{T}, 1, \widehat{P})$ is to bid at p_2 if $\widehat{T} \geq \theta$ and $\pi(p_2 - p_2) \leq \kappa$, or bid at p_1 otherwise.

Proof. Reusing earlier notation to denote the expected residual utility generated by the strategy that switches to p_2 after l consecutive failed bids at p_1.

$$\begin{aligned}\widetilde{U}^{(l)}(\widehat{T},1,\widehat{P}) =& vt-\widehat{P}-\sum_{i=0}^{l-1}(1-\pi)^i\pi[p_1+\kappa\max\{0,\widehat{T}+i-\theta\}]\\ &-(1-\pi)^l[\pi p_1+(1-\pi)p_2+\kappa\max\{0,\widehat{T}+l-\theta\}].\end{aligned}$$

Next, we prove the following three properties

P1) bidding at p_2 is dominated by $\Gamma(\theta-\widehat{T})$, *i.e.,* bidding at p_1 until the delay reaches θ, and then switching to p_2;

P2) when $\widehat{T}+l<\theta$, $\Gamma(l)$ is dominated by $\Gamma(\theta-\widehat{T})$;

P3) when $\widehat{T}\geq\theta$, the optimal strategy bids at p_1 if $\pi(p_2-p_1)>\kappa$, and bids at p_2 if $\pi(p_2-p_1)\leq\kappa$.

Combining **P1)** and **P2)** implies that when $\widehat{T}\leq\theta$, bidding at p_1 is optimal, while **P3)** states that switching to bidding at p_2 once $\widehat{T}\geq\theta$ is optimal only if $\pi(p_2-p_1)\leq\kappa$.

To prove **P1)**, we compare the expected residual utility generated by $\Gamma(\theta-\widehat{T})$ to that of bidding at p_2. The expected residual utility for $\Gamma(\theta-\widehat{T})$ is

$$\widetilde{U}^{(\theta-\widehat{T})}(\widehat{T},1,\widehat{P})=vt-\widehat{P}-\sum_{i=0}^{\theta-\widehat{T}-1}(1-\pi)^i\pi p_1-(1-\pi)^{\theta-\widehat{T}}[\pi p_1+(1-\pi)p_2].$$

Denote the expected residual utility for bidding at p_2 while $\widehat{T}<\theta$ as $U^{(0)}(\widehat{T},1,\widehat{P})$, we have

$$\widetilde{U}^{(0)}(\widehat{T},1,\widehat{P})=vt-\widehat{P}-\pi p_1-(1-\pi)p_2.$$

Basic algebraic manipulations yield

$$\widetilde{U}^{(\theta-\widehat{T})}(\widehat{T},1,\widehat{P})-\widetilde{U}^{(0)}(\widehat{T},1,\widehat{P})=\pi p_1[1-\sum_{i=0}^{\theta-\widehat{T}}(1-\pi)^i]+(1-\pi)p_2[1-(1-\pi)^{\theta-\widehat{T}+1}]>0.$$

This establishes that $\Gamma(\theta-\widehat{T})$ dominates $\Gamma^{(0)}$, *i.e.,* bidding at p_2 (when $\widehat{T}<\theta$).

Next we turn to **P2)** and show that if $\widehat{T}+l<\theta$, $\Gamma(\theta-\widehat{T})$ dominates $\Gamma(l)$. To do so, we prove that $\widetilde{U}^{(l)}(\widehat{T},1,\widehat{P})$ increases with l when $\widehat{T}+l\leq\theta$. Basic algebraic manipulations give

$$\widetilde{U}^{(l+1)}(\widehat{T},1,\widehat{P})-\widetilde{U}^{(l)}(\widehat{T},1,\widehat{P})=\pi(1-\pi)^{l+1}(p_2-p_1)>0,$$

which establishes that $\widetilde{U}^{(l)}(\widehat{T},1,\widehat{P})$ increases with l when $\widehat{T}+l\leq\theta$.

Finally, we turn to **P3)** and characterize the optimal bidding strategy when the delay exceeds θ, where for simplicity of notation we denote as $\widetilde{U}^{(l)}$ the residual utility $\widetilde{U}^{(l)}(\widehat{T},1,\widehat{P})$ of the strategy that bids at p_1 for $l\geq 0$ slots before switching to p_2. When $\widehat{T}\geq\theta$, we have

$$\widetilde{U}^{(l)}=vt-\widehat{P}-\sum_{i=0}^{l-1}(1-\pi)^i\pi[p_1+\kappa(\widehat{T}+i-\theta)]-(1-\pi)^l[\pi p_1+(1-\pi)p_2+\kappa(\widehat{T}+l-\theta)].$$

This gives

$$\widetilde{U}^{(l+1)}-\widetilde{U}^{(l)}=(1-\pi)^{l+1}[\pi(p_2-p_1)-\kappa],$$

which is positive when $\pi(p_2-p_1)>\kappa$.

When $\widetilde{U}^{(l+1)}-\widetilde{U}^{(l)}$ is positive, $\widetilde{U}^{(l)}$ increases with l, which when combined with **P1)** implies that continuing to bid at p_1 once $\widehat{T}\geq\theta$ remains optimal. Conversely, when $\widetilde{U}^{(l+1)}-\widetilde{U}^{(l)}$ is negative, $\widetilde{U}^{(l)}$ decreases with l once $\widehat{T}\geq\theta$, so that $\widetilde{U}^{(0)}>\widetilde{U}^{(l)}$. In other words, the expected residual utility under the bidding strategy that switches to bidding at p_2 once $\widehat{T}$ reaches θ is higher than that of any strategy that switches at a latter time. This establishes **P3)**. □

Lemma 34 specifies the optimal base bidding strategy for jobs for which $\widehat{t} = 1$, *i.e.,* they keep bidding at p_1 as long as $\widehat{T} \leq \theta$, and then keep bidding at p_1 if $\pi(p_2 - p_1) > \kappa$, or switch to p_2 if $\pi(p_2 - p_1) \leq \kappa$.

Recurrence In this scenario, possible actions are limited to bidding at p_1 or p_2. Consider a job with residual state $(\widehat{T}, \widehat{t})$. If the job bids at p_1, it will enter state $(\widehat{T}, \widehat{t}-1)$ and generate a payment of p_1 if its bid wins, or enter state $(\widehat{T}+1, \widehat{t})$ with no payment if its bid fails. Conversely, if the job bids at p_2, it will enter state $(\widehat{T}, \widehat{t}-1)$ with an expected payment of $(\pi p_1 + (1-\pi)p_2)$. This gives the following expression for the job's optimal expected residual utility:

$$\widetilde{U}^*(\widehat{T}, \hat{t}) = \max\Big\{\pi(\widetilde{U}^*(\widehat{T}, \hat{t}-1) - p_1) + (1-\pi)\widetilde{U}^*(\widehat{T}+1, \hat{t}),$$
$$\widetilde{U}^*(\widehat{T}, \hat{t}-1) - (\pi p_1 + (1-\pi)p_2)\Big\},$$

which again establishes the desired recurrence to identify the strategy that yields the best residual utility.

A.7.3 Case 3: Concave Delay Sensitivity

We turn next to the case of a concave delay penalty function. The approach parallels that followed for a convex delay penalty, namely, we first identify the optimal substrategy beyond $\widehat{T} = \theta$, and then construct the optimal bidding strategy before $\widehat{T} = \theta$ through backward induction and a corresponding MDP.

Optimal substrategy for $\widehat{T} \geq \theta$ Since under a concave delay sensitivity there is no further delay penalty once $\widehat{T} = \theta$, the optimal strategy will keep bidding at p_1 from thereon with a cumulative reward of $-\widehat{t}p_1$.

MDP Construction As with the convex case of Appendix A.7.2, we construct for each value of $\widehat{t} \leq t$ an aggregating state $S_{\widehat{t}}$, with a job transitioning to $S_{\widehat{t}}$ whenever it reaches $\widehat{T} = \theta$ with a residual job size of $\widehat{t}$. If a job (t, v, κ) reaches $S_{\widehat{t}}$, it gets a reward of $-\widehat{t}p_1$.

We characterize job (t, v, κ)'s bidding strategy using four types of states: start state (S_0), bid states ($S_{\widehat{T},\widehat{t}}$) for $\widehat{T} < \theta$, aggregating states ($S_{\widehat{t}}$) for $1 \leq \widehat{t} \leq t$, and finish state ($S_f$), where $S_{\widehat{t}}$ and S_f are absorbing states. A job starts in S_0, and goes to state $S_{0,t}$ with a reward of vt if it decides to adopt the service. Otherwise, it exits and goes to state S_f with a reward of 0. In state $S_{\widehat{T},\widehat{t}}$, a job bids with a value in $\{p_1, p_2\}$. If the job bids at p_2, it will go to state $S_{\widehat{T},\widehat{t}-1}$ with an expected reward of $-\pi p_1 - (1-\pi)p_2$. If the job bids at p_1, it will go to state $S_{\widehat{T},\widehat{t}-1}$ with a reward of $-p_1$ with probability π, and will go to state $S_{\widehat{T}+1,\widehat{t}}$ (or state $S_{\widehat{t}}$ when $\widehat{T}+1 = \theta$) with a reward of $-\kappa$ (or $(-\kappa - \widehat{t}p_1)$ when $\widehat{T}+1 = \theta$) with a probability of $(1-\pi)$. Formally:

Decision epochs:

$$T = \{1, 2, ..., \theta - 1\}.$$

States:

$$S = \{S_0,\ S_{\widehat{T},\widehat{t}},\ S_{\widehat{t}},\ S_f\}, \qquad \text{where } \widehat{T} < \theta, \widehat{t} \leq t, \text{ and } S_{\widehat{T},0} := S_f.$$

Actions:

$$A_0 = \{adopt, exit\}; \qquad A_{\widehat{T},\widehat{t}} = \{p_1, p_2\}.$$

Expected rewards:

$$r(adopt, S_0) = vt; \qquad r(exit, S_0) = 0; \qquad r(p_2, S_{\widehat{T},\widehat{t}}) = -\pi p_1 - (1-\pi)p_2;$$

$$r(p_1, S_{\widehat{T},\widehat{t}}) = -\pi p_1 - (1-\pi)\kappa, \text{ when } \widehat{T} < \theta - 1;$$

$$r(p_1, S_{\theta-1,\widehat{t}}) = -\pi p_1 - (1-\pi)(\kappa + \widehat{t}p_1).$$

Transition probabilities (all omitted transitions, except from absorbing states to themseleves, have a probability of 0):

$$Prob(S_{0,t}|S_0, adopt) = 1; \qquad Prob(S_f|S_0, exit) = 1;$$

$$Prob(S_{\widehat{T},\widehat{t}-1}|S_{\widehat{T},\widehat{t}}\ ,\ p_1) = \pi; \qquad Prob(S_{\widehat{T},\widehat{t}-1}|S_{\widehat{T},\widehat{t}}\ ,\ p_2) = 1;$$

$$Prob(S_{\widehat{T}+1,\widehat{t}}|S_{\widehat{T},\widehat{t}}\ ,\ p_1) = 1-\pi, \text{ when } \widehat{T} < \theta - 1; \qquad Prob(S_{\widehat{t}}|S_{\theta-1,\widehat{t}}\ ,\ p_1) = 1-\pi.$$

Base Strategies ($\widehat{t} = 1$) As with Lemma 34, for completeness and consistency, we again include the optimal bidding strategy for $\widehat{T} \geq \theta$ in Lemma 35.

Lemma 35. *Under a two-price pricing system with jobs (t, v, κ) having a delay sensitivity of the form*

$$D_2(\kappa, t) = \kappa\ \min\{T(t), \theta\},$$

the optimal bidding strategy for a job in state $(\widehat{T}, 1, \widehat{P})$ is to bid at p_2 if $\widehat{T} < \theta$ and $p_2 - p_1 < \kappa\theta(1-\pi)^{\theta-\widehat{T}-1} + \sum_{i=1}^{\theta-\widehat{T}-1} \kappa\pi(1-\pi)^{i-1}(i+\widehat{T}) - \kappa\widehat{T}$, or to bid at p_1 otherwise.

Proof. Using again the same notation as with a convex delay penalty, the expected residual utility of a job in state $(\widehat{T}, 1, \widehat{P})$ and using strategy $\Gamma^{(l)}$ that bids at p_1 for $l \geq 0$ slots and then switches to bidding at p_2 is of the form

$$\widetilde{U}^{(l)}(\widehat{T}, 1, \widehat{P}) = vt - \widehat{P} - \sum_{i=0}^{l-1}(1-\pi)^i\pi[p_1 + \kappa\min\{\theta, \widehat{T}+i\}]$$
$$-(1-\pi)^l[\pi p_1 + (1-\pi)p_2 + \kappa\min\{\theta, \widehat{T}+l\}],$$

where as before, the summation has a value of 0 when $l = 0$. Basic algebraic manipulations give

$$\widetilde{U}^{(l+1)}(\widehat{T}, 1, \widehat{P}) - \widetilde{U}^{(l)}(\widehat{T}, 1, \widehat{P}) = (1-\pi)^{l+1}[\pi(p_2 - p_1) + \kappa \min\{\theta, \widehat{T} + l\} - \kappa \min\{\theta, \widehat{T} + l + 1\}].$$

Next, we consider separately the two regimes $\widehat{T} + l \geq \theta$ and $\widehat{T} + l < \theta$. When $\widehat{T} + l \geq \theta$, we have

$$\widetilde{U}^{(l+1)}(\widehat{T}, 1, \widehat{P}) - \widetilde{U}^{(l)}(\widehat{T}, 1, \widehat{P}) = (1-\pi)^{l+1}\pi(p_2 - p_1) > 0.$$

In other words, when $\widehat{T}+l \geq \theta$, increasing l increases the expected residual utility of strategy $\Gamma^{(l)}$, so that if bidding at p_1 is the best strategy, it remains so until the bid succeeds and switching to p_2 never happens.

When $\widehat{T} + l < \theta$, we have

$$\widetilde{U}^{(l+1)}(\widehat{T}, 1, \widehat{P}) - \widetilde{U}^{(l)}(\widehat{T}, 1, \widehat{P}) = (1-\pi)^{l+1}[\pi(p_2 - p_1) - \kappa].$$

The sign of the above difference is a function of $\pi(p_2 - p_1) - \kappa$ and independent of l. If the difference is positive, we are in the same situation as when $\widehat{T} + l \geq \theta$, and switching from p_1 to p_2 never happens. Conversely, if the difference is negative, *i.e.,* $\pi(p_2 - p_1) - \kappa < 0$ the strategy $\Gamma^{(0)}$, *i.e.,* immediately bidding at p_2 outperforms all strategies with higher l values. In other words, fixed strategies are always optimal once jobs enter state $(\widehat{T}, 1, \widehat{P})$. Next, we identify which of bidding at p_1 or p_2 is optimal under what conditions.

We first note that when $\widehat{T} \geq \theta$, bidding at p_1 is trivially optimal, since under the concave delay penalty $D_2(\kappa, t)$, jobs incur no additional penalty once their execution delay exceeds θ. We therefore focus on the case $\widehat{T} < \theta$, and compute the residual utilities $\widetilde{U_1}(\widehat{T}, 1, \widehat{P})$ and

$\widetilde{U_2}(\widehat{T}, 1, \widehat{P})$ of the strategies that bid at p_1 and p_2, respectively.

$$\widetilde{U}_1(\widehat{T}, 1, \widehat{P}) = vt - \widehat{P} - p_1 - \sum_{i=0}^{\theta-\widehat{T}-1} \kappa\pi(1-\pi)^i(i+\widehat{T}) - \kappa\theta(1-\pi)^{\theta-\widehat{T}},$$
$$\widetilde{U}_2(\widehat{T}, 1, \widehat{P}) = vt - \widehat{P} - \kappa\widehat{T} - \pi p_1 - (1-\pi)p_2.$$

Basic algebraic manipulations give $\widetilde{U_1}(\widehat{T}, 1, \widehat{P}) < \widetilde{U_2}(\widehat{T}, 1, \widehat{P})$ iff

$$p_2 - p_1 < \kappa\theta(1-\pi)^{\theta-\widehat{T}-1} + \sum_{i=1}^{\theta-\widehat{T}-1} \kappa\pi(1-\pi)^{i-1}(i+\widehat{T}) - \kappa\widehat{T},$$

where the summation is zero if $\theta - \widehat{T} - 1 < 1$. □

Lemma 35 characterizes the optimal base bidding strategy for jobs for which $\widehat{t} = 1$: they bid at p_2 if $\widehat{T} < \theta$ and the difference between p_2 and p_1 is below a certain threshold, and keep bidding at p_1 otherwise.

Recurrence The recurrence is essentially identical to that of the convex delay penalty, namely,

$$\widetilde{U}^*(\widehat{T}, \widehat{t}) = \max\Big\{\pi(\widetilde{U}^*(\widehat{T}, \widehat{t}-1) - p_1) + (1-\pi)\widetilde{U}^*(\widehat{T}+1, \widehat{t}),$$
$$\widetilde{U}^*(\widehat{T}, \widehat{t}-1) - (\pi p_1 + (1-\pi)p_2)\Big\}.$$

Appendix B

Traffic Scheduling and Shaping for Inter-data Center Networks

B.1 Summary of Notation Used in the Chapter

Notation	Definition
n	number of flows inside the network
b_i	bucket size of flow i
$\boldsymbol{b}$	vector for all token-bucket sizes: $(b_1, b_2, \ldots, b_n)$
b'_i	bucket size of flow i's token-bucket reshaper
$\boldsymbol{b'}$	vector for all reshapers' bucket sizes: $(b'_1, b'_2, \ldots, b'_n)$
b'^*_i	the optimal bucket size of flow i's token-bucket reshaper under static priority
$\boldsymbol{b'}^*$	vector for all reshapers' optimal bucket sizes: $(b'^*_1, b'^*_2, \ldots, b'^*_n)$ under static priority
$\widehat{b}'^*_i$	the optimal bucket size of flow i's token-bucket reshaper under fifo
$\widehat{\boldsymbol{b}}'^*$	vector for all reshapers' optimal bucket sizes: $(\hat{b}'^*_1, \hat{b}'^*_2, \ldots, \hat{b}'^*_n)$ under fifo
B'_i	accumulative shaper size for flows with a priority no smaller than i, *i.e.,* $\sum_{j=i}^{n} b'_j$
$\widehat{B}_i$	accumulative burst size for flows 1 to i, *i.e.,* $\sum_{j=1}^{i} b_j$
$\widehat{B}'_i$	accumulative shaper size for flows from 1 to i, *i.e.,* $\sum_{j=1}^{i} b'_j$
d_i	end-to-end deadline for flow i
$\boldsymbol{d}$	vector for all end-to-end deadlines: $(d_1, d_2, \ldots, d_n)$
D^*_i	worst-case end-to-end delay for flow i under priority + shaping
$\widehat{D}^*_i$	worst-case end-to-end delay for flow i under fifo + shaping
r_i	rate of flow i
$\boldsymbol{r}$	vector for all token-bucket rates: $(r_1, r_2, \ldots, r_n)$
(r_i, b_i, d_i)	flow i's profile
R_i	accumulative shaper size for flows with a priority no smaller than i, *i.e.,* $\sum_{j=i}^{n} r_j$
R	bandwidth of the shared link for the one-hop scenario
R^*	optimal minimum required bandwidth for the one-hop scenario

$\widetilde{R}^*$	minimum required bandwidth for the shared link without (re)shapers under static priority for one-hop scenario
$\widetilde{R}_s^*$	minimum required bandwidth for the shared link with (re)shapers under static priority for the one-hop scenario
$\widehat{R}^*$	minimum required bandwidth for the shared link without (re)shapers under fifo for one-hop scenario
$\widehat{R}_s^*$	minimum required bandwidth for the shared link with (re)shapers under fifo for the one-hop scenario
t	time
H_i	$b_i - d_i r_i$
$\Pi_i(R)$	$\frac{r_i+R-R_{i+1}}{R-R_{i+1}}$
$V_i(R)$	$d_i(R - R_{i+1} - b_i)$
$\mathbb{S}_1(R)$	$V_1(R)$
$\mathbb{S}_i(r)$	$\mathbb{S}_{i-1}(R) \bigcup V_i(R) \bigcup \left\{ \frac{s-H_i}{\Pi_i(R)} \vert s \in \mathbb{S}_{i-1}(R) \right\}$
$\mathbb{Z}_i$	all the integers from 1 to i, *i.e.*, $\{1 \leq i \leq j \vert j \in \mathbb{Z}\}$
$X_F(R)$	$\max_{P_1,P_2 \subseteq \mathbb{Z}_n, P_2 \neq \mathbb{Z}_n, P_1 \bigcap P_2 = \emptyset} \frac{\sum_{i \in P_1} \frac{RH_i}{R+r_i} + \sum_{i \in P_2} \left(b_i - \frac{r_i d_i R}{R_1} \right)}{1 - \sum_{i \in P_1} \frac{r_i}{R+r_i} - \sum_{i \in P_2} \frac{r_i}{R_1}}$
$Y_F(R)$	$\min_{1 \leq i \leq n-1} \left\{ \widehat{B}_n, Rd_n, \min_{P_1,P_2 \subseteq \mathbb{Z}_i, P_1 \bigcap P_2 = \emptyset, P_1 \bigcup P_2 \neq \emptyset} \left\{ \frac{\widehat{B}_i - \sum_{j \in P_1} \frac{RH_j}{R+r_j} - \sum_{j \in P_2} \left(b_j - \frac{r_j d_j R}{R_1} \right)}{\sum_{j \in P_1} \frac{r_j}{R+r_j} + \sum_{j \in P_2} \frac{r_j}{R_1}} \right\} \right\}$
$T_i(\widehat{B}'_n, R)$	$\max \left\{ 0, \frac{R}{R+r_i} \left(H_i + \frac{r_i}{R} \widehat{B}'_n \right), b_i + \frac{r_i(\widehat{B}'_n - Rd_i)}{R_1} \right\}$
Γ_{sc}	a service curve assignment that gives each flow i a service curve of $SC_i(t)$
OPT	general network-wide optimization
OPT_S	optimization for static priority with (re)shapers for the one-hop scenario
OPT_F	optimization for fifo with (re)shapers for the one-hop scenario

B.2 Proofs for Dynamic Priority Scheduler

B.2.1 Proof for Proposition 10

For the reader's convenience, we restate Proposition 10.

PROPOSITION 10. *Consider a one-hop network shared by n token-bucket controlled flows, where flow $i, 1 \leq i \leq n$, has a traffic contract of (r_i, b_i) and a deadline of d_i, with $d_1 > d_2 > ... > d_n$ and $d_1 < \infty$. Consider a service-curve assignment Γ_{sc} that allocates flow i a service curve of*

$$SC_i(t) = \begin{cases} 0 & \text{when } t < d_i, \\ b_i + r_i(t - d_i) & \text{otherwise.} \end{cases} \tag{3.2}$$

Then

1. *For any flow $i, 1 \leq i \leq n$, $SC_i(t)$ ensures a worst-case end-to-end delay no larger than d_i.*

2. *Realizing Γ_{sc} requires a link bandwidth of at least*

$$R^* = \max_{1 \leq h \leq n} \left\{ \sum_{i=1}^{n} r_i, \frac{\sum_{i=h}^{n} b_i + r_i(d_h - d_i)}{d_h} \right\}. \tag{3.3}$$

3. *Any scheduling mechanism capable of meeting all the flows' deadlines requires a bandwidth of at least R^*.*

Proof. We first show that Γ_{sc} meets each flow's deadline, and then show that a bandwidth of R^* is enough to accommodate all the service curves defined in Γ_{sc}. After that, we show

that there exists no mechanism generating a minimum required bandwidth strictly smaller than R^*.

- For any flow $1 \leq i \leq n$, it has an token-bucket constrained arrival curve of

$$AC_i(t) = \begin{cases} 0 & \text{when } t = 0 \\ b_i + r_i t & \text{otherwise.} \end{cases}$$

  Combining it with flow i's service curve $SC_i(t)$, we have the worst-case end-to-end delay for flow i [58],

$$D_i^* = \sup_{t \geq 0} \inf_{\tau \geq 0} \{AC_i(t) \leq SC_i(t+\tau)\} = d_i.$$

- To accommodate all the service curves defined in Γ_{sc}, the system needs a bandwidth R such that 1) $R \geq \sum_{i=1}^{n} r_i$, which guarantees a finite worst-case end-to-end delay for any flow, and 2) for all $t > 0$, $Rt \geq \sum_{i=1}^{n} SC_i(t)$, *i.e.,*

$$R \geq \max \left\{ \sum_{i=1}^{n} r_i, \ \sup_{t>0} \frac{\sum_{i=1}^{n} SC_i(t)}{t} \right\}. \tag{B.1}$$

  As $\frac{SC_i(t)}{t}$ equals 0 when $t < d_i$, and decreases t when $t \geq d_i$, we know that the supremum of $\frac{\sum_{i=1}^{n} SC_i(t)}{t}$ is achieved only at values among $\{d_1, d_2, \ldots, d_n\}$. Combining this with the expression for $SC_i(t)$ gives

$$R \geq \max_{1 \leq h \leq n} \left\{ \sum_{i=1}^{n} r_i, \ \frac{\sum_{i=h}^{n} b_i + r_i(d_h - d_i)}{d_h} \right\} = R^*.$$

  Thus, R^* is enough to accommodates all service curves defined in Γ_{sc}.

[58] See THEOREM 1.4.2 in [120], page 23.

- We show that R^* is a lower bound of the minimum required bandwidth. Note that when $R^* = \sum_{i=1}^{n} r_i$, obviously there exists no mechanism achieving a strictly smaller minimum required bandwidth. Below we consider the case when $R^* > \sum_{i=1}^{n} r_i$, *i.e.*, there exists $1 \leq \hat{h} \leq n$, such that $R^* = \frac{\sum_{i=\hat{h}}^{n} b_i + r_i(d_{\hat{h}} - d_i)}{d_{\hat{h}}}$

  Suppose there exists a mechanism achieving a minimum required bandwidth $R' < R^*$. Next we construct an arrival pattern consistent with each flow's token-bucket arrival constraints, such that R' cannot satisfy all flows' deadlines.

  Consider the arrival pattern such that for all $1 \leq i \leq n$, flow i sends b_i at $t = 0$ (where the system restarts the clock), and then constantly sends at a rate of r_i. By time $d_{\hat{h}}$, in order to satisfy all flows' deadlines, for any flow i with $d_i \leq d_{\hat{h}}$ the shared link should process at least $b_i + r_i(d_{\hat{h}} - d_i)$ amount of data. Consequently, by $d_{\hat{h}}$ the shared link should process at least $\sum_{i=\hat{h}}^{n} b_i + r_i(d_{\hat{h}} - d_i)$ amount of data for all flows in accumulation. As $\sum_{i=\hat{h}}^{n} b_i + r_i(d_{\hat{h}} - d_i) = d_{\hat{h}} R^* > d_{\hat{h}} R'$, a bandwidth of R' must violate some flows' deadlines.

□

B.2.2 Proof for Proposition 11

Proposition 11. *Consider a one-hop network shared by n token-bucket controlled flows, where flow $i, 1 \leq i \leq n$, has a traffic contract of (r_i, b_i) and a deadline of d_i, with $d_1 > d_2 > ... > d_n$ and $d_1 < \infty$. The earliest deadline first (EDF) scheduler realizes Γ_{sc} under a link bandwidth of R^*.*

Proof. We first show that EDF satisfies Γ_{sc}, and then shows that EDF requires a minimum required bandwidth of R^*.

We show that EDF satisfies Γ_{sc} by contradiction. Suppose EDF cannot achieve Γ_{sc}. Then there exists $\hat{t} \geq d_i$ and $1 \leq i \leq n$, such that $\widehat{SC}_i(\hat{t}) < SC_i(\hat{t})$, where $\widehat{SC}_i$ is the service curve that EDF assigns to flow i. To satisfy flow i's deadline, at $\hat{t} - d_i$ EDF should yield a virtual delay no large than d_i, *i.e.,*

$$\inf_{\tau \geq 0} \left\{ b_i + r_i(\hat{t} - d_i) \leq \widehat{SC}_i(\hat{t} - d_i + \tau) \right\} \leq d_i,$$

which then gives $\widehat{SC}_i(\hat{t} - d_i + d_i) \geq b_i + r_i(\hat{t} - d_i)$. As $b_i + r_i(\hat{t} - d_i) = SC_i(\hat{t})$, this contradicts to the assumption that $\widehat{SC}_i(\hat{t}) < SC_i(\hat{t})$.

Below we show that EDF requires a minimum required bandwidth of R^*. Suppose flow i's data sent at t has a deadline of $(t + d_i)$. We show that EDF satisfies all flows' deadlines with a bandwidth of $R^* = \max_{1 \leq h \leq n} \left\{ \sum_{i=1}^{n} r_i, \frac{\sum_{i=h}^{n} b_i + r_i(d_h - d_i)}{d_h} \right\}$. Based on the utilization of the shared link, we consider two cases separately, where in both cases we prove the result by contradiction. Specifically, suppose under EDF, R^* cannot satisfy all flows' latency requirements. Then there exists $1 \leq \hat{h} \leq n$ and $\hat{t} \geq 0$, such that EDF processes at least one bit sent by flow $\hat{h}$ at time $\hat{t}$ after time $(\hat{t} + d_{\hat{h}})$. We consider first the case where the shared link uses up all its bandwidth during the period $[0, \hat{t} + d_{\hat{h}}]$ to transmit data with absolute deadlines no larger than $(\hat{t} + d_{\hat{h}})$, and then consider the case where there exists $t_0 \in [0, \hat{t} + d_{\hat{h}}]$, such that at t_0 the shared link is not busy with data whose absolute deadline is no larger than $(\hat{t} + d_{\hat{h}})$. By showing that R^* is enough for EDF to meet all flows' deadlines, we then have the result.

1. Consider the case where for all $t \in [0, \hat{t} + d_{\hat{h}}]$ the shared link uses up all its bandwidth to send bits with an absolute deadline no larger than $(\hat{t} + d_{\hat{t}})$. Then by $(\hat{t} + d_{\hat{h}})$ the shared link in accumulation processes $R^*(\hat{t} + d_{\hat{h}})$ amount of data that all have deadlines no larger than $(\hat{t} + d_{\hat{t}})$. From the fact that EDF violates an absolute deadline

of $(\hat{t}+d_{\hat{h}})$, we know that there exists an arrival pattern consistent with the token-bucket constraints, such that in accumulation flows send more than $R^*(\hat{t}+d_{\hat{h}})$ amount of data with absolute deadlines no larger than $(\hat{t}+d_{\hat{h}})$. From the token-bucket constraints, we know that by $(\hat{t}+d_{\hat{h}})$, flows can send at most $\sum_{i=1}^{n}\left[b_i+r_i(\hat{t}+d_{\hat{h}}-d_i)\right]\mathbf{I}_{\hat{\mathbf{t}}+\mathbf{d}_{\hat{\mathbf{h}}}-\mathbf{d}_\mathbf{i}\geq\mathbf{0}}$ amount of data whose absolute deadline is at most $(\hat{t}+d_{\hat{h}})$. Therefore, we have $R^*(\hat{t}+d_{\hat{h}})<\sum_{i=1}^{n}\left[b_i+r_i(\hat{t}+d_{\hat{h}}-d_i)\right]\mathbf{I}_{\hat{\mathbf{t}}+\mathbf{d}_{\hat{\mathbf{h}}}-\mathbf{d}_\mathbf{i}\geq\mathbf{0}}$. Define $d_0=\infty$. Then there exists $1\leq\hat{n}\leq n$ such that $\hat{t}+d_{\hat{h}}\in[d_{\hat{n}},d_{\hat{n}-1})$, then we have

$$R^*<\frac{\sum_{i=1}^{n}\left[b_i+r_i(\hat{t}+d_{\hat{h}}-d_i)\right]\mathbf{I}_{\hat{\mathbf{t}}+\mathbf{d}_{\hat{\mathbf{h}}}-\mathbf{d}_\mathbf{i}\geq\mathbf{0}}}{\hat{t}+d_{\hat{h}}}=\sum_{i=\hat{n}}^{n}\left(r_i+\frac{b_i-r_id_i}{\hat{t}+d_{\hat{h}}}\right):=R'.$$

If $\sum_{i=\hat{n}}^{n}b_i-r_id_i\leq 0$, we have $R^*<R'\leq\sum_{i=1}^{n}r_i$, which contradicts to $R^*>\sum_{i=1}^{n}r_i$. Hence we consider only $\sum_{i=\hat{n}}^{n}b_i-r_id_i>0$, where R' decreases with $(\hat{t}+d_{\hat{h}})$. Define $\hat{R}(u)=\sum_{i=\hat{n}}^{n}\left(r_i+\frac{b_i-r_id_i}{u}\right)$. We then have

$$R^*<R'\leq\hat{R}(d_{\hat{n}})=\frac{\sum_{i=\hat{n}}^{n}\left[b_i+r_i(d_{\hat{n}}-d_i)\right]}{d_{\hat{n}}},$$

which contradicts to the definition of R^*.

2. Otherwise, the shared link uses less than all its bandwidth at $t_0\in[0,\hat{t}+d_{\hat{h}}]$, and uses up all its bandwidth for all $t\in(t_0,\hat{t}+d_{\hat{h}}]$ to send bits with a absolute deadline no larger than $(\hat{t}+d_{\hat{t}})$. Then during $(t_0,\hat{t}+d_{\hat{h}}]$ the shared link processes $R^*(\hat{t}+d_{\hat{h}}-t_0)$ amount of data with absolute deadlines no larger than $(\hat{t}+d_{\hat{h}})$, and flows send strictly more than $R^*(\hat{t}+d_{\hat{h}}-t_0)$ amount of data with absolute deadlines no larger than $(\hat{t}+d_{\hat{h}})$. From the token-bucket constraints, we know that during $(t_0,\hat{t}+d_{\hat{h}}]$ flows can send at most $\sum_{i=1}^{n}\left[b_i+r_i(\hat{t}+d_{\hat{h}}-t_0-d_i)\right]\mathbf{I}_{\hat{\mathbf{t}}+\mathbf{d}_{\hat{\mathbf{h}}}-\mathbf{t_0}-\mathbf{d}_\mathbf{i}\geq\mathbf{0}}$ amount of data whose absolute deadlines are no larger than $(\hat{t}+d_{\hat{h}})$. Thus we have $R^*<\frac{\sum_{i=1}^{n}\left[b_i+r_i(\hat{t}+d_{\hat{h}}-t_0-d_i)\right]\mathbf{I}_{\hat{\mathbf{t}}+\mathbf{d}_{\hat{\mathbf{h}}}-\mathbf{t_0}-\mathbf{d}_\mathbf{i}\geq\mathbf{0}}}{d_{\hat{h}}+\hat{t}-t_0}$, which, similar as before, contradicts to the definition of R^*.

□

B.2.3 Proof for Proposition 12

PROPOSITION 12. *Consider a one-hop network shared by n token-bucket controlled flows, where flow $i, 1 \leq i \leq n$, has a traffic contract of (r_i, b_i) and a deadline of d_i, with $d_1 > d_2 > ... > d_n$ and $d_1 < \infty$. Adding ingress (re)shapers will not decrease the minimum bandwidth required to meet the flows' deadlines.*

Proof. We show that adding a (re)shaper to any of the flow does not decrease the optimal minimum required bandwidth.

Suppose we apply a reshaper b'_i to flow i, where $1 \leq i \leq n$ and $b'_i \geq b_i - d_i r_i$. Then flow i incurs a reshaping delay of $\frac{b_i - b'_i}{r_i}$. This then leaves a maximum in-network delay of $d_i - \frac{b_i - b'_i}{r_i}$, which is greater than 0 since $b'_i \geq b_i - d_i r_i$. Hence, to meet its deadline flow i requires a service curve of at least

$$SC'_i(t) = \begin{cases} 0, & \text{when } t < d_i - \dfrac{b_i - b'_i}{r_i} \\ b'_i + r_i \left(t - d_i + \dfrac{b_i - b'_i}{r_i} \right) = b_i + r_i(t - d_i), & \text{otherwise} \end{cases}$$

which is greater than $SC_i(t) = \begin{cases} 0, & \text{when } t < d_i \\ b_i + r_i(t - d_i), & \text{otherwise} \end{cases}$. Consequently, according to Proposition 10 the system needs a bandwidth of at least

$$R = \max \left\{ \sum_{i=1}^{n} r_i, \sup_{t \geq 0} \frac{\sum_{j \neq i} SC_j(t) + SC'_i(t)}{t} \right\} \geq \max \left\{ \sum_{i=1}^{n} r_i, \sup_{t \geq 0} \frac{\sum_{1 \leq j \leq n} SC_j(t)}{t} \right\} = R^*_{sc}, \tag{B.2}$$

to meet each flow's deadline. □

B.3 Proofs for Static Priority Scheduler

B.3.1 Proof for Proposition 13

We actually prove Proposition 13 under the more general packet-based model. By assuming that all packets to have a length of 0, the packet-based model defaults to the fluid model.

PROPOSITION 13. *Consider a one-hop network shared by n token-bucket controlled flows, where flow $i, 1 \leq i \leq n$, has a traffic contract of (r_i, b_i) and a deadline of d_i, with $d_1 > d_2 > ... > d_n$ and $d_1 < \infty$. Under a static-priority scheduler, there exists an assignment of flows to priorities that minimizes link bandwidth while meeting all flows deadlines such that flow i is assigned a priority strictly greater than that of flow j only if $d_i < d_j$.*

Proof. For a mechanism Γ, denote flows with priority h under Γ as $G_h(\Gamma)$. Define $d_h^{(max)}(\Gamma) = \max_{i \in G_h(\Gamma)} d_i$ and $d_h^{(min)}(\Gamma) = \min_{i \in G_h(\Gamma)} d_i$. Suppose priority class $h+1$ to have a higher priority than priority class h. Then we will prove the proposition by induction on the number k of priority classes. Our induction hypothesis $S(k)$ is expressed in the following statement:

$S(k)$: For a 1-hop topology with any number of flows, there exists an optimal k-priority mechanism Γ_k such that $\forall s < l \leq k$, $d_l^{(max)}(\Gamma_k) < d_s^{(min)}(\Gamma_k)$.

- **Base case:** consider the case when $k = 2$. We show that for any mechanism Γ_2, if $d_1^{(min)}(\Gamma_2) < d_2^{(max)}(\Gamma_2)$, then there exists a 2-priority mechanism Γ_2' such that $d_1^{(min)}(\Gamma_2') > d_2^{(max)}(\Gamma_2')$ and $R^*(\Gamma_2') \leq R^*(\Gamma_2)$.

For mechanism Γ_2, denote $l_1^{(max)}(\Gamma_2)$ to be the maximum packet size for flows in $G_1(\Gamma_2)$. To satisfy each flow's deadline, it requires a bandwidth R such that

$$\begin{cases} \dfrac{\sum_{i\in G_2(\Gamma_2)} b_i + l_1^{(max)}(\Gamma_2)}{R} \le d_2^{(min)}(\Gamma_2), \\ \dfrac{\sum_i b_i}{R - \sum_{i\in G_2(\Gamma_2)} r_i} \le d_1^{(min)}(\Gamma_2), \\ \sum_{i=1}^{n} r_i \le R, \end{cases} \tag{B.3}$$

which gives

$$R^*(\Gamma_2) = \max\left\{ \sum_{i=1}^{n} r_i, \frac{\sum_{i\in G_2(\Gamma_2)} b_i + l_1^{(max)}(\Gamma_2)}{d_2^{(min)}(\Gamma_2)}, \frac{\sum_i b_i}{d_1^{(min)}(\Gamma_2)} + \sum_{i\in G_2(\Gamma_2)} r_i \right\}.$$

Define $G_1'(\Gamma_2) = \{i \in G_2(\Gamma_2) \mid d_i > d_1^{(max)}(\Gamma_2)\}$ and $G_2'(\Gamma_2) = G_2(\Gamma_2) - G_1'(\Gamma_2)$. Consider the mechanism Γ_2' such that $G_2(\Gamma_2') = G_2'(\Gamma_2)$, and $G_1(\Gamma_2') = G_1'(\Gamma_2) \cup G_1(\Gamma_2)$. Note that $d_1^{(min)}(\Gamma_2') > d_2^{(max)}(\Gamma_2')$, and note also that $d_i^{(min)}(\Gamma_2') = d_i^{(min)}(\Gamma_2), i = 1, 2$. Similar as Eq. (B.3), we have

$$R^*(\Gamma_2') = \max\left\{ \sum_{i=1}^{n} r_i, \frac{\sum_{i\in G_2(\Gamma_2')} b_i + l_1^{(max)}(\Gamma_2')}{d_2^{(min)}(\Gamma_2)}, \frac{\sum_i b_i}{d_1^{(min)}(\Gamma_2)} + \sum_{i\in G_2(\Gamma_2) - G_1'(\Gamma_2)} r_i \right\}$$

Note that $\frac{\sum_i b_i}{d_1^{(min)}(\Gamma_2)} + \sum_{i\in G_2(\Gamma_2)} r_i \ge \frac{\sum_i b_i}{d_1^{(min)}(\Gamma_2)} + \sum_{i\in G_2(\Gamma_2) - G_1'(\Gamma_2)} r_i$. Next we show $\sum_{i\in G_2(\Gamma_2)} b_i + l_1^{(max)}(\Gamma_2) \ge \sum_{i\in G_2(\Gamma_2')} b_i + l_1^{(max)}(\Gamma_2')$, from which we then have $R^*(\Gamma_2) \ge R^*(\Gamma_2')$, and therefore $S(2)$. Since $G_1(\Gamma_2) \subsetneqq G_1(\Gamma_2')$, it has $l_1^{(max)}(\Gamma_2') \ge l_1^{(max)}(\Gamma_2)$.

- When $l_1^{(max)}(\Gamma_2') = l_1^{(max)}(\Gamma_2)$, from $G_2(\Gamma_2') \subsetneqq G_2(\Gamma_2)$ we have $\sum_{i\in G_2(\Gamma_2)} b_i \ge \sum_{i\in G_2(\Gamma_2')} b_i$, and therefore have $\sum_{i\in G_2(\Gamma_2)} b_i + l_1^{(max)}(\Gamma_2) \ge \sum_{i\in G_2(\Gamma_2')} b_i + l_1^{(max)}(\Gamma_2')$.

– When $l_1^{(max)}(\Gamma_2') > l_1^{(max)}(\Gamma_2)$, *i.e.,* there exists a flow $\hat{i} \in G_1'(\Gamma_2)$ such that $l_{\hat{i}} > l_1^{(max)}(\Gamma_2)$. From $b_{\hat{i}} \geq l_{\hat{i}}$ and $G_2(\Gamma_2') \subseteq G_2(\Gamma_2) - \hat{i}$, we have

$$\sum_{i \in G_2(\Gamma_2)} b_i + l_1^{(max)}(\Gamma_2) = \sum_{i \in G_2(\Gamma_2) - \hat{i}} b_i + b_{\hat{i}} + l_1^{(max)}(\Gamma_2) \geq \sum_{i \in G_2(\Gamma_2')} b_i + l_{\hat{i}} + l_1^{(max)}(\Gamma_2)$$
$$> \sum_{i \in G_2(\Gamma_2')} b_i + l_1^{(max)}(\Gamma_2')$$

- **Induction Step:** Let $k \geq 2$ and suppose $S(k)$ holds. Below we show that $S(k+1)$ holds.

  Consider any (k+1)-priority mechanism Γ_{k+1}. For $1 \leq h \leq (1+k)$, denote $l_h^{(max)}(\Gamma_{k+1})$ to be the maximum packet size for all flows with priority strictly smaller than h, and define $l_h^{(max)}(\Gamma_{k+1})$ to be 0 if no flow has a priority strictly smaller than h. Define $B_h^{(k)}(\Gamma_{k+1}) = \sum_{j=h}^{k} \sum_{i \in G_j(\Gamma_{k+1})} b_i$, *i.e.,* the sum of bucket sizes for flows with priority $h \leq j \leq k$, and $R_h^{(k)}(\Gamma_{k+1}) = \sum_{j=h}^{k} \sum_{i \in G_j(\Gamma_{k+1})} r_i$, *i.e.,* the sum of rates for flows with priority in $h \leq j \leq k$. Then to satisfy each flow's deadline Γ_{k+1} requires a bandwidth R satisfying

$$\begin{cases} \frac{B_h^{(k+1)}(\Gamma_{k+1}) + l_h^{(max)}(\Gamma_{k+1})}{R - R_{h+1}^{(k+1)}(\Gamma_{k+1})} \leq d_h^{(min)}(\Gamma_{k+1}), \quad \forall h \in [1, k+1]; \\ \\ \sum_{i=1}^{n} r_i \leq R; \end{cases} \tag{B.4}$$

  which gives a minimum required bandwidth of

$$R^*(\Gamma_{k+1}) = \max_{1 \leq h \leq k+1} \left\{ \sum_{i=1}^{n} r_i, \ \frac{B_h^{(k+1)}(\Gamma_{k+1}) + l_h^{(max)}(\Gamma_{k+1})}{d_h^{(min)}(\Gamma_{k+1})} + R_{h+1}^{(k+1)}(\Gamma_{k+1}) \right\}. \tag{B.5}$$

  Afterwards, we first show that there exists a $(k+1)$-priority mechanism Γ_{k+1}' satisfying

– **Condition 1**: $G_{k+1}(\Gamma'_{k+1}) = \{i \mid d_n \leq d_i < d_{\hat{n}}\}$ where $1 \leq \hat{n} \leq n$, and $R^*(\Gamma'_{k+1}) \leq R^*(\Gamma_{k+1})$, where $G_{k+1}(\Gamma'_{k+1}) = \emptyset$ if $\hat{n} = n$.

After that, under a slight abuse of notation, we show that for any Γ_{k+1} satisfying Condition 1, there exists a $(k+1)$-priority mechanism Γ'_{k+1} satisfying

– **Condition 2**: $R^*(\Gamma'_{k+1}) \leq R^*(\Gamma_{k+1})$, and $d_i^{(max)}(\Gamma'_{k+1}) < d_j^{(min)}(\Gamma'_{k+1})$ for all $j < i \leq k+1$.

Combining them gives $S(k+1)$.

1. We first show the existence of a mechanism Γ'_{k+1} satisfying condition 1. If Γ_{k+1} satisfies Condition 1, then $\Gamma_{k+1} = \Gamma'_{k+1}$. Otherwise, for all $1 \leq \hat{n} < n$, $G_{k+1}(\Gamma_{k+1}) \neq \{i \mid d_n \leq d_i < d_{\hat{n}}\}$. Define $\hat{i} = \max\{1 \leq i \leq n \mid i \notin G_{k+1}(\Gamma_{k+1})\}$ and suppose $\hat{i} \in G_{\hat{h}}(\Gamma_{k+1})$. Further define $G'_{k+1} = \{i \in G_{k+1}(\Gamma_{k+1}) \mid d_i < d_{\hat{h}}^{(min)}(\Gamma_{k+1})\}$ and $G'_{\hat{h}} = G_{k+1}(\Gamma_{k+1}) - G'_{k+1}$.

   Consider the mechanism Γ'_{k+1} such that 1) $G_{k+1}(\Gamma'_{k+1}) = G'_{k+1}$, 2) $G_{\hat{h}}(\Gamma'_{k+1}) = G'_{\hat{h}} + G_{\hat{h}}(\Gamma_{k+1})$, and 3) $G_i(\Gamma'_{k+1}) = G_i(\Gamma_{k+1})$, when $1 \leq i \leq k$ and $i \neq \hat{h}$. Note that

   – when $h \leq \hat{h}$, $B_h^{(k+1)}(\Gamma'_{k+1}) = B_h^{(k+1)}(\Gamma_{k+1})$, $R_h^{(k+1)}(\Gamma'_{k+1}) = R_h^{(k+1)}(\Gamma_{k+1})$, and $l_h^{(max)}(\Gamma'_{k+1}) = l_h^{(max)}(\Gamma_{k+1})$;
   – when $h > \hat{h}$, $B_h^{(k+1)}(\Gamma'_{k+1}) = B_h^{(k+1)}(\Gamma_{k+1}) - \sum_{i \in G'_{\hat{h}}} b_i$, $R_h^{(k+1)}(\Gamma'_{k+1}) = R_h^{(k+1)}(\Gamma'_{k+1}) - \sum_{i \in G'_{\hat{h}}} r_i$, and $l_h^{(max)}(\Gamma'_{k+1}) = \max\left\{l_h^{(max)}(\Gamma_{k+1}),\ \max_{i \in G'_{\hat{h}}} l_i\right\}$.

Next we show that $R^*(\Gamma'_{k+1}) \leq R^*(\Gamma_{k+1})$. To satisfy each flow's deadline, Γ'_{k+1} requires a bandwidth R such that[59]

$$\begin{cases} \sum_{i=1}^{n} r_i \leq R, \\ \dfrac{B_{k+1}^{(k+1)}(\Gamma_{k+1}) - \sum_{i\in G'_{\hat{h}}} b_i + l_{k+1}^{(max)}(\Gamma'_{k+1})}{R} \leq d_n, \\ \dfrac{B_{\hat{h}}^{(k+1)}(\Gamma_{k+1}) + l_{\hat{h}}^{(max)}(\Gamma_{k+1})}{R - R_{\hat{h}+1}^{(k+1)}(\Gamma_{k+1}) + \sum_{i\in G'_{\hat{h}}} r_i} \leq d_{\hat{h}}^{(min)}(\Gamma_{k+1}) \\ \dfrac{B_{h}^{(k+1)}(\Gamma_{k+1}) - \sum_{i\in G'_{\hat{h}}} b_i + l_{h}^{(max)}(\Gamma'_{k+1})}{R - R_{h+1}^{(k+1)}(\Gamma_{k+1}) + \sum_{i\in G'_{\hat{h}}} r_i} \leq d_{h}^{(min)}(\Gamma_{k+1}), \text{ when } \hat{h} < h \leq k \\ \dfrac{B_{h}^{(k+1)}(\Gamma_{k+1}) + l_{h}^{(max)}(\Gamma_{k+1})}{R - R_{h+1}^{(k+1)}(\Gamma_{k+1})} \leq d_{h}^{(min)}(\Gamma_{k+1}), \text{ when } h < \hat{h} \end{cases}$$

which gives a minimum required bandwidth $R^*(\Gamma')$ of

$$\max \begin{cases} \sum_{i=1}^{n} r_i, \\ \dfrac{B_{k+1}^{(k+1)}(\Gamma_{k+1}) - \sum_{i\in G'_{\hat{h}}} b_i + l_{k+1}^{(max)}(\Gamma'_{k+1})}{d_n}, \\ \dfrac{B_{\hat{h}}^{(k+1)}(\Gamma_{k+1}) + l_{\hat{h}}^{(max)}(\Gamma_{k+1})}{d_{\hat{h}}^{(min)}(\Gamma_{k+1})} + R_{\hat{h}+1}^{(k+1)}(\Gamma_{k+1}) - \sum_{i\in G'_{\hat{h}}} r_i \\ \dfrac{B_{h}^{(k+1)}(\Gamma_{k+1}) - \sum_{i\in G'_{\hat{h}}} b_i + l_{h}^{(max)}(\Gamma'_{k+1})}{d_{h}^{(min)}(\Gamma_{k+1})} + R_{h+1}^{(k+1)}(\Gamma_{k+1}) - \sum_{i\in G'_{\hat{h}}} r_i, \text{ when } \hat{h} < h \leq k \\ \dfrac{B_{h}^{(k+1)}(\Gamma_{k+1}) + l_{h}^{(max)}(\Gamma_{k+1})}{d_{h}^{(min)}(\Gamma_{k+1})} + R_{h+1}^{(k+1)}(\Gamma_{k+1}), \text{ when } h < \hat{h} \end{cases}$$

[59] When $G_{k+1}(\Gamma'_{k+1}) = \emptyset$, $\frac{B_{k+1}^{(k+1)}(\Gamma_{k+1}) - \sum_{i\in G'_{\hat{h}}} b_i + l_{k+1}^{(max)}(\Gamma'_{k+1})}{R} = \frac{\max_i l_i}{R}$, which is also no greater than d_n.

Note that if for all $\hat{h} < h \leq k+1$, $l_h^{(max)}(\Gamma'_{k+1}) - \sum_{i \in G'_{\hat{h}}} b_i \leq l_h^{(max)}(\Gamma_{k+1})$, we will then have $R^*(\Gamma'_{k+1}) \leq R^*(\Gamma_{k+1})$. In fact, as $\max_{i \in G'_{\hat{h}}} l_i - \sum_{i \in G'_{\hat{h}}} b_i \leq 0$ we have

$$\begin{aligned} l_h^{(max)}(\Gamma'_{k+1}) - \sum_{i \in G'_{\hat{h}}} b_i &= \max \left\{ l_h^{(max)}(\Gamma_{k+1}),\ \max_{i \in G'_{\hat{h}}} l_i \right\} - \sum_{i \in G'_{\hat{h}}} b_i \\ &\leq \max \left\{ l_h^{(max)}(\Gamma_{k+1}) - \sum_{i \in G'_{\hat{h}}} b_i, 0 \right\} < l_h^{(max)}(\Gamma_{k+1}). \end{aligned}$$

Thus, we show the existence of a mechanism Γ'_{k+1} satisfying Condition 1.

2. Next we show that for any $(k+1)$-priority mechanism Γ_{k+1} satisfying Condition 1, there exists a $(k+1)$-priority mechanism Γ'_{k+1} satisfying Condition 2.

   For Γ_{k+1}, there to exist $1 \leq \hat{n} \leq n$ such that $G_{k+1}(\Gamma_{k+1}) = \{ i \mid d_n \leq d_i < d_{\hat{n}} \}$. If $G_{k+1}(\Gamma_{k+1}) = \emptyset$, by induction of hypothesis $S(k)$ we have $S(k+1)$. Afterwards we consider the case where $G_{k+1}(\Gamma_{k+1}) \neq \emptyset$.

   Consider flows $\tilde{F} = \{(r_1, b_1, l_1, d_1), ..., (r_{\hat{n}-1}, b_{\hat{n}-1}, l_{\hat{n}-1}, d_{\hat{n}-1}), (r_{\hat{n}}, \sum_{i \geq \hat{n}} b_i, l_{\hat{n}}, d_{\hat{n}})\}$. According to $S(k)$, there exists a k-priority mechanism Γ'_k for $\tilde{F}$ such that $\forall j < i \leq k$, $\tilde{d}_i^{(max)}(\Gamma'_k) < \tilde{d}_j^{(min)}(\Gamma'_k)$. Γ_k gives a minimum required bandwidth of

$$R^*(\Gamma'_k) = \max_{1 \leq h \leq k} \left\{ \sum_{i=1}^{\hat{n}} r_i,\ \frac{\tilde{B}_h^{(k)}(\Gamma'_k) + l_h^{(max)}(\Gamma'_k)}{d_h^{(min)}(\Gamma'_k)} + \tilde{R}_{h+1}^{(k)}(\Gamma'_k) \right\}$$

   Consider the k-priority mechanism Γ_k, where $G_h(\Gamma_k) = G_h(\Gamma_{k+1})$ for all $1 \leq h \leq k$. Applying Γ_k to $\tilde{F}$, we have

$$\begin{aligned} R^*(\Gamma_k) &= \max_{1 \leq h \leq k} \left\{ \sum_{i=1}^{\hat{n}} r_i, \frac{\tilde{B}_h^{(k)}(\Gamma_k) + l_h^{(max)}(\Gamma_k)}{d_h^{(min)}(\Gamma_k)} + \tilde{R}_{h+1}^{(k)}(\Gamma_k) \right\} \\ &= \max_{1 \leq h \leq k} \left\{ \sum_{i=1}^{\hat{n}} r_i, \frac{B_h^{(k+1)}(\Gamma_{k+1}) + l_h^{(max)}(\Gamma_{k+1})}{d_h^{(min)}(\Gamma_{k+1})} + R_{h+1}^{(k+1)}(\Gamma_{k+1}) - \sum_{i=\hat{n}+1}^{n} r_i \right\}. \end{aligned}$$

From $R^*(\Gamma_k) \geq R^*(\Gamma'_k)$, we know that

$$\max_{1\leq h\leq k}\left\{\frac{\tilde{B}_h^{(k)}(\Gamma'_k)+l_h^{(max)}(\Gamma'_k)}{d_h^{(min)}(\Gamma'_k)}+\tilde{R}_{h+1}^{(k)}(\Gamma'_k)\right\}$$
$$\leq \max\left\{\sum_{i=1}^{\hat{n}} r_i,\ \max_{1\leq h\leq k}\left\{\frac{B_h^{(k+1)}(\Gamma_{k+1})+l_h^{(max)}(\Gamma)}{d_h^{(min)}(\Gamma_{k+1})}+R_{h+1}^{(k+1)}(\Gamma_{k+1})-\sum_{i=\hat{n}+1}^{n} r_i\right\}\right\}$$

which further gives

$$\max_{1\leq h\leq k}\left\{\frac{\tilde{B}_h^{(k)}(\Gamma'_k)+l_h^{(max)}(\Gamma'_k)}{d_h^{(min)}(\Gamma'_k)}+\tilde{R}_{h+1}^{(k)}(\Gamma'_k)+\sum_{i=\hat{n}+1}^{n} r_i\right\}$$
$$\leq \max\left\{\sum_{i=1}^{n} r_i, \max_{1\leq h\leq k}\left\{\frac{B_h^{(k+1)}(\Gamma_{k+1})+l_h^{(max)}(\Gamma_{k+1})}{d_h^{(min)}(\Gamma_{k+1})}+R_{h+1}^{(k+1)}(\Gamma_{k+1})\right\}\right\} \tag{B.6}$$

Now consider the $(k+1)$-priority mechanism Γ'_{k+1}, where $G_h(\Gamma'_{k+1}) = G_h(\Gamma'_k)$ for all $1 \leq h \leq k$, and $G_{k+1}(\Gamma'_{k+1}) = G_{k+1}(\Gamma_{k+1})$. By the definition of $G_{k+1}(\Gamma_{k+1})$ and Γ'_k, we know that $d_i^{(max)}(\Gamma'_{k+1}) < d_j^{(min)}(\Gamma'_{k+1}), \forall j < i \leq k+1$. Next we show that $R^*(\Gamma'_{k+1}) \leq R^*(\Gamma_{k+1})$.

Applying Γ'_{k+1} to flow $F = \{(r_i, b_i, l_i, d_i) \mid 1 \leq i \leq n\}$, we have

$$R^*(\Gamma'_{k+1}) = \max_{1\leq h\leq k+1}\left\{\sum_{i=1}^{n} r_i,\ \frac{B_h^{(k+1)}(\Gamma'_{k+1})+l_h^{(max)}(\Gamma'_{k+1})}{d_h^{(min)}(\Gamma'_{k+1})}+R_{h+1}^{(k+1)}(\Gamma'_{k+1})\right\}$$
$$= \max_{1\leq h\leq k}\left\{\sum_{i=1}^{n} r_i, \frac{B_{k+1}^{(k+1)}(\Gamma_{k+1})+l_{k+1}^{(max)}(\Gamma_{k+1})}{d_{k+1}^{(min)}(\Gamma_{k+1})}, \frac{B_h^{(k+1)}(\Gamma'_{k+1})+l_h^{(max)}(\Gamma'_{k+1})}{d_h^{(min)}(\Gamma'_{k+1})}+R_{h+1}^{(k+1)}(\Gamma'_{k+1})\right\}$$
$$= \max_{1\leq h\leq k}\left\{\sum_{i=1}^{n} r_i, \frac{B_{k+1}^{(k+1)}(\Gamma_{k+1})+l_{k+1}^{(max)}(\Gamma_{k+1})}{d_{k+1}^{(min)}(\Gamma_{k+1})}, \frac{\tilde{B}_h^{(k)}(\Gamma'_k)+l_h^{(max)}(\Gamma'_k)}{d_h^{(min)}(\Gamma'_k)}+\tilde{R}_{h+1}^{(k)}(\Gamma'_k)+\sum_{i=\hat{n}+1}^{n} r_i\right\}$$

Combining it with Eq. (B.6), we have

$$
\begin{aligned}
R^*(\Gamma'_{k+1}) &\leq \max \left\{ \begin{array}{l} \sum_{i=1}^{n} r_i, \\ \\ \frac{B_{k+1}^{(k+1)}(\Gamma_{k+1}) + l_{k+1}^{(max)}(\Gamma_{k+1})}{d_{k+1}^{(min)}(\Gamma_{k+1})}, \\ \\ \max_{1 \leq h \leq k} \left\{ \frac{B_h^{(k+1)}(\Gamma_{k+1}) + l_h^{(max)}(\Gamma_{k+1})}{d_h^{(min)}(\Gamma_{k+1})} + R_{h+1}^{(k+1)}(\Gamma_{k+1}) \right\} \end{array} \right. \\
&= \max \left\{ \sum_{i=1}^{n} r_i, \max_{1 \leq h \leq k+1} \left\{ \frac{B_h^{(k+1)}(\Gamma_{k+1}) + l_h^{(max)}(\Gamma_{k+1})}{d_h^{(min)}(\Gamma_{k+1})} + R_{h+1}^{(k+1)}(\Gamma_{k+1}) \right\} \right\} \\
&= R^*(\Gamma_{k+1})
\end{aligned}
$$

Hence we show the existence of a mechanism Γ'_{k+1} satisfying Condition 2.

□

B.3.2 Proof for Proposition 14

PROPOSITION 14. *Consider a one-hop network shared by n token-bucket controlled flows, where flow $i, 1 \leq i \leq n$, has a traffic contract of (r_i, b_i). Assume a static priority scheduler that assigns flow i a priority of i, where priority n is the highest priority, and (re)shapes flow i to (r_i, b'_i), where $0 \leq b'_i \leq b_i$. Given a shared link bandwidth of $R \geq \sum_{j=1}^{n} r_j$, the worst-case delay for flow i is*

$$
D_i^* = max \left\{ \frac{b_i + B'_{i+1}}{R - R_{i+1}}, \ \frac{b_i - b'_i}{r_i} + \frac{B'_{i+1}}{R - R_{i+1}} \right\}. \tag{3.7}
$$

Proof. Flow $1 \leq i \leq n$ receives a service curve of $SC_1^{(i)} = \begin{cases} b_i' + r_i t, \text{ when } t > 0 \\ 0 \qquad \text{otherwise} \end{cases}$ inside the shaper, and a service curve of $SC_2^{(i)}(t) = \left[(R - R_{i+1})t - B_{i+1}'\right]^+$ at the shared link. Overall it receives a service curve of[60]

$$SC^{(i)}(t) = SC_1^{(i)} \otimes SC_2^{(i)} = \min\left\{\left[b_i' + r_i\left(t - \frac{B_{i+1}'}{R - R_{i+1}}\right)\right]^+, [(R - R_{i+1})t - B_{i+1}']^+\right\}.$$

Then we have

$$d_i^* = \sup_{t \geq 0} \inf_{\tau \geq 0} \left\{b_i + r_i t \leq SC^{(i)}(t + \tau)\right\} = \max\left\{\frac{b_i + B_{i+1}'}{R - R_{i+1}},\ \frac{b_i - b_i'}{r_i} + \frac{B_{i+1}'}{R - R_{i+1}}\right\}.$$

□

B.3.3 Proofs for Proposition 16 and 17

This section provides the solution for **OPT_S**, from which Proposition 16 and 17 derive. For the reader's convenience, we restate **OPT_S**. Remember that we define $B_i' = \sum_{j=i}^n b_j'$ and $R_i = \sum_{j=i}^n r_j$, where $B_i' = R_i = 0$ when $i > n$.

$$\begin{aligned} \textbf{OPT\_S} \quad & \min_{\boldsymbol{b'}} R \\ \text{s.t} \quad & \max\left\{\frac{b_i + B_{i+1}'}{R - R_{i+1}},\ \frac{b_i - b_i'}{r_i} + \frac{B_{i+1}'}{R - R_{i+1}}\right\} \leq d_i, \quad \forall\ 1 \leq i \leq n, \\ & R_1 \leq R, \qquad b_1' \leq b_1, \qquad 0 \leq b_i' \leq b_i, \qquad \forall\ 2 \leq i \leq n. \end{aligned}$$

Instead of solving **OPT_S**, for technical simplicity we consider **OPT_S'**, whose solution directly gives that for **OPT_S**. Next we first demonstrate the relationship between **OPT_S**

[60] See THEOREM 1.4.6 in [120], page 28

and **OPT_S'** (Lemma 36), and then proceed to solve **OPT_S'** (Lemma 37). Combining Lemma 36 and 37, we then have Proposition 16 and 17.

Lemma 36. *For $1 \leq i \leq n$, define $H_i = b_i - r_i d_i$, and $\boldsymbol{B'} = (B'_1, ..., B'_n)$. Consider the following optimization:*

$$\begin{aligned}
\textbf{OPT_S'} \quad & \min_{\boldsymbol{B'}} R \\
s.t \quad & R_1 \leq R. \\
& B'_2 \leq d_1(R - R_2) - b_1, \\
& B'_i \in \left[\max \left\{ \frac{R - R_{i+1} + r_i}{R - R_{i+1}} B'_{i+1} + H_i,\ B'_{i+1} \right\},\ B'_{i+1} + \frac{b_i(R - R_i)}{R - R_{i+1}} \right], \quad \forall\ 2 \leq i \leq n.
\end{aligned} \tag{B.7}$$

*Suppose the optimal solution for **OPT_S'** is $(R^*, \boldsymbol{B'}^*)$. Then $(R^*, \boldsymbol{b'}^*)$, where*

$$\boldsymbol{b'}^* = (b_1, B'^*_2 - B'^*_3, ..., B'^*_{n-1} - B'^*_n, B'^*_n),$$

*is an optimal solution for **OPT_S**.*

Proof. We first show that $\boldsymbol{b'}^* = (b_1, B'^*_2 - B'^*_3, ..., B'^*_{n-1} - B'^*_n, B'^*_n)$ and R^* satisfy all the constraints for **OPT_S**, and then show that $(R^*, \boldsymbol{b'}^*)$ is an optimal solution for **OPT_S**.

Substituting $b'_1 = b_1$ into $\max \left\{ \frac{b_1 + B'_2}{R - R_2},\ \frac{b_1 - b'_1}{r_1} + \frac{B'_2}{R - R_2} \right\} \leq d_1$ gives $\frac{b_1 + B'_2}{R - R_2} \leq d_1$, which is equivalent to $B'_2 \leq d_1(R - R_2) - b_1$. Thus, to show the feasibility of $(R^*, \boldsymbol{b'}^*)$, we only need to show that $(R^*, \boldsymbol{b'}^*)$ satisfies $\max \left\{ \frac{b_i + B'^*_{i+1}}{R^* - R_{i+1}},\ \frac{b_i - b'^*_i}{r_i} + \frac{B'^*_{i+1}}{R^* - R_{i+1}} \right\} \leq d_i$ and $b'^*_i \in [0, b_i]$ for all $2 \leq i \leq n$. Below we consider each constraint separately.

- Basic algebraic manipulation gives that

$$\max\left\{\frac{b_i+B'^*_{i+1}}{R^*-R_{i+1}},\ \frac{b_i-b'^*_i}{r_i}+\frac{B'^*_{i+1}}{R^*-R_{i+1}}\right\}=\begin{cases}\dfrac{b_i+B'^*_{i+1}}{R^*-R_{i+1}}, & \text{when } b'^*_i\geq\dfrac{b_i(R^*-R_i)}{R^*-R_{i+1}},\\ \dfrac{b_i-b'^*_i}{r_i}+\dfrac{B'^*_{i+1}}{R^*-R_{i+1}}, & \text{otherwise.}\end{cases} \tag{B.8}$$

  From **OPT_S'** we have $B'^*_i\leq B'^*_{i+1}+\frac{b_i(R^*-R_i)}{R^*-R_{i+1}}$, *i.e.*, $b'^*_i=B'^*_i-B'^*_{i+1}\leq\frac{b_i(R^*-R_i)}{R^*-R_{i+1}}$, and therefore $\max\left\{\frac{b_i+B'^*_{i+1}}{R^*-R_{i+1}},\ \frac{b_i-b'^*_i}{r_i}+\frac{B'^*_{i+1}}{R^*-R_{i+1}}\right\}=\frac{b_i-b'^*_i}{r_i}+\frac{B'^*_{i+1}}{R^*-R_{i+1}}$. From **OPT_S'** we also have $B'^*_i\geq\frac{R^*-R_{i+1}+r_i}{R^*-R_{i+1}}B'^*_{i+1}+H_i$, *i.e.*, $B'^*_i-B'^*_{i+1}-H_i=b'^*_i-b_i+r_id_i\geq\frac{r_iB'^*_{i+1}}{R-R_{i+1}}$, which is equivalent to $\frac{b_i-b'^*_i}{r_i}+\frac{B'^*_{i+1}}{R-R_{i+1}}\leq d_i$. Combining them gives $\max\left\{\frac{b_i+B'^*_{i+1}}{R^*-R_{i+1}},\ \frac{b_i-b'^*_i}{r_i}+\frac{B'^*_{i+1}}{R^*-R_{i+1}}\right\}\leq d_i$.

- From **OPT_S'** we have $B'^*_i\geq B'^*_{i+1}$, and therefore $b'^*_i=B'^*_i-B'^*_{i+1}\geq 0$. From **OPT_S'** we also have $B'^*_i\leq B'^*_{i+1}+\frac{b_i(R^*-R_i)}{R^*-R_{i+1}}$, and therefore $b'^*_i\leq\frac{b_i(R^*-R_i)}{R^*-R_{i+1}}<b_i$. Thus, we have $b'^*_i\in[0,b_i]$.

Next we show by contradiction that $(R^*,\boldsymbol{b'}^*)$ is optimal for **OPT_S**. Suppose $(\tilde{R},\tilde{\boldsymbol{b}}')$, where $\tilde{R}<R^*$ is an optimal solution for **OPT_S**. Denote $\tilde{B}'_i=\sum_{j=i}^n\tilde{b}'_j$, and $\tilde{\boldsymbol{B}}'=(\tilde{B}'_1,...,\tilde{B}'_n)$.

- If $(\tilde{R},\tilde{\boldsymbol{B}}')$ satisfies all constraints for **OPT_S'**, then by $(R^*,\boldsymbol{b'}^*)$'s optimality we know that $\tilde{R}\geq R^*$, which contradicts to the assumption that $\tilde{R}<R^*$.

- Otherwise, there exists $2\leq i\leq n$ such that $\max\left\{\frac{b_i+\tilde{B}'_{i+1}}{\tilde{R}-R_{i+1}},\ \frac{b_i-\tilde{b}'_i}{r_i}+\frac{\tilde{B}'_{i+1}}{\tilde{R}-R_{i+1}}\right\}\leq d_i$ and $\tilde{b}'_i\in[0,b_i]$, whereas $\tilde{B}'_i\notin\left[\max\left\{\frac{\tilde{R}-R_{i+1}+r_i}{\tilde{R}-R_{i+1}}\tilde{B}'_{i+1}+H_i,\ \tilde{B}'_{i+1}\right\},\ \tilde{B}'_{i+1}+\frac{b_i(\tilde{R}-R_i)}{\tilde{R}-R_{i+1}}\right]$.

  - When $\tilde{b}'_i\leq\frac{b_i(\tilde{R}-R_i)}{\tilde{R}-R_{i+1}}$, from Eq. (B.8) we have that $\max\left\{\frac{b_i+\tilde{B}'_{i+1}}{\tilde{R}-R_{i+1}},\ \frac{b_i-\tilde{b}'_i}{r_i}+\frac{\tilde{B}'_{i+1}}{\tilde{R}-R_{i+1}}\right\}=\frac{b_i-\tilde{b}'_i}{r_i}+\frac{\tilde{B}'_{i+1}}{\tilde{R}-R_{i+1}}$. Combining $\frac{b_i-\tilde{b}'_i}{r_i}+\frac{\tilde{B}'_{i+1}}{\tilde{R}-R_{i+1}}\leq d_i$ and $\tilde{b}'_i\in[0,\frac{b_i(\tilde{R}-R_i)}{\tilde{R}-R_{i+1}}]$ with $\tilde{b}'_i=$

$\tilde{B}'_i - \tilde{B}'_{i-1}$, we have

$$\tilde{B}'_i \in \left[\max\left\{\frac{\tilde{R}-R_{i+1}+r_i}{\tilde{R}-R_{i+1}}\tilde{B}'_{i+1}+H_i,\ \tilde{B}'_{i+1}\right\},\ \tilde{B}'_{i+1}+\frac{b_i(\tilde{R}-R_i)}{\tilde{R}-R_{i+1}}\right],$$

and therefore a contradiction.

– When $\tilde{b}'_i > \frac{b_i(\tilde{R}-R_i)}{\tilde{R}-R_{i+1}}$, we show that there exists an optimal solution $(\tilde{R}, \hat{\boldsymbol{b}}')$ for **OPT_S** such that $\hat{b}'_i \le \frac{b_i(\tilde{R}-R_i)}{\tilde{R}-R_{i+1}}$. Define $\hat{\boldsymbol{b}}'$ as 1) $\hat{b}'_i = \frac{b_i(\tilde{R}-R_i)}{\tilde{R}-R_{i+1}}$, and 2) $\hat{b}'_h = \tilde{b}'_h$ when $h \neq i$. Denote $\hat{B}'_h = \sum_{j=h}^{n} \hat{b}'_j \le \tilde{B}'_h$, and $\hat{\boldsymbol{B}}' = (\hat{B}'_1, ..., \hat{B}'_n)$. Next we show that $(\tilde{R}, \hat{\boldsymbol{b}}')$ satisfies all the constraints in **OPT_S**, and therefore is optimal. Observe first that from $\hat{B}'_2 < \tilde{B}'_2$, we have $\frac{b_1+\hat{B}'_2}{\tilde{R}-R_2} < \frac{b_1+\tilde{B}'_2}{\tilde{R}-R_2} \le d_1$. Below we show that for all $2 \le h \le n$, $\max\left\{\frac{b_h+\hat{B}'_{h+1}}{\tilde{R}-R_{h+1}},\ \frac{b_h-\hat{b}'_h}{r_h}+\frac{\hat{B}'_{h+1}}{\tilde{R}-R_{h+1}}\right\} \le d_h$.

  * When $h > i$, by definition $\hat{B}'_{h+1} = \tilde{B}'_{h+1}$. Thus we have

$$\max\left\{\frac{b_h+\hat{B}'_{h+1}}{\tilde{R}-R_{h+1}},\ \frac{b_h-\hat{b}'_h}{r_h}+\frac{\hat{B}'_{h+1}}{\tilde{R}-R_{h+1}}\right\} = \max\left\{\frac{b_h+\tilde{B}'_{h+1}}{\tilde{R}-R_{h+1}},\ \frac{b_h-\tilde{b}'_h}{r_h}+\frac{\tilde{B}'_{h+1}}{\tilde{R}-R_{h+1}}\right\} \le d_h.$$

  * When $h = i$, $\max\left\{\frac{b_h+\hat{B}'_{h+1}}{\tilde{R}-R_{h+1}},\ \frac{b_h-\hat{b}'_h}{r_h}+\frac{\hat{B}'_{h+1}}{\tilde{R}-R_{h+1}}\right\} = \frac{b_h+\hat{B}'_{h+1}}{\tilde{R}-R_{h+1}}$. Combining it with $\hat{B}'_{h+1} < \tilde{B}'_{h+1}$ and $\max\left\{\frac{b_h+\tilde{B}'_{h+1}}{\tilde{R}-R_{h+1}},\ \frac{b_h-\tilde{b}'_h}{r_h}+\frac{\tilde{B}'_{h+1}}{\tilde{R}-R_{h+1}}\right\} = \frac{b_h+\tilde{B}'_{h+1}}{\tilde{R}-R_{h+1}} \le d_h$ gives the result.

  * When $h < i$, from $\hat{B}'_{h+1} < \tilde{B}'_{h+1}$ we have $\frac{b_h+\hat{B}'_{h+1}}{\tilde{R}-R_{h+1}} < \frac{b_h+\tilde{B}'_{h+1}}{\tilde{R}-R_{h+1}}$ and $\frac{b_h-\hat{b}'_h}{r_h} + \frac{\hat{B}'_{h+1}}{\tilde{R}-R_{h+1}} < \frac{b_h-\tilde{b}'_h}{r_h}+\frac{\tilde{B}'_{h+1}}{\tilde{R}-R_{h+1}}$, *i.e.*, $\max\left\{\frac{b_h+\hat{B}'_{h+1}}{\tilde{R}-R_{h+1}},\ \frac{b_h-\hat{b}'_h}{r_h}+\frac{\hat{B}'_{h+1}}{\tilde{R}-R_{h+1}}\right\} < \max\left\{\frac{b_h+\tilde{B}'_{h+1}}{\tilde{R}-R_{h+1}},\ \frac{b_h-\tilde{b}'_h}{r_h}+\frac{\tilde{B}}{\tilde{R}-}\right.$ d_h.

If $(\tilde{R}, \hat{\boldsymbol{b}}')$ satisfies all the constraints for **OPT_S**, it contradicts to the assumption that $\tilde{R} < R^*$. Otherwise, from the case for $\tilde{b}'_i \le \frac{b_i(\tilde{R}-R_i)}{\tilde{R}-R_{i+1}}$, we know it again to produce a contradiction.

□

Next we proceed to solve **OPT_S'**, combining which with Lemma 36 then gives Proposition 16 and 17. For the reader's convenience, we restate the Propositions.

PROPOSITION 16. *For $1 \leq i \leq n$, denote $H_i = b_i - d_i r_i$, $\Pi_i(R) = \frac{r_i + R - R_{i+1}}{R - R_{i+1}}$ and $V_i(R) = d_i(R - R_{i+1}) - b_i$. Define $S_1(R) = \{V_1(R)\}$, and $S_i(R) = S_{i-1}(R) \bigcup \{V_i(R)\} \bigcup \left\{ \frac{s - H_i}{\Pi_i(R)} \mid s \in S_{i-1}(R) \right\}$ for $2 \leq i \leq n$. Then we have $\tilde{R}_s^* = \max\{R_1, \inf\{R \mid \forall s \in S_n(R), s \geq 0\}\}$.*

PROPOSITION 17. *$\tilde{R}_s^*$ can be achieved by*

$$b_i'^* = \begin{cases} \max\{0, b_n - r_n d_n\}, & \text{when } i = n; \\ \max\left\{0,\ b_i - r_i d_i + \dfrac{r_i B_{i+1}'^*}{\tilde{R}_s^* - R_{i+1}}\right\}, & \text{when } 2 \leq i \leq n-1. \end{cases} \tag{B.9}$$

Denote $\mathbb{B}_i := \left[\max\left\{\frac{R - R_{i+1} + r_i}{R - R_{i+1}} B_{i+1}' + H_i,\ B_{i+1}'\right\},\ B_{i+1}' + \frac{b_i(R - R_i)}{R - R_{i+1}}\right]$, where the interval overlaps with that in the third constraint of **OPT_S'**. We then show that the system meets each flow's deadline only if the shared link has a bandwidth no less than $\max\{R_1, \inf\{R \mid \forall s \in S_n(R), s \geq 0\}$, which in turn gives the minimum required bandwidth R^*. As mentioned before, we achieve this by first solving **OPT_S'**, from which we then get the solution for **OPT_S** based on Lemma 36.

Lemma 37. *Define $s_1^{(i)} = \max\left\{\Pi_i(R) B_{i+1}' + H_i,\ B_{i+1}'\right\}$, $s_2^{(i)} = \min\left\{S_{i-1}(R), B_{i+1}' + \frac{b_i(R - R_i)}{R - R_{i+1}}\right\}$, and $\mathbb{S}_i = \left[s_1^{(i)}, s_2^{(i)}\right]$ for $2 \leq i \leq n$, where $\Pi_i(R) = \frac{r_i + R - R_{i+1}}{R - R_{i+1}}$, $H_i = b_i - d_i r_i$, $S_1(R) = \{V_1(R)\}$, $V_i(R) = d_i(R - R_{i+1}) - b_i$, and $S_i(R) = S_{i-1}(R) \bigcup \{V_i(R)\} \bigcup \left\{ \frac{s - H_i}{\Pi_i(R)} \mid s \in S_{i-1}(R) \right\}$. Then R and $\boldsymbol{b'}$ satisfies all the constraints in **OPT_S'** iff $\mathbb{S}_n \neq \emptyset$ and $R \geq R_1$,* i.e., *$R \geq \max\{R_1, \inf\{R \mid \forall s \in S_n(R), s \geq 0\}\}$.*

Proof. We rely on the following statement to show the Lemma:

Statement 1. $\left\{(R, \boldsymbol{b}') \mid B'_{i-1} \in \mathbb{S}_{i-1}\right\} \bigcap \{(R, \boldsymbol{b}') \mid B'_i \in \mathbb{B}_i\} = \{(R, \boldsymbol{b}') \mid B'_i \in \mathbb{S}_i\}$.

Given Statement 1, we then show that

Statement 2. $\{(R, \boldsymbol{b}') \mid B'_2 \leq V_1(R), B'_i \in \mathbb{B}_i, \forall\ 2 \leq i \leq n\} = \{(R, \boldsymbol{b}') \mid B'_n \in \mathbb{S}_n\}$.

Note that $B'_2 \leq V_1(R)$ corresponds to the second constraint in **OPT_S'**, while $B'_i \in \mathbb{B}_i, \forall\ 2 \leq i \leq n$ corresponds to its third constraint. Therefore, from Statement 2 we have that R and $\boldsymbol{b}'$ satisfies all the constraints in **OPT_S'** iff $\mathbb{S}_n \neq \emptyset$ and $R \geq R_1$. Note that $\mathbb{S}_n \neq \emptyset$ iff $s_1^{(n)} \leq s_2^{(n)}$, *i.e.,* $\max\{H_n, 0\} \leq \min\left\{S_{n-1}(R), \frac{b_n(R-R_n)}{R}\right\}$. As $(R - R_n) \geq 0$, $\max\{H_n, 0\} \leq \frac{b_n(R-R_n)}{R}$ iff $V_n(R) \geq 0$, whereas $\max\{H_n, 0\} \leq \min\{S_{n-1}(R)\}$ iff $s \geq \max\{0, H_n\}, \forall s \in S_{n-1}(R)$. Since $\Pi_n(R) > 0$, by the definition of S_n we have $\mathbb{S}_n \neq \emptyset$ and $R \geq R_1$ iff $R \geq \max\{R_1, \inf\{R \mid \forall s \in S_n(R), s \geq 0\}\}$.

Proof for Statement 1. Note that $\left\{(R, \boldsymbol{b}') \mid B'_{i-1} \in \mathbb{S}_{i-1}\right\} = \{(R, \boldsymbol{b}') \mid \mathbb{S}_{i-1} \neq \emptyset\}$. Below we prove that $\mathbb{S}_{i-1} \neq \emptyset$ iff $B'_i \leq \min\{S_{i-1}(R)\}$. As $s_2^{(i)} = \min\left\{S_{i-1}(R), B'_{i+1} + \frac{b_i(R-R_i)}{R-R_{i+1}}\right\}$, combining $B'_i \leq \min\{S_{i-1}(R)\}$ with $B'_i \in \mathbb{B}_i = \left[s_i^{(1)},\ B'_{i+1} + \frac{b_i(R-R_i)}{R-R_{i+1}}\right]$ directly gives $B'_i \in \mathbb{S}_i = \left[s_1^{(i)}, s_2^{(i)}\right]$.

From $\mathbb{S}_{i-1} \neq \emptyset \Leftrightarrow s_2^{(i-1)} \geq s_1^{(i-1)}$, we have

$$\begin{cases} B'_i \leq s_2^{(i-1)} = \min\left\{S_{i-2}(R), B'_i + \dfrac{b_{i-1}(R - R_{i-1})}{R - R_i}\right\} \Leftrightarrow B'_i \leq \min\{S_{i-2}(R)\} \\ \Pi_{i-1}(R)B'_i + H_{i-1} \leq s_2^{(i-1)} \Leftrightarrow B'_i \leq \min\{V_{i-1}(R)\} \bigcup \left\{\dfrac{s - H_{i-1}}{\Pi_{i-1}(R)} \mid s \in S_{i-2}(R)\right\} \end{cases}$$

As $S_{i-1}(R) = S_{i-2}(R) \bigcup \{V_{i-1}(R)\} \bigcup \left\{\frac{s-H_{i-1}}{\Pi_{i-1}(R)} \mid s \in S_{i-2}(R)\right\}$, we have $\mathbb{S}_{i-1} \neq \emptyset$ iff $B'_i \leq \min\{S_{i-1}(R)\}$.

Proof for Statement 2. We show by induction on the value of n.

- Base case: when $n = 2$, basic algebraic manipulation gives that

$$\begin{aligned}\{(R, \boldsymbol{b}') \mid B_2' \le V_1(R), B_2' \in \mathbb{B}_i\} &= \left\{(R, \boldsymbol{b}') \mid B_2' \in \left[s_1^{(2)}, \min\left\{V_1(R), B_3' + \frac{b_2(R - R_2)}{R - R_3}\right\}\right]\right\} \\ &= \{(R, \boldsymbol{b}') \mid B_2' \in \mathbb{S}_2\}\end{aligned}$$

- Induction step: suppose $\{(R, \boldsymbol{b}') \mid B_2' \le V_1(R), B_i' \in \mathbb{B}_i, \forall\ 2 \le i \le k\} = \{(R, \boldsymbol{b}') \mid B_k' \in \mathbb{S}_k\}$. Then we have

$$\begin{aligned}&\{(R, \boldsymbol{b}') \mid B_2' \le V_1(R), B_i' \in \mathbb{B}_i, \forall\ 2 \le i \le k+1\} \\ =&\{(R, \boldsymbol{b}') \mid B_2' \le V_1(R), B_i' \in \mathbb{B}_i, \forall\ 2 \le i \le k\} \bigcap \left\{(R, \boldsymbol{b}') \mid B_{k+1}' \in \mathbb{B}_{k+1}\right\} \\ =&\{(R, \boldsymbol{b}') \mid B_k' \in \mathbb{S}_k\} \bigcap \left\{(R, \boldsymbol{b}') \mid B_{k+1}' \in \mathbb{B}_{k+1}\right\} \\ =&\left\{(R, \boldsymbol{b}') \mid B_{k+1}' \in \mathbb{S}_{k+1}\right\}\end{aligned}$$

  where the last equation comes from Statement 1.

- Conclusion: by the principle of induction, we have

$$\{(R, \boldsymbol{b}') \mid B_2' \le V_1(R), B_i' \in \mathbb{B}_i, \forall\ 2 \le i \le n\} = \{(R, \boldsymbol{b}') \mid B_n' \in \mathbb{S}_n\}.$$

□

Observe from the proof of Lemma 37 that for all $2 \le i \le n$, setting $B_i' = \max\left\{\Pi_i(R) B_{i+1}' + H_i,\ B_{i+1}'\right\}$ gives $\tilde{R}_s^*$. Combining it with Lemma 36, we know that R^* is the optimal solution for **OPT_S**, and therefore we have Proposition 16. Besides, since $\tilde{R}_s^*$ can be achieved by setting $b_i' = \max\left\{\frac{r_i B_{i+1}'}{R - R_{i+1}} + H_i,\ 0\right\}$, we then have Proposition 17.

B.4 Proofs for FIFO Scheduler

B.4.1 Proof for Proposition 18

PROPOSITION 18. *Consider a system with n rate-controlled flows with traffic envelopes (r_i, b_i), where $1 \leq i \leq n$, which share a FIFO link with bandwidth $R \geq R_1 = \sum_{j=1}^{n} r_j$. Assume that the system reshapes flow i's traffic envelope to (r_i, b_i'). Then the worst-case delay for flow i is*

$$\widehat{D}_i^* = \max\left\{\frac{b_i - b_i'}{r_i} + \frac{\sum_{j\neq i} b_j'}{R}, \frac{\sum_{j=1}^{n} b_j'}{R} + \frac{(b_i - b_i')R_1}{r_i R}\right\}. \tag{3.10}$$

Remark: we will hardly rely on network calculus to show Proposition 18. This is because current work regarding aggregate multiplexing is not very rich, and none of the existing work provides a tight bound for our system.

Proof. W.l.o.g we consider the worst-case delay for flow 1. First we show that there always exists a traffic pattern such that flow 1's worst-case delay can be achieved by the last bit inside a burst of size b_1, and then we characterize the worst-case delay for that b_1^{th} bit.

- Consider the traffic pattern $\boldsymbol{T}(t) = \{T_1(t), ..., T_n(t)\}$ that realizes flow 1's worst-case delay, where $T_i(t)$ is right continuous and specifies the cumulative amount of data sent by flow i during time $[0, t]$. Suppose the worst-case delay is achieved at t_0, and at t_0 flow 1 sends a burst of $b \leq b_1$. As under FIFO the last bit gets a strictly larger delay compared with all the other bits inside the burst, the b^{th} bit sent at t_0 achieves the worst-case delay.

If $b = b_1$, flow 1's worst-case delay is achieved by the b_1^{th} bit inside a burst, *i.e.,* $\boldsymbol{T}(t)$ is the traffic pattern we want. Afterwards we consider the case $b < b_1$.

First note that b is the maximum amount of data flow 1 can send at t_0 without violating its arrival-curve constraint, *i.e.,* there exists $t_s \in [0, t_0)$ such that $T(t_0) - T(t_s) = b_1 + r_1(t_0 - t_s)$. Otherwise, we can produce a worse delay by increasing b to the maximum value that remains conformant with the arrival-curve constraint. Next we show that if $b < b_1$, there exists $T_1'(t)$ sending a burst of b_1 at t_0, such that under $\boldsymbol{T}'(t) = \{T_1'(t), ..., T_n(t)\}$ the last bit flow 1 sends at t_0 also achieves flow 1's worst-case delay.

Define $\hat{t} = \sup\{t \mid T_1(t_0) - T(t) \geq b_1\}$. As $b < b_1$, $\hat{t} \in (t_s, t_0)$. Note that $T_1(t_0) - T_1(\hat{t}) \leq b_1$ as $T_1(t)$ is right continuous. Define

$$T_1'(t) = \begin{cases} T_1(t), & \text{when } t < \hat{t} \\ T_1(t_0) - b_1, & \text{when } \hat{t} \leq t < t_0 \\ T_1(t_0), & \text{otherwise} \end{cases}$$

which sends a burst of b_1 at t_0. Note that $T_1'(t)$ satisfies flow 1's arrival curve. Specifically, consider any $0 \leq t^{(1)} < t^{(2)}$.

- When $t^{(2)} < \hat{t}$, it has $T_1'\left(t^{(2)}\right) - T_1'\left(t^{(1)}\right) = T_1\left(t^{(2)}\right) - T_1\left(t^{(1)}\right)$.
- When $\hat{t} \leq t^{(2)} < t_0$,
  - If $t^{(1)} < \hat{t}$, $T_1'\left(t^{(2)}\right) - T_1'\left(t^{(1)}\right) = T_1(t_0) - b_1 - T_1\left(t^{(1)}\right) \leq T_1(\hat{t}) - T_1\left(t^{(1)}\right) \leq T_1\left(t^{(2)}\right) - T_1\left(t^{(1)}\right)$
  - If $\hat{t} \leq t^{(1)} < t_0$, $T_1'\left(t^{(2)}\right) - T_1'\left(t^{(1)}\right) = T_1(t_0) - b_1 - [T_1(t_0) - b_1] = 0$
- When $t^{(2)} \geq t_0$,
  - If $t^{(1)} \leq \hat{t}$, $T_1'\left(t^{(2)}\right) - T_1'\left(t^{(1)}\right) = T_1(t_0) - T_1\left(t^{(1)}\right) \leq T_1\left(t^{(2)}\right) - T_1\left(t^{(1)}\right)$

* If $\hat{t} < t^{(1)} < t_0$, $T_1'\left(t^{(2)}\right) - T_1'\left(t^{(1)}\right) = T_1(t_0) - [T_1(t_0) - b_1] = b_1 \leq b_1 + r_1\left(t^{(2)} - t^{(1)}\right)$

* If $t^{(1)} \geq t_0$, $T_1'\left(t^{(2)}\right) - T_1'\left(t^{(1)}\right) = T_1(t_0) - T_1(t_0) = 0$

We then show that under $\boldsymbol{T}'(t)$ the last bit sent at t_0 also achieves flow 1's worst-case delay. First observe that under $\boldsymbol{T}'(t)$ the last bit sent at t_0 arrives at the shared link no earlier than that under $\boldsymbol{T}(t)$. This is because it arrives at the reshaper later than under $T(t)$ and experiences a no smaller reshaping delay upon its arrival. Particularly, given the shaper's service curve of $\begin{cases} b_1' + r_1 t, & \text{when } t > 0 \\ 0 & \text{otherwise} \end{cases}$, the b_1^{th} bit of a burst gets a delay of $\frac{b_1 - b_1'}{r_1}$, which equals the worst-case delay for flow 1 inside the shaper. Next we show that under $\boldsymbol{T}'(t)$ the last bit leaves the shared link no earlier than that under $\boldsymbol{T}(t)$, *i.e.,* overall it gets a no smaller delay under $\boldsymbol{T}'(t)$. Since under $\boldsymbol{T}(t)$ the last bit achieves the worst-case delay, so does under $\boldsymbol{T}(t)$.

Suppose the last bit arrives at the shared link at $\hat{t}_0$ under $\boldsymbol{T}(t)$, and at $\hat{t}_0' \geq \hat{t}_0$ under $\boldsymbol{T}'(t)$. If under $\boldsymbol{T}(t)$ the last bit arrives to find the shared link with an empty queue, *i.e.,* it has no delay at the shared link, then combining it with $\hat{t}_0' \geq \hat{t}_0$ gives what we want. Afterwards, we consider the case where under $\boldsymbol{T}(t)$ the last bit arrives at the shared link with a non-empty queue, *i.e.,* there exists $\hat{t}_s < \hat{t}_0$ and $\delta > 0$ such that the shared link processes data at full speed R during $[\hat{t}_s, \hat{t}_0]$, and at a speed strictly less than R during $[\hat{t}_s - \delta, \hat{t}_s)$.

Under $\boldsymbol{T}(t)$, $T_1(\hat{t}_s, \hat{t}_0)$ data from flow 1 arrives at the shared link during $[\hat{t}_s, \hat{t}_0]$. Then the last bit leaves the shared link at time $\hat{t}_e = \hat{t}_s + \frac{\sum_{j \geq 2}[T_j(\hat{t}_0) - T_j(\hat{t}_s)] + T_1(\hat{t}_s, \hat{t}_0)}{R}$. Whereas under $\boldsymbol{T}'(t)$, suppose there is $M \geq 0$ amount of data in the buffer at $\hat{t}_s$, and $T_1'(\hat{t}_s, \hat{t}_0')$ amount of data from flow 1 arrives at the shared link during $[\hat{t}_s, \hat{t}_0']$. As $T_1'(t)$ delays some data to t_0, and by $\hat{t}_0'$ all of the delayed data arrives at the shared link, it has

$T_1'(\hat{t}_s, \hat{t}_0') \geq T_1(\hat{t}_s, \hat{t}_0)$. Then the b_1^{th} bit leaves the shared link at a time no less than $\hat{t}_s + \frac{M+\sum_{j\geq 2}[T_j(\hat{t}_0')-T_j(\hat{t}_s)]+T_1'(\hat{t}_s,\hat{t}_0')}{R}$, which is no less than $\hat{t}_e$.

- Next we characterize the worst-case delay. Given that there always exists a traffic pattern that the b_1^{th} bit of a burst gives flow 1's worst-case delay, w.l.o.g we assume the worst-case delay to be achieved by the b_1^{th} bit sent at t_0.

  Remember that the b_1^{th} bit of a burst gets a delay of $\frac{b_1-b_1'}{r_1}$ inside the shaper. Next we consider the delay at the shared link. Suppose the b_1^{th} bit arrives at the shared link at $\hat{t}_0$. If at $\hat{t}_0$ the shared link processes at a speed strictly less than R, then the b_1^{th} bit gets no delay at the shared link, and therefore gets a overall worst-case delay of $\frac{b_1-b_1'}{r_1}$. Otherwise, suppose the last busy period at the shared link starts at $0 \leq \hat{t}_s \leq \hat{t}_0$.

  1. When $\hat{t}_0 - \hat{t}_s \geq \frac{b_1-b_1'}{r_1}$, during $[\hat{t}_s, \hat{t}_0]$ at most $\sum_{j=1}^n b_j' + \sum_{j=1}^n r_j\left(\hat{t}_0 - \hat{t}_s\right)$ amount of data arrives at the shared link. Thus, the delay for the b_1^{th} bit at the shared link is $\frac{\sum_{j=1}^n b_j' + \sum_{j=1}^n r_j\left(\hat{t}_0-\hat{t}_s\right)}{R} - \hat{t}_0$, which decreases with $\hat{t}_0$ since $R \geq \sum_{j=1}^n r_j$. Given $\hat{t}_0 \geq \frac{b_1-b_1'}{r_1} + \hat{t}_s$, the worst-case delay at the shared link is achieved at $\hat{t}_0 = \frac{b_1-b_1'}{r_1} + \hat{t}_s$, and has a value of $\frac{\sum_{j=1}^n b_j' + \frac{b_1-b_1'}{r_1}\sum_{j=1}^n r_j}{R} - \frac{b_1-b_1'}{r_1} - \hat{t}_s$. Thus the overall worst-case delay is achieved at $\hat{t}_s = 0$, with a value of

  $$\frac{\sum_{j=1}^n b_j' + \frac{b_1-b_1'}{r_1}\sum_{j=1}^n r_j}{R} - \frac{b_1-b_1'}{r_1} + \frac{b_1-b_1'}{r_1} = \frac{\sum_{j=1}^n b_j'}{R} + \frac{(b_1-b_1')R_1}{r_1 R}.$$

  2. When $\hat{t}_0 - \hat{t}_s < \frac{b_1-b_1'}{r_1}$, as flow 1's burst of b_1' arrived at the shared link before $\hat{t}_0 - \frac{b_1-b_1'}{r_1}$, it should has been cleared before $\hat{t}_s$. Hence, during $[\hat{t}_s, \hat{t}_0]$ at most $\sum_{j\neq 1}^n b_j' + \sum_{j=1}^n r_j\left(\hat{t}_0 - \hat{t}_s\right)$ data arrive at the shared link. Therefore, the b_1^{th} bit gets a delay of $\frac{\sum_{j\neq 1}^n b_j' + \sum_{j=1}^n r_j\left(\hat{t}_0-\hat{t}_s\right)}{R} - \hat{t}_0$ at the shared link, which decreases with $\hat{t}_0$ under $R \geq \sum_{j=1}^n r_j$. Consequently, its worst-case delay is achieved at $\hat{t}_0 = \hat{t}_s$, with

a value of $\frac{\sum_{j\neq 1}^{n} b'_j}{R} - \hat{t}_s$, which is maximized at $\hat{t}_s = 0$. Thus the overall worst-case delay is $\frac{b_1 - b'_1}{r_1} + \frac{\sum_{j\neq 1}^{n} b'_j}{R}$.

Combining case 1 and 2 gives flow 1's worst-case delay, *i.e.,*

$$\widehat{D}_1^* = \max\left\{ \frac{b_1 - b'_1}{r_1} + \frac{\sum_{j\neq 1} b'_j}{R}, \frac{\sum_{j=1}^{n} b'_j}{R} + \frac{(b_1 - b'_1)R_1}{r_1 R} \right\}.$$

□

B.4.2 Proofs for Proposition 19 and 20

This part provides the solution for **OPT_F**, from which we then have Proposition 19 and 20. For the reader's convenience, we restate **OPT_F**, where $R_1 = \sum_{i=1}^{n} r_i$.

$$\begin{aligned}
\textbf{OPT_F} \quad & \min_{\boldsymbol{b'}} R \\
\text{s.t} \quad & \max\left\{ \frac{b_i - b'_i}{r_i} + \frac{\sum_{j\neq i} b'_j}{R}, \frac{\sum_{j=1}^{n} b'_j}{R} + \frac{(b_i - b'_i)R_1}{r_i R} \right\} \leq d_i, \quad \forall\ 1 \leq i \leq n, \\
& R_1 \leq R, \quad 0 \leq b'_i \leq b_i, \qquad\qquad \forall\ 1 \leq i \leq n.
\end{aligned}$$

Lemma 38. *For* $1 \leq i \leq n$ *define* $T_i^{(1)} = \frac{R}{R+r_i}\left(H_i + \frac{r_i}{R}\widehat{B}'_n\right)$ *and* $T_i^{(2)} = b_i + \frac{r_i(\widehat{B}'_n - Rd_i)}{R_1}$. *Denote* $\widehat{B}'_0 = 0$ *and* $\widehat{\boldsymbol{B}}' = (\widehat{B}'_1, ...\widehat{B}'_n)$. *Consider the following optimization:*

$$\begin{aligned}
\textbf{\textit{OPT_F'}} \quad & \min_{\widehat{\boldsymbol{B}}'} R \\
s.t \quad & \max\left\{0, T_i^{(1)}, T_i^{(2)}\right\} \leq \widehat{B}'_i - \widehat{B}'_{i-1} \leq b_i, \quad \forall\ 1 \leq i \leq n, \\
& R_1 \leq R.
\end{aligned}$$

Suppose the optimal solution for ***OPT_F'*** *is* $(R^*, \widehat{\boldsymbol{B}}'^*)$*. Then* $(R^*, \boldsymbol{b}'^*)$*, where*

$$\boldsymbol{b}'^* = (\widehat{B}'^*_1, \widehat{B}'^*_2 - \widehat{B}'^*_1, ..., \widehat{B}'^*_n - \widehat{B}'^*_{n-1})$$

is an optimal solution for ***OPT_F****.*

Proof. Define $\widehat{B}_i = \sum_{j=1}^{i} b_j$. From basic algebraic manipulation **OPT_F** is equivalent to **OPT_F"**:

$$\begin{aligned} \textbf{OPT\_F"} \quad & \min_{\boldsymbol{b'}} R \\ \text{s.t} \quad & \max\left\{0, T_i^{(1)}, T_i^{(2)}\right\} \leq \widehat{B}'_i - \widehat{B}'_{i-1} \leq b_i \quad \forall\ 1 \leq i \leq n, \\ & R_1 \leq R. \end{aligned}$$

As the constraints of **OPT_F'** and **OPT_F"** are the same, the two optimizations share the same optimal value R^*. From $\widehat{B}_i = \sum_{j=1}^{i} b_j$, $\boldsymbol{b}'^* = (\widehat{B}'^*_1, \widehat{B}'^*_2 - \widehat{B}'^*_1, ..., \widehat{B}'^*_n - \widehat{B}'^*_{n-1})$ is then the optimal variable for **OPT_F"**. Hence, $(R^*, \boldsymbol{b}'^*)$ is an optimal solution for **OPT_F"**, and therefore an optimal solution for **OPT_F**. □

Next we proceed to solve **OPT_F'**, combining which with Lemma 38 then gives Proposition 19 and 20. Define $\mathbb{Z}_i = \{1 \leq j \leq i \mid j \in \mathbb{Z}\}$ for $1 \leq i \leq n$,

$$X_F(R) = \max_{P_1, P_2 \subseteq \mathbb{Z}_n, P_2 \neq \mathbb{Z}_n, P_1 \bigcap P_2 = \emptyset} \frac{\sum_{i \in P_1} \frac{RH_i}{R + r_i} + \sum_{i \in P_2} \left(b_i - \frac{r_i d_i R}{R_1}\right)}{1 - \sum_{i \in P_1} \frac{r_i}{R + r_i} - \sum_{i \in P_2} \frac{r_i}{R_1}},$$

and

$$Y_F(R) = \min_{1 \leq i \leq n-1} \left\{ \widehat{B}_n, Rd_n, \min_{P_1, P_2 \subseteq \mathbb{Z}_i, P_1 \bigcap P_2 = \emptyset, P_1 \bigcup P_2 \neq \emptyset} \left\{ \frac{\widehat{B}_i - \sum_{j \in P_1} \frac{RH_j}{R + r_j} - \sum_{j \in P_2} \left(b_j - \frac{r_j d_j R}{R_1}\right)}{\sum_{j \in P_1} \frac{r_j}{R + r_j} + \sum_{j \in P_2} \frac{r_j}{R_1}} \right\} \right\},$$

where $\widehat{B}_i = \sum_{j=1}^{i} b_j$ for $1 \le i \le n$. Then from Lemma 39 we know that R forms a feasible solution for **OPT_F'** iff $R \ge \max\left\{R_1, \frac{\widehat{B}_n R_1}{\sum_{i=1}^{n} r_i d_i}\right\}$ and $X_F(R) \le Y_F(R)$. Therefore, R^* is the minimum value satisfying these conditions.

Given R^*, from Lemma 39 we know that there exists an optimal solution with $\widehat{B}'^*_n = X_F(R^*)$. From Statement 2 in Lemma 40, we know that suppose there exists an optimal solution with $\widehat{B}'_n = \widehat{B}'^*_n$ and $\widehat{B}'_i = \widehat{B}'^*_i$ where $2 \le i \le n$, then there exists an optimal solution with $\widehat{B}'_n = \widehat{B}'^*_n$, $\widehat{B}'_i = \widehat{B}'^*_i$, and $\widehat{B}'_{i-1} = \max\left\{\sum_{j=1}^{i-1} T_j, \widehat{B}'^*_i - b_i\right\}$. Thus, based on $\widehat{B}'^*_n$ we can sequentially compute $\widehat{B}'^*_i$ from $i = n-1$ to $i = 1$.

Lemma 39. *Define* $T_i = \max\left\{0, T_i^{(1)}, T_i^{(2)}\right\}$. *Then* $\left\{\widehat{\boldsymbol{B}}' \mid T_h \le \widehat{B}'_h - \widehat{B}'_{h-1} \le b_h,\ \forall 1 \le h \le n\right\} \neq \emptyset$ *iff all of the following conditions hold:*

- $R \ge \frac{\sum_{i=1}^{n} b_i R_1}{\sum_{i=1}^{n} r_i d_i} = \frac{\widehat{B}_n R_1}{\sum_{i=1}^{n} r_i d_i}$
- $\widehat{B}'_n \ge \max_{P_1, P_2 \subseteq \mathbb{Z}_n, P_2 \neq \mathbb{Z}_n, P_1 \bigcap P_2 = \emptyset} \frac{\sum_{i \in P_1} \frac{RH_i}{R+r_i} + \sum_{p \in P_2}\left(b_i - \frac{r_i d_i R}{R_1}\right)}{1 - \sum_{i \in P_1} \frac{r_i}{R+r_i} - \sum_{i \in P_2} \frac{r_i}{R_1}}$,
- $\widehat{B}'_n \le \min_{1 \le i \le n-1}\left\{\widehat{B}_n, Rd_n, \min_{P_1, P_2 \subseteq \mathbb{Z}_i, P_1 \bigcap P_2 = \emptyset, P_1 \bigcup P_2 \neq \emptyset}\left\{\frac{\widehat{B}_i - \sum_{j \in P_1} \frac{RH_j}{R+r_j} - \sum_{j \in P_2}\left(b_j - \frac{r_j d_j R}{R_1}\right)}{\sum_{j \in P_1} \frac{r_j}{R+r_j} + \sum_{j \in P_2} \frac{r_j}{R_1}}\right\}\right\}$

Proof. Define $\widehat{B}_i = \sum_{j=1}^{i} b_j$ for $1 \le i \le n$, and define $\widehat{\mathbb{B}}_i = \left[\max\left\{T_i, \widehat{B}'_{i+1} - b_{i+1}\right\},\ \min\left\{\widehat{B}_i, \widehat{B}'_{i+1} - T_{i+1}\right\}\right]$ for $1 \le i \le n-1$. As $\widehat{B}'_0 = 0$, from basic algebraic manipulation $T_i = \max\left\{0, T_i^{(1)}, T_i^{(2)}\right\} \le \widehat{B}'_i - \widehat{B}'_{i-1} \le b_i,\ \forall 1 \le i \le n$ is equivalent to

$$\begin{cases} \widehat{B}'_1 \in \left[\max\left\{T_1, \widehat{B}'_2 - b_2\right\},\ \min\left\{b_1, \widehat{B}'_2 - T_2\right\}\right] = \widehat{\mathbb{B}}_1, \\ T_i \le \widehat{B}'_i - \widehat{B}'_{i-1} \le b_i,\ \forall\ 2 < i \le n \end{cases}$$

Define $\widehat{\boldsymbol{B}}'_i = \left\{\widehat{B}_i, ..., \widehat{B}_n\right\}$, and $\widehat{\mathbb{B}}^{(i)} = \left\{\widehat{\boldsymbol{B}}'_i \mid T_h \le \widehat{B}'_h - \widehat{B}'_{h-1} \le b_h,\ \forall i < h \le n\right\}$. Then we have

$$\left\{\boldsymbol{B}' \mid T_i \le \widehat{B}'_i - \widehat{B}'_{i-1} \le b_i,\ \forall 1 \le i \le n\right\} \neq \emptyset \iff \widehat{\mathbb{B}}_1 \neq \emptyset \text{ and } \widehat{\mathbb{B}}^{(2)} \neq \emptyset,$$

which from Lemma 40 is equivalent to

$$\left\{\sum_{j=1}^{n} T_j \le \widehat{B}'_n \le \min\left\{\widehat{B}_n, Rd_n\right\} \mid \sum_{j=1}^{i} T_j \le \widehat{B}_i,\ \forall 1 \le i \le n-1\right\} \ne \emptyset. \tag{B.10}$$

Denote $\mathbb{Z}_i = \{1 \le j \le i \mid j \in \mathbb{Z}\}$ for $1 \le i \le n$, then

- $\sum_{j=1}^{n} T_j \le \widehat{B}'_n$ implies that for all $P_1, P_2 \subseteq \mathbb{Z}_n$ and $P_1 \bigcap P_2 = \emptyset$, $\sum_{i\in P_1} T_i^{(1)} + \sum_{i\in P_2} T_i^{(2)} \le B'_n$, *i.e.*, $\sum_{i\in P_1} \frac{RH_i}{R+r_i} + \sum_{p\in P_2}\left(b_i - \frac{r_i d_i R}{R_1}\right) \le \left(1 - \sum_{i\in P_1}\frac{r_i}{R+r_i} - \sum_{i\in P_2}\frac{r_i}{R_1}\right) B'_n$, which is equivalent to $R \ge \frac{\sum_{i=1}^{n} b_i R_1}{\sum_{i=1} r_i d_i}$ when $P_2 = \mathbb{Z}_n$, and $\widehat{B}'_n \ge \frac{\sum_{i\in P_1}\frac{RH_i}{R+r_i} + \sum_{p\in P_2}\left(b_i - \frac{r_i d_i R}{R_1}\right)}{1 - \sum_{i\in P_1}\frac{r_i}{R+r_i} - \sum_{i\in P_2}\frac{r_i}{R_1}}$ otherwise.

- For $1 \le i \le n-1$, $\sum_{j=1}^{i} T_j \le \widehat{B}_i$ implies that for all $P_1, P_2 \subseteq \mathbb{Z}_i$ and $P_1 \bigcap P_2 = \emptyset$, $\sum_{i\in P_1} T_i^{(1)} + \sum_{i\in P_2} T_i^{(2)} \le \widehat{B}_i$, *i.e.*, $\sum_{i\in P_1}\frac{RH_i}{R+r_i} + \sum_{p\in P_2}\left(b_i - \frac{r_i d_i R}{R_1}\right) + \left(\sum_{i\in P_1}\frac{r_i}{R+r_i} + \sum_{i\in P_2}\frac{r_i}{R_1}\right) B'_n \le \widehat{B}_i$, which is equivalent to $\widehat{B}'_n \le \frac{\widehat{B}_i - \sum_{j\in P_1}\frac{RH_j}{R+r_j} - \sum_{j\in P_2}\left(b_j - \frac{r_j d_j R}{R_1}\right)}{\sum_{j\in P_1}\frac{r_j}{R+r_j} + \sum_{j\in P_2}\frac{r_j}{R_1}}$.

Therefore, Eq. (B.10) holds iff

$$\begin{cases} R \ge \dfrac{\sum_{i=1}^{n} b_i R_1}{\sum_{i=1} r_i d_i} \\ \widehat{B}'_n \ge \max\limits_{P_1,P_2\subseteq\mathbb{Z}_n, P_2\ne\mathbb{Z}_n, P_1\bigcap P_2=\emptyset} \dfrac{\sum_{i\in P_1}\frac{RH_i}{R+r_i} + \sum_{i\in P_2}\left(b_i - \frac{r_i d_i R}{R_1}\right)}{1 - \sum_{i\in P_1}\frac{r_i}{R+r_i} - \sum_{i\in P_2}\frac{r_i}{R_1}} \\ \widehat{B}'_n \le \min\limits_{1\le i\le n-1}\left\{\widehat{B}_n, Rd_n, \min\limits_{P_1,P_2\subseteq\mathbb{Z}_i, P_1\bigcap P_2=\emptyset, P_1\bigcup P_2\ne\emptyset}\left\{\dfrac{\widehat{B}_i - \sum_{j\in P_1}\frac{RH_j}{R+r_j} - \sum_{j\in P_2}\left(b_j - \frac{r_j d_j R}{R_1}\right)}{\sum_{j\in P_1}\frac{r_j}{R+r_j} + \sum_{j\in P_2}\frac{r_j}{R_1}}\right\}\right\} \end{cases}$$

□

Next we establish Lemma 40. For $\widehat{\mathbb{B}}_1 = \left[\max\left\{T_1, \widehat{B}'_2 - b_2\right\},\ \min\left\{b_1, \widehat{B}'_2 - T_2\right\}\right]$, $\widehat{\boldsymbol{B}}'_2 = \left\{\widehat{B}'_2, ..., \widehat{B}'_n\right\}$, and $\widehat{\mathbb{B}}^{(2)} = \left\{\widehat{\boldsymbol{B}}'_2 \mid T_h \le \widehat{B}'_h - \widehat{B}'_{h-1} \le b_h,\ \forall 2 < h \le n\right\}$, we have

Lemma 40. $\widehat{\mathbb{B}}_1 \neq \emptyset$ *and* $\widehat{\mathbb{B}}^{(2)} \neq \emptyset$ *iff*

$$\left\{ \sum_{j=1}^{n} T_j \leq \widehat{B}'_n \leq \min\left\{ \widehat{B}_n, Rd_n \right\} \mid \sum_{j=1}^{i} T_j \leq \widehat{B}_i,\ \forall 1 \leq i \leq n-1 \right\} \neq \emptyset.$$

Proof. For $1 \leq i \leq n$ define $\widehat{B}_i = \sum_{j=1}^{i} b_j$ and $\widehat{\boldsymbol{B}}'_i = \left\{ \widehat{B}_i, ..., \widehat{B}_n \right\}$. For $1 \leq i \leq n-1$ further define $\widehat{\mathbb{B}}_i = \left[\max\left\{ \sum_{j=1}^{i} T_j, \widehat{B}'_{i+1} - b_{i+1} \right\},\ \min\left\{ \widehat{B}_i, \widehat{B}'_{i+1} - T_{i+1} \right\} \right]$ and $\widehat{\mathbb{B}}^{(i)} = \left\{ \widehat{\boldsymbol{B}}'_i \mid T_h \leq \widehat{B}'_h - \widehat{B}'_{h-1} \leq b_h,\ \forall i < h \leq n \right\}$. Suppose we have

Statement 1. $\widehat{\mathbb{B}}_1 \neq \emptyset$ and $\widehat{\mathbb{B}}^{(2)} \neq \emptyset$ iff 1) $\widehat{\mathbb{B}}_{n-1} \neq \emptyset$, 2) $\sum_{j=1}^{h} T_h \leq \widehat{B}_h$, for $1 \leq h \leq n-2$, and 3) $\widehat{B}'_n \leq Rd_{n-1}$.

Basic algebraic manipulations give that $\widehat{\mathbb{B}}_{n-1} \neq \emptyset$, *i.e.,* $\max\left\{ \sum_{j=1}^{n-1} T_j, \widehat{B}'_n - b_n \right\} \leq \min\left\{ \widehat{B}_{n-1}, \widehat{B}'_n - T_n \right\}$, iff 1) $\sum_{j=1}^{n} T_j \leq \widehat{B}'_n \leq \widehat{B}_n$, 2) $\sum_{j=1}^{n-1} T_j \leq \widehat{B}_{n-1}$, and 3) $\widehat{B}'_n \leq Rd_n$. Combining them with $\sum_{j=1}^{h} T_h \leq \widehat{B}_h$, for $1 \leq h \leq n-2$ and $\widehat{B}'_n \leq Rd_{n-1}$ gives $\sum_{j=1}^{n} T_j \leq \widehat{B}'_n \leq \min\left\{ \widehat{B}_n, Rd_n \right\}$ and $\sum_{j=1}^{h} T_h \leq \widehat{B}_h$, for $1 \leq h \leq n-1$. Therefore, we have Lemma 40.

Next, we show Statement 1 based on Statement 2. For convenience, define $\widehat{\mathbb{B}}^{(n)} = \{1\}$. Then we have

Statement 2. For $1 \leq i \leq n-2$, $\widehat{\mathbb{B}}_i \neq \emptyset$ and $\widehat{\mathbb{B}}^{(i+1)} \neq \emptyset$ iff 1) $\widehat{\mathbb{B}}_{i+1} \neq \emptyset$ and $\widehat{\mathbb{B}}^{(i+2)} \neq \emptyset$, 2) $\sum_{j=1}^{i} T_j \leq \widehat{B}_i$, and 3) $\widehat{B}'_n \leq Rd_{i+1}$.

Proof for Statement 1: we show Statement 1 by induction. For $1 \leq i \leq n-1$, define

$$S_i : \widehat{\mathbb{B}}_i \neq \emptyset,\ \widehat{\mathbb{B}}^{(i+1)} \neq \emptyset,\ \widehat{B}'_n \leq Rd_i,\ \text{and}\ \sum_{j=1}^{h} T_h \leq \widehat{B}_h, \forall 1 \leq h \leq i-1.$$

- When $i = 1$, Statement 2 directly gives that $\widehat{\mathbb{B}}_1 \neq \emptyset$ and $\widehat{\mathbb{B}}^{(2)} \neq \emptyset$ iff S_2 holds.

- When $i \geq 1$, suppose $\widehat{\mathbb{B}}_1 \neq \emptyset$ and $\widehat{\mathbb{B}}^{(2)} \neq \emptyset$ iff S_i holds, *i.e.,* 1) $\widehat{\mathbb{B}}_i \neq \emptyset$ and $\widehat{\mathbb{B}}^{(i+1)} \neq \emptyset$, 2) $\widehat{B}'_n \leq Rd_i$, and 3)$\sum_{j=1}^{h} T_h \leq \widehat{B}_h, \forall 1 \leq h \leq i-1$. Note that by Statement 2 we have $\widehat{\mathbb{B}}_i \neq \emptyset$ and $\widehat{\mathbb{B}}^{(i+1)} \neq \emptyset$ iff i) $\widehat{\mathbb{B}}_{i+1} \neq \emptyset$ and $\widehat{\mathbb{B}}^{(i+2)} \neq \emptyset$, ii) $\sum_{j=1}^{i} T_i \leq \widehat{B}_i$, and iii) $\widehat{B}'_n \leq Rd_{i+1}$. Thus, $\widehat{\mathbb{B}}_1 \neq \emptyset$ and $\widehat{\mathbb{B}}^{(2)} \neq \emptyset$ iff 1) $\widehat{\mathbb{B}}_{i+1} \neq \emptyset$ and $\widehat{\mathbb{B}}^{(i+2)} \neq \emptyset$, 2) $\widehat{B}'_n \leq \min\{Rd_i, Rd_{i+1}\} = Rd_{i+1}$, and 3) $\sum_{j=1}^{h} T_h \leq \widehat{B}_h, \forall 1 \leq h \leq i$, *i.e.,* S_{i+1} holds.

Thus, we have $\widehat{\mathbb{B}}_1 \neq \emptyset$ and $\widehat{\mathbb{B}}^{(2)} \neq \emptyset$ iff S_{n-1} holds: $\widehat{\mathbb{B}}_{n-1} \neq \emptyset$, $\widehat{\mathbb{B}}^{(n)} = \{1\} \neq \emptyset$, $\widehat{B}'_n \leq Rd_{n-1}$, and $\sum_{j=1}^{h} T_h \leq \widehat{B}_h, \forall 1 \leq h \leq n-2$, which then gives Statement 1.

Proof for Statement 2: Consider $\widehat{\mathbb{B}}_i \neq \emptyset$ and $\widehat{\mathbb{B}}^{(i+1)} \neq \emptyset$.

- $\widehat{\mathbb{B}}_i \neq \emptyset$ iff $\max\left\{\sum_{j=1}^{i} T_j, \widehat{B}'_{i+1} - b_{i+1}\right\} \leq \min\left\{\widehat{B}_i, \widehat{B}'_{i+1} - T_{i+1}\right\}$, which from basic algebraic manipulation is equivalent to 1) $\sum_{j=1}^{i+1} T_j \leq \widehat{B}'_{i+1} \leq \widehat{B}_{i+1}$, 2) $\sum_{j=1}^{i} T_j \leq \widehat{B}_i$, and 3) $b_{i+1} \geq T_{i+1} \iff \widehat{B}'_n \leq Rd_{i+1}$.

- Consider $\widehat{\mathbb{B}}^{(i+1)} \neq \emptyset$. When $i < n-2$, from basic algebraic manipulation it is equivalent to 1) $T_{n+2} \leq \widehat{B}'_{i+2} - \widehat{B}'_{i+1} \leq b_{i+2}$ and 2) $\widehat{\mathbb{B}}^{(i+2)} \neq \emptyset$. When $i = n-2$, $\widehat{\mathbb{B}}^{(i+1)} = \left\{\widehat{\boldsymbol{B}}'_{n-1} \mid T_n \leq \widehat{B}'_n - \widehat{B}'_{n-1} \leq b_n\right\}$, which is non-empty iff $T_n \leq \widehat{B}'_n - \widehat{B}'_{n-1} \leq b_n$. Since $\widehat{\mathbb{B}}^{n} = \{1\}$, it also has $\widehat{\mathbb{B}}^{(i+2)} \neq \emptyset$.

Note that $\sum_{j=1}^{i+1} T_j \leq \widehat{B}'_{i+1} \leq \widehat{B}_{i+1}$ and $T_{n+2} \leq \widehat{B}'_{i+2} - \widehat{B}'_{i+1} \leq b_{i+2}$ iff $\widehat{\mathbb{B}}_{i+1} \neq \emptyset$. Thus we have Statement 2. □

B.5 On the Benefit of Grouping Flows With Different Deadlines

In this section, we explore scenarios that consist of two flows sharing a common link whose access is arbitrated by a static priority scheduler. The goal is to identify configurations that minimize the link bandwidth required to meet the flows' deadlines. Of particular interest is assessing when the two flows should be assigned different priorities or instead merged into the same priority class.

Recalling the discussion of Section 3.3.5, specializing Propositions 16 and 19 to two flows, we find that the minimum required bandwidth for the two-flow scenario under static priority + shaping is

$$\widetilde{R}_s^* = \begin{cases} max\left\{r_1 + r_2, \frac{b_2}{d_2}, \frac{b_1 + b_2 - r_2 d_2}{d_1} + r_2\right\}, & \text{when } \frac{b_2}{r_2} \geq \frac{b_1}{r_1} \\ max\left\{r_1 + r_2, \frac{b_2}{d_2}, \frac{b_1 + \max\{b_2 - r_2 d_2, 0\}}{d_1} + r_2\right\}, & \text{otherwise} \end{cases} \tag{3.15}$$

and that under fifo + shaping it is

$$\widehat{R}_s^* = \max\left\{r_1 + r_2, \frac{b_2}{d_2}, \frac{(b_1 + b_2)(r_2 + r_2)}{d_1 r_1 + d_2 r_2}, \frac{b_1 + b_2 - d_1 r_1 + \sqrt{(b_1 + b_2 - d_1 r_1)^2 + 4 r_1 d_2 b_2}}{2 d_2}\right\}. \tag{3.17}$$

Comparing them gives

Proposition 41. *For the two-flow scenario,* $\widetilde{R}_s^* > \widehat{R}_s^*$ *iff*

$$d_1 \in \left(\frac{b_2}{r_2}, \frac{b_1}{r_1}\right) \text{ and } d_2 \in \left(\frac{(b_1 + b_2)(r_1 + r_2)}{r_2(b_1/d_1 + r_2)} - \frac{d_1 r_1}{r_2}, d_1\right).$$

Proof. When $\widetilde{R}_s^* = max\left\{r_1 + r_2, \frac{b_2}{d_2}\right\}$, from Eq. (3.16) $\widetilde{R}_s^* \leq \widehat{R}_s^*$. Below we consider 1) $\frac{b_2}{d_2} \geq \frac{b_1}{d_1}$ and 2) $\frac{b_2}{d_2} < \frac{b_1}{d_1}$ separately under $\widetilde{R}_s^* > max\left\{r_1 + r_2, \frac{b_2}{d_2}\right\}$.

1. When $\frac{b_2}{d_2} \geq \frac{b_1}{d_1}$, we show that $\widetilde{R}_s^* \leq \widehat{R}_s^*$. Specifically, $\widetilde{R}_s^* > max\left\{r_1 + r_2, \frac{b_2}{d_2}\right\}$ iff $\frac{b_1+b_2-r_2d_2}{d_1} + r_2 > \max\left\{r_1 + r_2, \frac{b_2}{d_2}\right\}$, which from basic algebraic manipulation equivalents to $b_1 + b_2 > r_1d_1 + r_2d_2$ and $d_2 < \frac{b_1+b_2-\sqrt{(b_1+b_2)^2-4r_2b_2d_1}}{2r_2}$. Next we show that under $b_1 + b_2 > r_1d_1 + r_2d_2$, it has $\frac{(b_1+b_2)(r_2+r_2)}{d_1r_1+d_2r_2} \geq \frac{b_1+b_2-r_2d_2}{d_1} + r_2$, and therefore $\widetilde{R}_s^* \leq \widehat{R}_s^*$.

   Consider $f(d_2) = \frac{(b_1+b_2)(r_2+r_2)}{d_1r_1+d_2r_2} - \frac{b_1+b_2-r_2d_2}{d_1} - r_2$, which equals 0 when $d_2 = d_1$. Basic algebraic gives that

$$\begin{aligned}
\frac{df(d_2)}{dd_2} &= \frac{r_2}{d_1} - \frac{(b_1+b_2)(r_2+r_2)r_2}{(d_1r_1+d_2r_2)^2} \\
&= \frac{r_2}{(d_1r_1+d_2r_2)^2}\left(\frac{(d_1r_1+d_2r_2)^2}{d_1} - (b_1+b_2)(r_2+r_2)\right) \\
&\leq \frac{r_2}{(d_1r_1+d_2r_2)^2}\left(\frac{(d_1r_1+d_2r_2)^2}{d_1} - (d_1r_1+d_2r_2)(r_2+r_2)\right) \\
&= \frac{r_2^2(d_2-d_1)}{d_1(d_1r_1+d_2r_2)} \leq 0
\end{aligned}$$

   Thus, for all $d_2 \leq d_1$, it has $f(d_2) \geq f(d_1) = 0$, *i.e.*, $\frac{(b_1+b_2)(r_2+r_2)}{d_1r_1+d_2r_2} \geq \frac{b_1+b_2-r_2d_2}{d_1} + r_2$.

2. When $\frac{b_2}{d_2} < \frac{b_1}{d_1}$, we show that $\widetilde{R}_s^* > \widehat{R}_s^*$ iff $d_1 \in \left(\frac{b_2}{r_2}, \frac{b_1}{r_1}\right)$ and $d_2 \in \left(\frac{(b_1+b_2)(r_1+r_2)}{r_2(b_1/d_1+r_2)} - \frac{d_1r_1}{r_2}, d_1\right)$. Specifically, $\widetilde{R}_s^* > max\left\{r_1 + r_2, \frac{b_2}{d_2}\right\}$ iff $\frac{b_1+\max\{b_2-r_2d_2,0\}}{d_1} + r_2 > \max\left\{r_1 + r_2, \frac{b_2}{d_2}\right\}$, which from basic algebraic manipulation equivalents to

$$\begin{cases}
\dfrac{b_1+b_2-r_2d_2}{d_1} + r_2 > \max\left\{r_1 + r_2, \dfrac{b_2}{d_2}\right\}, & \text{when } d_2 \leq \dfrac{b_2}{r_2} \\
\dfrac{b_1}{d_1} + r_2, & \text{otherwise .}
\end{cases}$$

   When $d_2 \leq \frac{b_2}{r_2}$, similar as before we have $\widetilde{R}_s^* \leq \widehat{R}_s^*$. When $d_2 > \frac{b_2}{r_2}$, basic algebraic manipulation gives that $\frac{b_1}{d_1} + r_2 > max\left\{r_1 + r_2, \frac{b_2}{d_2}\right\}$ iff $d_1 < \frac{b_1}{r_1}$. Combining them

gives that $\widetilde{R}_s^* \leq \widehat{R}_s^*$ when $(d_1, d_2) \notin \left\{(d_1, d_2) \mid \frac{b_2}{r_2} < d_2 < d_1 < \frac{b_1}{r_1}\right\}$. When $(d_1, d_2) \in \left\{(d_1, d_2) \mid \frac{b_2}{r_2} < d_2 < d_1 < \frac{b_1}{r_1}\right\}$, basic algebraic manipulation gives that $\frac{(b_1+b_2)(r_2+r_2)}{d_1 r_1 + d_2 r_2} > \frac{b_1+b_2-d_1 r_1+\sqrt{(b_1+b_2-d_1 r_1)^2+4 r_1 d_2 b_2}}{2 d_2}$ and $r_1 + r_2 > \frac{b_2}{d_2}$, *i.e.,*

$$\widehat{R}_s^* = \max\left\{r_1 + r_2, \frac{(b_1+b_2)(r_2+r_2)}{d_1 r_1 + d_2 r_2}\right\} = \begin{cases} r_1 + r_2, & \text{if } b_1 + b_2 \leq r_1 d_1 + r_2 d_2 \\ \dfrac{(b_1+b_2)(r_2+r_2)}{d_1 r_1 + d_2 r_2}, & \text{otherwise} \end{cases}$$

When $b_1 + b_2 \leq r_1 d_1 + r_2 d_2$, it has $\widetilde{R}_s^* > \widehat{R}_s^*$. Otherwise, basic algebraic manipulation gives that $\widetilde{R}_s^* > \widehat{R}_s^*$ iff $d_2 > \frac{(b_1+b_2)(r_1+r_2)}{r_2(b_1/d_1+r_2)} - \frac{d_1 r_1}{r_2} := g(d_1)$. Note that when $\frac{b_2}{r_2} < d_1 < \frac{b_1}{r_1}$, $g(d_1) > \frac{b_2}{r_2}$; and when $d_1 = \frac{b_2}{r_2}$ or $d_1 = \frac{b_1}{r_1}$, $g(d_1) = d_1$. Therefore, $\widetilde{R}_s^* > \widehat{R}_s^*$ iff $d_2 \in \left(\frac{(b_1+b_2)(r_1+r_2)}{r_2(b_1/d_1+r_2)} - \frac{d_1 r_1}{r_2}, d_1\right)$.

□

B.6 Extensions to Packet-Based Models in the Two-Flow Static Priority Case

In this section, we consider a more general packet-based model, where flow i has maximum packet size of l_i and the scheduler relies on static priorities. For ease of exposition, we only consider scenarios that consist of $n = 2$ flows, and consequently two priority classes (low and high).

We first characterize in Proposition 42 the worst-case delay of high-priority packets, and use the result to identify a condition for when adding a reshaper can help lower the required bandwidth (Corollary 43). We also confirm (Proposition 44) the intuitive property that reshaping the low-priority flow does not contribute to lowering the required bandwidth. We

then proceed to characterize in Proposition 45 the worst-case delay of low-priority packets. The results of Propositions 42 and 45 are used to formulate an optimization, **OPT_2**, that seeks to identify the optimum reshaping parameters for the high-priority flow that minimizes the link bandwidth required to meet the flows' deadlines. The bulk the section is devoted to solving this optimization, while also establishing the intermediate result that the optimal reshaping can be realized simply reducing the flow's burst size, *i.e.,* keeping its rate constant.

Returning to our two-flow, packet-based scenario, consider two token-bucket controlled flows (r_1, b_1) and (r_2, b_2) sharing a link with a bandwidth of R whose access is controlled by a static priority scheduler. Assume that flow (r_i, b_i) has a deadline of d_i $(d_1 > d_2 > 0)$, and (r_2, b_2) has non-preemptive priority over (r_1, b_1) at the shared link. Denote the maximum packet length of (r_i, b_i) as $l_i < b_i$. Note that by setting $l_i = 0$, the packet-based model defaults to the fluid model. To guarantee that none of the packets in (r_1, b_1) or (r_2, b_2) misses the deadline, R needs to satisfy [120]:

$$r_1 + r_2 \leq R, \tag{B.11a}$$

$$\frac{b_2 + l_1}{R} \leq d_2, \tag{B.11b}$$

$$\frac{b_2 + b_1}{R - r_2} \leq d_1. \tag{B.11c}$$

Therefore, the minimum bandwidth for the link satisfies:

$$\tilde{R}^{(2)*} = \max\left\{r_1 + r_2, \frac{l_1 + b_2}{d_2}, \frac{b_1 + b_2}{d_1} + r_2\right\}. \tag{B.12}$$

Now consider adding a lossless packet-based greedy leaky-bucket (re)shaper for each flow before the shared link, as shown in the above figure. Denote (r_i, b_i)'s shaper as (r_i', b_i'), where

$b'_i \geq l_i$. To guarantee a finite delay inside the shaper, we also need $r'_i \geq r_i$. Under this assumption, if $b'_i \geq b_i$, the shaper has no effect. Hence, we further require that $b'_i < b_i$.

Next, we proceed to characterize the optimal minimum required bandwidth under static priority and (re)shaping. Denote it as $\tilde{R}_s^{(2)*}$.

Under a non-preemptive static priority discipline, the worst-case delay of the high-priority flow is unaffected by the low-priority flow's arrival curve (r_1, b_1), and only depends on its maximum packet size l_1. This is because high-priority packets arriving to an empty high-priority queue wait for at most the transmission time of one low-priority packet. This property holds whether shapers are present or not. Specifically,

Proposition 42. *For a high priority flow (r_2, b_2) traversing a lossless packet-based greedy leaky-bucket shaper (r'_2, b'_2), where $r_2 \leq r'_2$ and $b_2 > b'_2$, before going through a shared link with bandwidth $R > r_2$, the worst-case delay is*

$$D_2^* = \max\left\{\frac{b_2 + l_1}{R}, \frac{b_2 - b'_2}{r'_2} + \frac{l_1 + l_2}{R}\right\},$$

where l_1 is the maximum packet length of the low-priority flow with which it shares the link, and l_2 is its own maximum packet length.

Proof. Denote the virtual delay at t inside the shaper as $D_1(t)$, and that at the shared link as $D_2(t)$. Then for a packet arriving the system at t, its virtual delay inside the system is $D_1(t) + D_2(t + D_1(t))$. For $D_1(t)$, we have

$$D_1(t) = \inf_{0 \leq \tau} \{b_2 + r_2 t \leq b'_2 + t'_2(t + \tau)\} = \left[\frac{b_2 - b'_2 + r_2 t}{r'_2} - t\right]^+.$$

For $D_2(t + D_1(t))$, we have

$$D_2(t+D_1(t)) = \frac{l_1 + l_2}{R} + \inf_{0 \le \tau} \{b_2 + r_2 t - l_2 \le R(t + \tau + D_1(t))\} = \frac{l_1 + l_2}{R} + \left[\frac{b_2 + r_2 t - l_2}{R} - t - D_1(t)\right]^+.$$

Then we have

$$\begin{aligned} D_2^* &= \sup_{0 \le t} \{D_1(t) + D_2(t + D_1(t))\} \\ &\le \sup_{0 \le t} \left\{ \max \left\{ \frac{l_1 + l_2}{R} + \frac{b_2 - b_2' + r_2 t}{r_2'} - t, \frac{l_1 + l_2}{R} + \frac{b_2 + r_2 t - l_2}{R} - t \right\} \right\} \qquad \text{(B.13)} \\ &\le \max \left\{ \frac{l_1 + l_2}{R} + \frac{b_2 - b_2'}{r_2'}, \frac{l_1 + b_2}{R} \right\} \end{aligned}$$

□

Note that after adding the shaper, we have $D_2^* \ge \frac{b_2 + l_1}{R}$. Comparing this expression to Eq. (B.11b), we know that adding (re)shapers will never decrease the high-priority flow's worst-case delay. Furthermore, since ensuring stability of the shared queue mandates $R \ge r_1 + r_2$ irrespective of whether shapers are used, Eq. (B.12) then gives

Corollary 43. *Adding shapers decreases the minimum required bandwidth only when*

$$\tilde{R}^{(2)*} = \frac{b_1 + b_2}{d_1} + r_2 > \max \left\{ r_1 + r_2, \frac{b_2 + l_1}{d_2} \right\}. \qquad \text{(B.14)}$$

Conversely, we note that for the low-priority flow (r_1, b_1), its service curve is determined by both the high-priority flow's arrival curve at the shared link and the shared link's bandwidth, and does not depend on the presence or absence of its own (re)shaper. As a result, adding a (re)shaper to the low-priority flow cannot decrease its worst-case delay (though it can increase it). Consequently, reshaping the low-priority flow cannot contribute to reducing

the bandwidth of the shared link while meeting the delay bounds of both the high and low-priority flows. This is formally stated in the next proposition that simplifies the investigation of the worst-case delay experienced by the low-priority flow by allowing us to omit the use of a shaper for it.

Proposition 44. *Given the service curve assigned to a low-priority flow, adding a packet-based greedy shaper cannot decrease its worst-case delay, and consequently cannot reduce the minimum link bandwidth required to meet the worst-case delay guarantees of both the high and low-priority flows.*

Proof. Denote the service curve of the low-priority flow as $\beta(t)$ and its arrival curve as $\alpha(t)$. Without shaper, the system's virtual delay at t is

$$D'(t) = \inf_{\tau \geq 0}\{\tau : \alpha(t) \leq \beta(t+\tau)\}.$$

Denote the shaper's maximum service curve as $\sigma(t)$, which is also the arrival curve for the shared link. Due to packetization, the system provides the flow a service curve no greater than

$$\sigma \otimes \beta(t) = \inf_{0 \leq s \leq t}\{\sigma(s) + \beta(t-s)\} \leq \sigma(0) + \beta(t) = \beta(t),$$

Hence, the virtual delay given the server is

$$D(t) \geq \inf_{\tau \geq 0}\left\{\tau : \alpha(t) \leq \inf_{0 \leq s \leq t}\{\sigma(s) + \beta(t+\tau-s)\}\right\} \geq \inf_{\tau \geq 0}\{\tau : \alpha(t) \leq \beta(t+\tau)\} = D'(t).$$

As $D(t) \geq D'(t)$, $\forall t \geq 0$, we have $\sup_{t \geq 0}\{D(t)\} \geq \sup_{t \geq 0}\{D'(t)\}$, *i.e.,* adding a shaper cannot decrease the system's worst-case delay.

□

Next, we characterize the worst-case delay D_1^* of the low-priority flow.

Proposition 45. *Given a high priority flow (r_2, b_2) with a packet-based greedy leaky-bucket shaper (r_2', b_2') going through a shared link with bandwidth R, the low-priority flow (r_1, b_1)'s worst-case delay d_1^* is*

1. *when $r_2 = r_2'$, $D_1^* = \frac{b_1 + b_2'}{R - r_2}$;*
2. *when $r_2 < r_2'$ and $\frac{(R - r_2')(b_2 - b_2')}{r_2' - r_2} - (b_1 + b_2') < 0$, $D_1^* = \frac{b_1 + b_2}{R - r_2}$;*
3. *otherwise,* i.e., *$r_2 < r_2'$ and $\frac{(R - r_2')(b_2 - b_2')}{r_2' - r_2} - (b_1 + b_2') \geq 0$ (recall that $r_2 \leq r_2'$)*
   $D_1^* = \max\left\{\frac{b_1 + b_2'}{R - r_2'}, \frac{b_1 + b_2}{r_1} - \frac{(R - r_1 - r_2)(b_2 - b_2')}{r_1(r_2' - r_2)}\right\}$.

Proof. The high-priority flow's arrival curve at the shared link is

$$\alpha_2(t) = \min\left\{\gamma_{r,b}(t), \gamma_{r',b'}(t)\right\}, \tag{B.15}$$

and the low-priority flow's service curve at the shared link is

$$\beta_1(t) = [Rt - \alpha_2(t)]^+ = \left[Rt - \min\left\{\gamma_{r_2,b_2}(t), \gamma_{r_2',b_2'}(t)\right\}\right]^+. \tag{B.16}$$

Hence, the virtual delay at $t > 0$ is

$$D(t) = \inf_{\tau \geq 0}\left\{\tau : r_1 t + b_1 \leq [R(t+\tau) - \min\{r_2(t+\tau) + b_2, r_2'(t+\tau) + b_2'\}]^+\right\} \tag{B.17}$$

When $r_2 = r_2'$, we have

$$\begin{aligned} D(t) &= \inf_{\tau \geq 0}\left\{\tau : r_1 t + b_1 \leq [R(t+\tau) - r_2(t+\tau) - b_2']^+\right\} \\ &= \left[\frac{r_1 t + b_1 + b_2'}{R - r_2} - t\right]^+ \leq \frac{b_1 + b_2'}{R - r_2}. \end{aligned} \tag{B.18}$$

When $r_2 < r_2'$, we can rewrite Eq. (B.16) as

$$\beta_1(t) = [Rt - \min\{\gamma_{r,b}(t), \gamma_{r',b'}(t)\}]^+ = \begin{cases} 0 & \text{when } t = 0 \\ [(R-r_2')t - b_2']^+ & \text{when } t < \frac{b_2-b_2'}{r_2'-r_2} \\ [(R-r_2)t - b_2]^+ & \text{otherwise.} \end{cases} \tag{B.19}$$

- When $\beta_1(\frac{b_2-b_2'}{r_2'-r_2}) \leq b_1 + r_1 t$, *i.e.*, $t \geq \frac{(R-r_2')(b_2-b_2')}{r_1(r_2'-r_2)} - \frac{b_1+b_2'}{r_1}$ we have

$$\begin{aligned} D(t) &= \inf_{\tau \geq 0} \left\{\tau : r_1 t + b_1 \leq [(R-r_2)(t+\tau) - b_2]^+\right\} \\ &= \left[\frac{b_1+b_2}{R-r_2} - \frac{t(R-r_1-r_2)}{R-r_2}\right]^+ \\ &\leq \left[\frac{b_1+b_2}{R-r_2} - \frac{R-r_1-r_2}{R-r_2}\left[\frac{(R-r_2')(b_2-b_2')}{r_1(r_2'-r_2)} - \frac{b_1+b_2'}{r_1}\right]^+\right]^+ \\ &= \begin{cases} \frac{b_1+b_2}{R-r_2} & \text{when } \frac{(R-r_2')(b_2-b_2')}{r_1(r_2'-r_2)} - \frac{b_1+b_2'}{r_1} < 0 \\ \left[\frac{b_1+b_2}{r_1} - \frac{(b_2-b_2')(R-r_1-r_2)}{r_1(r_2'-r_2)}\right]^+ & \text{otherwise.} \end{cases} \end{aligned} \tag{B.20}$$

  Note that when $\frac{(R-r_2')(b_2-b_2')}{r_1(r_2'-r_2)} - \frac{b_1+b_2'}{r_1} < 0$, we have $t \geq \frac{(R-r_2')(b_2-b_2')}{r_1(r_2'-r_2)} - \frac{b_1+b_2'}{r_1}, \forall t$. Therefore, $d_1^* = \frac{b_1+b_2}{R-r_2}$. Next we consider the case when $\frac{(R-r_2')(b_2-b_2')}{r_1(r_2'-r_2)} - \frac{b_1+b_2'}{r_1} \geq 0$.

- When $\beta_1(\frac{b_2-b_2'}{r_2'-r_2}) > b_1 + r_1 t$, *i.e.*, $t < \frac{(R-r_2')(b_2-b_2')}{r_1(r_2'-r_2)} - \frac{b_1+b_2'}{r_1}$. As when $\frac{(R-r_2')(b_2-b_2')}{r_1(r_2'-r_2)} - \frac{b_1+b_2'}{r_1} > 0$ implies $R - r_2' > 0$, we have

$$\begin{aligned} D(t) &= \inf_{\tau \geq 0} \left\{\tau : r_1 t + b_1 \leq [(R-r_2')(t+\tau) - b_2']^+\right\} \\ &= \left[\frac{b_1+b_2'}{R-r_2'} + \frac{t(r_1+r_2'-R)}{R-r_2'}\right]^+, \end{aligned} \tag{B.21}$$

  As $\left[\frac{b_1+b_2'}{R-r_2'} + \frac{t(r_1+r_2'-R)}{R-r_2'}\right]^+$ is a linear function with t, we know $D(t)$ achieves its maximum at either $t = 0$ or $t = \frac{(R-r_2')(b_2-b_2')}{r_1(r_2'-r_2)} - \frac{b_1+b_2'}{r_1}$, which gives $d_1^* = \max\left\{\frac{b_1+b_2'}{R-r_2'}, \frac{b_1+b_2}{r_1} - \frac{(R-r_1-r_2)(b_2-b_2')}{r_1(r_2'-r_2)}\right\}$.

Combine it with Eq. (B.20), when $r_2' > r_2$ we have:

$$D_1^* = \begin{cases} \frac{b_1+b_2}{R-r_2}, & \text{when } \frac{(R-r_2')(b_2-b_2')}{r_1(r_2'-r_2)} - \frac{b_1+b_2'}{r_1} < 0 \\ \max\left\{\frac{b_1+b_2'}{R-r_2'}, \frac{b_1+b_2}{r_1} - \frac{(R-r_1-r_2)(b_2-b_2')}{r_1(r_2'-r_2)}\right\}, & \text{otherwise} \end{cases} \tag{B.22}$$

□

Note that when $r_2 < r_2'$ and $\frac{(R-r_2')(b_2-b_2')}{r_1(r_2'-r_2)} - \frac{b_1+b_2'}{r_1} < 0$ (case 2 of Proposition 45), $\tilde{R}_s^{(2)*}$ ensures $d_1 \geq d_1^* = \frac{b_1+b_2}{R^*-r_2}$, *i.e.,* $R^* \geq \frac{b_1+b_2}{d_1} + r_2$. Combined with Corollary 43, we then know that in this case $\tilde{R}_s^{(2)*}$ is no smaller than $\tilde{R}^{(2)*}$, *i.e.,* the optimal system needs no (re)shaping. Therefore, we only need to focus on cases 1 and 3 when seeking to characterize $\tilde{R}_s^{(2)*}$ in the presence of shapers.

Next, we establish that these two cases can be combined. Specifically, note that $\frac{(R-r_2')(b_2-b_2')}{r_1(r_2'-r_2)} - \frac{b_1+b_2'}{r_1} \geq 0$ implies $R-r_2' > 0$. This means that as $r_2' \to r_2^+$, $\frac{(R-r_2')(b_2-b_2')}{r_1(r_2'-r_2)} - \frac{b_1+b_2'}{r_1} \to +\infty > 0$, so that we are always in case 3 as $r_2' \to r_2^+$. Furthermore, $\lim_{r_2' \to r_2^+} \max\left\{\frac{b_1+b_2'}{R-r_2'}, \frac{b_1+b_2}{r_1} - \frac{(R-r_1-r_2)(b_2-b_2')}{r_1(r_2'-r_2)}\right\} = \max\left\{\frac{b_1+b_2'}{R-r_2'}, -\infty\right\} = \frac{b_1+b_2'}{R-r_2}$, or in other words the value of d_1^* of case 3 is the same as that of case 1 as $r_2' \to r_2^+$. This therefore allows us to write that when $\frac{(R-r_2')(b_2-b_2')}{r_2'-r_2} - (b_1+b_2') \geq 0$, $d_1^* = \max\left\{\frac{b_1+b_2'}{R-r_2'}, \frac{b_1+b_2}{r_1} - \frac{(R-r_1-r_2)(b_2-b_2')}{r_1(r_2'-r_2)}\right\}$.

Together with Proposition 42, this yields the following optimization for $\tilde{R}_s^{(2)*}$:

$$
\begin{aligned}
\textbf{OPT\_2} \quad & \min_{r_2', b_2'} R \\
\text{subject to} \quad & \max\left\{\frac{b_1+b_2'}{R-r_2'}, \frac{b_1+b_2}{r_1} - \frac{(R-r_1-r_2)(b_2-b_2')}{r_1(r_2'-r_2)}\right\} \leq d_1, \\
& \max\left\{\frac{b_2+l_1}{R}, \frac{b_2-b_2'}{r_2'} + \frac{l_1+l_2}{R}\right\} \leq d_2, \\
& \frac{(R-r_2')(b_2-b_2')}{r_1(r_2'-r_2)} - \frac{b_1+b_2'}{r_1} \geq 0, \\
& r_1+r_2 \leq R, \qquad r_2 \leq r_2', \qquad l_2 \leq b_2' \leq b_2
\end{aligned}
\tag{B.23}
$$

Solving **OPT_2** gives the following combination of five cases, four of which yield values $\tilde{R}_s^{(2)*} < \tilde{R}^{(2)*}$, *i.e.,* the introduction of shapers helps reduce the link bandwidth required to meet the flows delay targets, where $r_2'^*$ and $b_2'^*$ defines the optimal shaper:

(i) $\tilde{R}_s^{(2)*} = r_1+r_2 < \tilde{R}^{(2)*}$, $r_2'^* = r_2$, and $b_2'^*$ can be any values inside $[b_2+r_2\left(\frac{l_1+l_2}{r_1+r_2} - d_2\right), d_1r_1 - b_1] \cap [l_2, b_2)$, when $d_1 \in [\frac{l_2+b_1}{r_1}, \frac{b_1+b_2}{r_1})$ and $d_2 \geq \max\left\{\frac{b_2+l_1}{r_1+r_2}, \frac{b_1+b_2-d_1r_1}{r_2} + \frac{l_1+l_2}{r_1+r_2}\right\}$;

(ii) $\tilde{R}_s^{(2)*} = \frac{b_2+l_1}{d_2} < \tilde{R}^{(2)*}$, $r_2'^*$ can be any values inside $[r_2, \min\left\{\tilde{R}_s^{(2)*} - r_1, \tilde{R}_s^{(2)*} - \frac{b_1+l_2}{d_1-(b_2-l_2)/\tilde{R}_s^{(2)*}}\right\}]$, and $b_2'^*$ can be any values inside $[b_2 - \frac{r_2'^*(b_2-l_2)}{\tilde{R}_s^{(2)*}}, d_1(\tilde{R}_s^{(2)*} - r_2'^*) - b_1] \cap [l_2, b_2)$, when $d_2 < \frac{b_2+l_1}{r_1+r_2}$ and $d_1 \in [\frac{d_2(b_1+l_2)}{b_2+l_1-d_2r_2} + \frac{d_2(b_2-l_2)}{b_2+l_1}, \frac{d_2(b_1+b_2)}{b_2+l_1-d_2r_2})$.

(iii) $\tilde{R}_s^{(2)*} = \frac{l_2+b_1}{d_1} + r_2 < \tilde{R}^{(2)*}$, $r_2'^* = r_2$, and $b_2' = l_2$, when $d_1 < \min\left\{\frac{b_1+l_2}{r_1}, \frac{(b_1+l_2)\left(d_2 - \frac{b_2-l_2}{r_2}\right)}{l_1+b_2-r_2d_2}\right\}$ and $d_2 < \frac{l_1+b_2}{r_2}$; and when $d_2 \geq \frac{l_1+b_2}{r_2}$ and $d_1 < \frac{b_1+l_2}{r_1}$.

(iv) $\tilde{R}_s^{(2)*} = \frac{(d_1-d_2)r_2+(b_1+b_2)+\sqrt{((d_1-d_2)r_2+b_1+b_2)^2+4d_1r_2(l_1+l_2)}}{2d_1} < R_0^*$, $r_2'^* = r_2$, and $b_2' = \frac{b_2-b_1-(d_1+d_2)r_2+\sqrt{((d_1-d_2)r_2+b_1+b_2)^2+4d_1r_2(l_1+l_2)}}{2}$

- when $d_2 < \frac{b_2+l_1}{r_1+r_2}$, and $d_1 \in [\frac{(b_1+l_2)\left(d_2-\frac{b_2-l_2}{r_2}\right)}{b_2+l_1-r_2d_2}, \frac{d_2(b_1+l_2)}{b_2+l_1-d_2r_2} + \frac{d_2(b_2-l_2)}{b_2+l_1})$; and
- when $\frac{b_2+l_1}{r_1+r_2} \leq d_2 \leq \frac{b_2+l_1}{r_2}$, and $d_1 \in [\frac{(b_1+l_2)\left(d_2-\frac{b_2-l_2}{r_2}\right)}{b_2+l_1-r_2d_2}, \frac{r_2(l_1+l_2)}{r_1(r_1+r_2)} + \frac{b_1+b_2-d_2r_2}{r_1})$.

(v) otherwise, $\tilde{R}_s^{(2)*} = \tilde{R}^{(2)*}$.

From the solution of **OPT_2**, we directly get

Corollary 46. *We can achieve the optimality of* ***OPT_2*** *through setting* $r_2' = r_2$.

Solving OPT_2

We can divide the optimization into two sub-optimizations:

Sub-optimization 1**:**

$$\begin{aligned}
&\text{minimize}_{r_2', b_2'} \quad R \\
&\text{subject to} \quad r_1 + r_2' - R \le 0, \qquad \frac{b_1 + b_2'}{R - r_2'} - d_1 \le 0, \\
&\qquad\qquad \frac{b_2 + l_1}{R} - d_2 \le 0, \qquad \frac{b_2 - b_2' + l_2}{r_2'} + \frac{l_1}{R} - d_2 \le 0, \\
&\qquad\qquad \frac{b_1 + b_2'}{r_1} - \frac{(R - r_2')(b_2 - b_2')}{r_1(r_2' - r_2)} \le 0, \\
&\qquad\qquad r_1 + r_2 - R \le 0, \qquad r_2 - r_2' \le 0, \\
&\qquad\qquad l_2 - b_2' \le 0, \qquad b_2' - b_2 \le 0
\end{aligned} \tag{B.24}$$

Sub-optimization 2**:**

$$\begin{aligned}
&\text{minimize}_{r_2', b_2'} \quad R \\
&\text{subject to} \quad R - r_1 - r_2' < 0, \qquad \frac{b_1 + b_2}{r_1} - \frac{(R - r_1 - r_2)(b_2 - b_2')}{r_1(r_2' - r_2)} - d_1 \le 0, \\
&\qquad\qquad \frac{b_2 + l_1}{R} - d_2 \le 0, \qquad \frac{b_2 - b_2' + l_2}{r_2'} + \frac{l_1}{R} - d_2 \le 0, \\
&\qquad\qquad \frac{b_1 + b_2'}{r_1} - \frac{(R - r_2')(b_2 - b_2')}{r_1(r_2' - r_2)} \le 0, \\
&\qquad\qquad r_1 + r_2 - R \le 0, \qquad r_2 - r_2' \le 0, \\
&\qquad\qquad l_2 - b_2' \le 0, \qquad b_2' - b_2 \le 0
\end{aligned} \tag{B.25}$$

Denote the solution of sub-optimizations 1 and 2 as R_1^* and R_2^*, respectively. Then we have $R^* = \min\{R_1^*, R_2^*\}$. Next, we solve sub-optimizations 1 and 2. Note than when $R = r_1 + r_2'$,

the two sub-optimizations are the same. Therefore, when solving sub-optimization 2, we only consider the case where $R - r_1 - r_2' < 0$. Then we have:

Lemma 47. *The solution for Sub-optimiaztion* 1 *is*

- $R_1^* = r_1 + r_2$, *when* $d_1 \in [\frac{l_2+b_1}{r_1}, \frac{b_1+b_2}{r_1})$ *and* $d_2 \geq \max\left\{\frac{b_2+l_1}{r_1+r_2}, \frac{b_1+b_2-d_1r_1}{r_2} + \frac{l_1+l_2}{r_1+r_2}\right\}$;
- $R_1^* = \frac{b_2+l_1}{d_2}$, *when* $d_2 < \frac{b_2+l_1}{r_1+r_2}$ *and* $d_1 \in [\frac{d_2(b_1+l_2)}{b_2+l_1-d_2r_2} + \frac{d_2(b_2-l_2)}{b_2+l_1}, \frac{d_2(b_1+b_2)}{b_2+l_1-d_2r_2})$;
- $R_1^* = \frac{l_2+b_1}{d_1} + r_2$, *when* $d_2 < \frac{l_1+b_2}{r_2}$ *and* $d_1 < \min\left\{\frac{b_1+l_2}{r_1}, \frac{(b_1+l_2)\left(d_2-\frac{b_2-l_2}{r_2}\right)}{l_1+b_2-r_2d_2}\right\}$; *and when* $d_2 \geq \frac{l_1+b_2}{r_2}$ *and* $d_1 < \frac{b_1+l_2}{r_1}$;
- $R_1^* = \frac{(d_1-d_2)r_2+(b_1+b_2)+\sqrt{((d_1-d_2)r_2+b_1+b_2)^2+4d_1r_2(l_1+l_2)}}{2d_1}$,
  - *when* $d_2 < \frac{b_2+l_1}{r_1+r_2}$, *and* $d_1 \in [\frac{(b_1+l_2)\left(d_2-\frac{b_2-l_2}{r_2}\right)}{b_2+l_1-r_2d_2}, \frac{d_2(b_1+l_2)}{b_2+l_1-d_2r_2} + \frac{d_2(b_2-l_2)}{b_2+l_1})$; *and*
  - *when* $\frac{b_2+l_1}{r_1+r_2} \leq d_2 \leq \frac{b_2+l_1}{r_2}$, *and* $d_1 \in [\frac{(b_1+l_2)\left(d_2-\frac{b_2-l_2}{r_2}\right)}{b_2+l_1-r_2d_2}, \frac{r_2(l_1+l_2)}{r_1(r_1+r_2)} + \frac{b_1+b_2-d_2r_2}{r_1})$.

Lemma 48. *The solution for Sub-optimization* 2 *is*

- *when* $d_2 < \frac{b_2+l_1}{r_1+r_2}$, *and* $d_1 \in [\frac{d_2(b_2-l_2)}{b_2+l_1} + \frac{b_1+l_2}{r_1}, \frac{d_2(b_1+b_2)}{b_2+l_1-d_2r_2})$, $R_2^* = \frac{b_2+l_1}{d_2} < R_0^*$;
- *otherwise,* $R_2^* = R_0^*$.

Basic algebraic manipulation gives that when $d_2 < \frac{b_2+l_1}{r_1+r_2}$, $\frac{d_2(b_2-l_2)}{b_2+l_1} + \frac{b_1+l_2}{r_1} \geq \frac{d_2(b_1+l_2)}{b_2+l_1-d_2r_2} + \frac{d_2(b_2-l_2)}{b_2+l_1}$. Therefore, we have $R_2^* \geq R_1^*$.

Solution for Sub-optimization 1

The Lagrangian function for sub-optimization 1 is

$$\begin{aligned} L_1(R, r_2', b_2', \boldsymbol{\lambda}) = R &+ \lambda_1(r_1 + r_2' - R) + \lambda_2\left(\frac{b_1 + b_2'}{R - r_2'} - d_1\right) + \lambda_3\left(\frac{b_2 + l_1}{R} - d_2\right) \\ &+ \lambda_4\left(\frac{b_2 - b_2'}{r_2'} + \frac{l_1 + l_2}{R} - d_2\right) + \lambda_5\left(\frac{b_1 + b_2'}{r_1} - \frac{(R - r_2')(b_2 - b_2')}{r_1(r_2' - r_2)}\right) \\ &+ \lambda_6(r_1 + r_2 - R) + \lambda_7(r_2 - r_2') + \lambda_8(l_2 - b_2') + \lambda_9(b_2' - b_2) \end{aligned} \tag{B.26}$$

$$\nabla_{R,r_2',b_2'} L_1 = \begin{bmatrix} 1 - \lambda_1 - \frac{\lambda_2(b_1+b_2')}{(R-r_2')^2} - \frac{\lambda_3(b_2+l_1)}{R^2} - \frac{\lambda_4(l_1+l_2)}{R^2} - \frac{\lambda_5(b_2-b_2')}{r_1(r_2'-r_2)} - \lambda_6 \\ \lambda_1 + \frac{\lambda_2(b_1+b_2')}{(R-r_2')^2} - \frac{\lambda_4(b_2-b_2')}{r_2'^2} + \frac{\lambda_5(b_2-b_2')(R-r_2)}{r_1(r_2'-r_2)^2} - \lambda_7 \\ \frac{\lambda_2}{R-r_2'} - \frac{\lambda_4}{r_2'} + \lambda_5\left(\frac{1}{r_1} + \frac{R-r_2'}{r_1(r_2'-r_2)}\right) - \lambda_8 + \lambda_9 \end{bmatrix} \tag{B.27}$$

$$\operatorname{diag}(\triangle_{R,r_2',b_2'} L_1) = \begin{bmatrix} \frac{2\lambda_2(b_1+b_2')}{(R-r_2')^3} + \frac{2\lambda_3(b_2+l_1)}{R^3} + \frac{2\lambda_4(l_1+l_2)}{R^3} \\ \frac{2\lambda_2(b_1+b_2')}{(R-r_2')^3} + \frac{2\lambda_4(b_2-b_2')}{r_2'^3} - \frac{2\lambda_5(b_2-b_2')(R-r_2)}{r_1(r_2'-r_2)^3} \\ 0 \end{bmatrix} \tag{B.28}$$

From Eq. (B.20), we know that when $\frac{b_1+b_2'}{r_1} - \frac{(R-r_2')(b_2-b_2')}{r_1(r_2'-r_2)} = 0$, $d_1^* = \frac{b_1+b_2}{R-r_2}$. Combine it with Corollary 43, we have $R_1^* = R_0^*$.

Next we consider the case where $\frac{b_1+b_2'}{r_1} - \frac{(R-r_2')(b_2-b_2')}{r_1(r_2'-r_2)} > 0$. Then from KKT conditions' complementary slackness requirement, we have $\lambda_5 = 0$. Substitute $\lambda_5 = 0$ into Eq. (B.28),

we have

$$\operatorname{diag}(\triangle_{R,r_2',b_2'}L_1) = \begin{bmatrix} \frac{2\lambda_2(b_1+b_2')}{(R-r_2')^3} + \frac{2\lambda_3(b_2+l_1)}{R^3} + \frac{2\lambda_4(l_1+l_2)}{R^3} \\ \\ \frac{2\lambda_2(b_1+b_2')}{(R-r_2')^3} + \frac{2\lambda_4(b_2-b_2')}{r_2'^3} \\ \\ 0 \end{bmatrix} \geq 0 \tag{B.29}$$

Therefore, for any $(r_2', b_2', \boldsymbol{\lambda})$ satisfying KKT's necessary conditions, it is a local optimum. The necessary conditions for the optimization under $\lambda_5 = 0$ is:

$$\begin{aligned}
&1 - \lambda_1 - \frac{\lambda_2(b_1+b_2')}{(R-r_2')^2} - \frac{\lambda_3(b_2+l_1)}{R^2} - \frac{\lambda_4(l_1+l_2)}{R^2} - \lambda_6 = 0, && \frac{\lambda_2}{R-r_2'} - \frac{\lambda_4}{r_2'} - \lambda_8 + \lambda_9 = 0, \\
&\lambda_1 + \frac{\lambda_2(b_1+b_2')}{(R-r_2')^2} - \frac{\lambda_4(b_2-b_2')}{r_2'^2} - \lambda_7 = 0, && \lambda_i \geq 0, \text{ for } i = 1, ...9, \\
&r_1 + r_2' - R \leq 0, && \lambda_1(r_1 + r_2' - R) = 0, \\
&\frac{b_1+b_2'}{R-r_2'} - d_1 \leq 0, && \lambda_2\left(\frac{b_1+b_2'}{R-r_2'} - d_1\right) = 0, \\
&\frac{b_2+l_1}{R} - d_2 \leq 0, && \lambda_3\left(\frac{b_2+l_1}{R} - d_2\right) = 0, \\
&\frac{b_2-b_2'}{r_2'} + \frac{l_1+l_2}{R} - d_2 \leq 0, && \lambda_4\left(\frac{b_2-b_2'}{r_2'} + \frac{l_1+l_2}{R} - d_2\right) = 0, \\
&\frac{b_1+b_2'}{r_1} - \frac{(R-r_2')(b_2-b_2')}{r_1(r_2'-r_2)} < 0, && \lambda_5 = 0, \\
&r_1 + r_2 - R \leq 0, && \lambda_6(r_1 + r_2 - R) = 0, \\
&r_2 - r_2' \leq 0, && \lambda_7(r_2 - r_2') = 0, \\
&l_2 - b_2' \leq 0, && \lambda_8(l_2 - b_2') = 0, \\
&b_2' - b_2 \leq 0 && \lambda_9(b_2' - b_2) = 0
\end{aligned} \tag{B.30}$$

For the conditions in Eq. (B.30), we have:

- $R_1^* = r_1 + r_2,$ when $d_1 \in [\frac{l_2+b_1}{r_1}, \frac{b_1+b_2}{r_1})$ and $d_2 \geq \max\left\{\frac{b_2+l_1}{r_1+r_2}, \frac{b_1+b_2-d_1r_1}{r_2} + \frac{l_1+l_2}{r_1+r_2}\right\}$;
- $R_1^* = \frac{b_2+l_1}{d_2} \geq r_1 + r_2,$ when $d_2 < \frac{b_2+l_1}{r_1+r_2}$ and $d_1 \in [\frac{d_2(b_1+l_2)}{b_2+l_1-d_2r_2} + \frac{d_2(b_2-l_2)}{b_2+l_1}, \frac{d_2(b_1+b_2)}{b_2+l_1-d_2r_2})$;
- $R_1^* = \frac{l_2+b_1}{d_1} + r_2$, when $d_2 < \frac{l_1+b_2}{r_2}$ and $d_1 < \min\left\{\frac{b_1+l_2}{r_1}, \frac{(b_1+l_2)\left(d_2-\frac{b_2-l_2}{r_2}\right)}{l_1+b_2-r_2d_2}\right\}$; and when $d_2 \geq \frac{l_1+b_2}{r_2}$ and $d_1 < \frac{b_1+l_2}{r_1}$;
- $R_1^* = \frac{(d_1-d_2)r_2+(b_1+b_2)+\sqrt{((d_1-d_2)r_2+b_1+b_2)^2+4d_1r_2(l_1+l_2)}}{2d_1}$,
  - when $d_2 < \frac{b_2+l_1}{r_1+r_2}$, and $d_1 \in [\frac{(b_1+l_2)\left(d_2-\frac{b_2-l_2}{r_2}\right)}{b_2+l_1-r_2d_2}, \frac{d_2(b_1+l_2)}{b_2+l_1-d_2r_2} + \frac{d_2(b_2-l_2)}{b_2+l_1})$; and
  - when $\frac{b_2+l_1}{r_1+r_2} \leq d_2 \leq \frac{b_2+l_1}{r_2}$, and $d_1 \in [\frac{(b_1+l_2)\left(d_2-\frac{b_2-l_2}{r_2}\right)}{b_2+l_1-r_2d_2}, \frac{r_2(l_1+l_2)}{r_1(r_1+r_2)} + \frac{b_1+b_2-d_2r_2}{r_1})$.

Proof. Note that when $b_2' = b_2$, the shaper has no effect, *i.e.,* $R^* = R_0^*$. Therefore, we consider only $\lambda_9 = 0$ and $b_2' < b_2$. Then from $\frac{\lambda_2}{R-r_2'} - \frac{\lambda_4}{r_2'} - \lambda_8 + \lambda_9 = 0$ we have 1) if $\lambda_2 = 0$, then $\lambda_4 = \lambda_8 = 0$; and 2) if $\lambda_2 > 0$, then $\lambda_4 + \lambda_8 > 0$.

- When $\lambda_2 = 0$, $\lambda_4 = \lambda_8 = 0$, from $\lambda_1 + \frac{\lambda_2(b_1+b_2')}{(R-r_2')^2} - \frac{\lambda_4(b_2-b_2')}{r_2'^2} - \lambda_7 = 0$, we have $\lambda_1 = \lambda_7$.

  • When $\lambda_1 = \lambda_7 > 0$, it has $R = r_1 + r_2'$ and $r_2 = r_2'$. Therefore, $R = r_1 + r_2$. Then the constraints become:

$$\begin{cases} \frac{b_1+b_2'}{r_1} - d_1 \leq 0 \implies b_2' \leq d_1r_1 - b_1, \\ \frac{b_2+l_1}{r_1+r_2} - d_2 \leq 0 \implies d_2 \geq \frac{b_2+l_1}{r_1+r_2}, \\ \frac{b_2-b_2'}{r_2'} + \frac{l_1+l_2}{r_1+r_2} - d_2 \leq 0, \implies b_2' \geq b_2 + r_2\left(\frac{l_1+l_2}{r_1+r_2} - d_2\right) \\ l_2 \leq b_2' < b_2 \end{cases} \tag{B.31}$$

Hence we have $b_2' \in [b_2 + r_2\left(\frac{l_1+l_2}{r_1+r_2} - d_2\right), d_1 r_1 - b_1] \cap [l_2, b_2)$. As $d_2 \geq \frac{b_2+l_1}{r_1+r_2} > \frac{l_1+l_2}{r_1+r_2}$, we have $b_2 - r_2\left(\frac{l_1+l_2}{r_1+r_2} - d_2\right) < b_2$. Therefore, to guarantee $[b_2 + r_2\left(\frac{l_1+l_2}{r_1+r_2} - d_2\right), d_1 r_1 - b_1] \cap [l_2, b_2) \neq \emptyset$:

$$\begin{cases} d_1 r_1 - b_1 \geq l_2 \implies d_1 \geq \frac{l_2+b_1}{r_1} \\ \\ b_2 + r_2\left(\frac{l_1+l_2}{r_1+r_2} - d_2\right) \leq d_1 r_1 - b_1 \implies d_2 \geq \frac{b_1+b_2-d_1 r_1}{r_2} + \frac{l_1+l_2}{r_1+r_2} \end{cases} \tag{B.32}$$

Remember that adding shaper is beneficial only when $R_0^* = \frac{b_1+b_2}{d_1} + r_2 > \max\left\{r_1 + r_2, \frac{b_2+l_1}{d_2}\right\}$, *i.e.,* $d_1 < \frac{b_1+b_2}{r_1}$ and $d_2 > \frac{b_2+l_1}{\frac{b_1+b_2}{d_1}+r_2}$, which gives $\frac{b_2+l_1}{r_1+r_2} > \frac{b_2+l_1}{\frac{b_1+b_2}{d_1}}$. Therefore, we have

$$\circ\ d_1 \in [\frac{l_2+b_1}{r_1}, \frac{b_1+b_2}{r_1}) \text{ and } d_2 \geq \max\left\{\frac{b_2+l_1}{r_1+r_2},\ \frac{b_1+b_2-d_1 r_1}{r_2} + \frac{l_1+l_2}{r_1+r_2}\right\}. \tag{B.33}$$

• When $\lambda_1 = \lambda_7 = 0$, from $1 - \lambda_1 - \frac{\lambda_2(b_1+b_2')}{(R-r_2')^2} - \frac{\lambda_3(b_2+l_1)}{R^2} - \frac{\lambda_4(l_1+l_2)}{R^2} - \lambda_6 = 0$ we have $\lambda_3 + \lambda_6 > 0$. If $\lambda_6 > 0$, it has $r_1 + r_2 - R = 0$. As $r_1 + r_2' - R \leq 0$, it has $r_2' = r_2$. Therefore, it produces the same optimization as Eq. (B.31). Therefore, we only consider $\lambda_6 = 0$ in this case. When $\lambda_6 = 0$, $\lambda_3 > 0$, then we have $R = \frac{b_2+l_1}{d_2}$. Note that $R = \frac{b_2+l_1}{d_2}$ implies $\frac{b_2+l_1}{R} \geq \frac{b_2-b_2'}{r_2'} + \frac{l_1+l_2}{R}$, *i.e.,* $b_2' \geq b_2 - \frac{r_2'(b_2-l_2)}{R}$. Then the constraints become

$$\begin{cases} r_2' \in [r_2, R - r_1], \qquad b_2' \in [l_2, b_2), \qquad r_1 + r_2 \leq R, \\ \\ \frac{b_1+b_2'}{R-r_2'} \leq d_1 \implies b_2' \leq d_1(R - r_2') - b_1, \\ \\ b_2' \geq b_2 - \frac{r_2'(b_2-l_2)}{R}, \\ \\ b_1 + b_2' - \frac{(R-r_2')(b_2-b_2')}{r_2'-r_2} < 0 \implies b_2' < b_2 - \frac{(b_1+b_2)(r_2'-r_2)}{R-r_2}. \end{cases} \tag{B.34}$$

Note that $b_2 - \frac{(b_1+b_2)(r_2'-r_2)}{R-r_2} - d_1(R-r_2') + b_1 = (R-r_2')\left(\frac{b_1+b_2}{R-r_2} - d_1\right) = \frac{d_1(R-r_2')(R_0^*-R)}{R-r_2}$. As $r_2' < R$, under $R < R_0^*$, it has $b_2 - \frac{(b_1+b_2)(r_2'-r_2)}{R-r_2} > d_1(R-r_2') - b_1$. Hence, we have $b_2' \in [b_2 - \frac{r_2'(b_2-l_2)}{R}, d_1(R-r_2') - b_1]$. Next we configure the conditions where $\exists r_2' \in [r_2, R-r_1]$, such that $[b_2 - \frac{r_2'(b_2-l_2)}{R}, d_1(R-r_2') - b_1] \cap [l_2, b_2) \neq \emptyset$.

For $[b_2 - \frac{r_2'(b_2-l_2)}{R}, d_1(R-r_2') - b_1] \cap [l_2, b_2) \neq \emptyset$, it requires

$$\begin{cases} b_2 - \frac{r_2'(b_2-l_2)}{R} < b_2 \implies b_2 > l_2, \\ d_1(R-r_2') - b_1 \geq l_2 \implies r_2' \leq R - \frac{b_1+l_2}{d_1}, \\ b_2 - \frac{r_2'(b_2-l_2)}{R} - d_1(R-r_2') + b_1 \leq 0 \implies r_2' \leq R - \frac{b_1+l_2}{d_1-(b_2-l_2)/R} \end{cases} \tag{B.35}$$

Basic algebraic manipulation gives $R - \frac{b_1+l_2}{d_1} > R - \frac{b_1+l_2}{d_1-(b_2-l_2)/R}$. Hence, Eq. (B.35) gives $r_2' \leq R - \frac{b_1+l_2}{d_1-(b_2-l_2)/R}$. Combine it with $r_2' \in [r_2, R-r_1]$ and $R = \frac{b_2+l_1}{d_2}$, we have

$$r_2 \leq R - \frac{b_1+l_2}{d_1-(b_2-l_2)/R} \implies d_1 \geq \frac{d_2(b_1+l_2)}{b_2+l_1-d_2r_2} + \frac{(b_2-l_2)d_2}{b_2+l_1}, \tag{B.36}$$

Also, from $r_1 + r_2 < R < \frac{b_1+b_2}{d_1} + r_2$, we have $d_2 < \frac{b_2+l_1}{r_1+r_2}$ and $d_1 < \frac{d_2(b_1+b_2)}{b_2+l_1-d_2r_2}$. Hence, we have

$$\circ\ d_2 < \frac{b_2+l_1}{r_1+r_2}, \text{ and } d_1 \in [\frac{d_2(b_1+l_2)}{b_2+l_1-d_2r_2} + \frac{(b_2-l_2)d_2}{b_2+l_1}, \frac{d_2(b_1+b_2)}{b_2+l_1-d_2r_2}). \tag{B.37}$$

Basic algebraic manipulation shows that the interval is always valid.

- When $\lambda_2 > 0$, as $\lambda_9 = 0$, from $\frac{\lambda_2}{R-r_2'} - \frac{\lambda_4}{r_2'} - \lambda_8 + \lambda_9 = 0$ we have $\frac{\lambda_2}{R-r_2'} = \frac{\lambda_4}{r_2'} + \lambda_8$ and $\lambda_4 + \lambda_8 > 0$. Combining it with $\lambda_1 + \frac{\lambda_2(b_1+b_2')}{(R-r_2')^2} - \frac{\lambda_4(b_2-b_2')}{r_2'^2} - \lambda_7$ and $\frac{b_1+b_2'}{R-r_2'} = d_1$. We have $\lambda_8 d_1 + \frac{\lambda_4}{r_2'}\left(d_1 - \frac{b_2-b_2'}{r_2'}\right) + \lambda_1 - \lambda_7 = 0$. As $\frac{b_2-b_2'}{r_2'} + \frac{l_1+l_2}{R} \leq d_2$, $\lambda_8 d_1 + \frac{\lambda_4}{r_2'}\left(d_1 - \frac{b_2-b_2'}{r_2'}\right) +$

$\lambda_1 - \lambda_7 \geq d_1\lambda_8 + \lambda_1 - \lambda_7 + \frac{\lambda_4}{r_2'}\left(d_1 - d_2 + \frac{l_1+l_2}{R}\right) = 0$. Therefore, given $\lambda_4 + \lambda_8 > 0$, we have $\lambda_7 > 0$, *i.e.*, $r_2' = r_2$.

• When $\lambda_8 = 0$, $\lambda_4 > 0$, *i.e.*, $\frac{b_2 - b_2'}{r_2} + \frac{l_1+l_2}{R} = d_2$. Then the constraints become

$$\begin{cases} r_1 + r_2 \leq R, \qquad b_2' \in [l_2, b_2), \\ \frac{b_1+b_2'}{R-r_2} = d_1 \implies R = \frac{b_1+b_2'}{d_1} + r_2, \\ \frac{b_2-b_2'}{r_2} + \frac{l_1+l_2}{R} = d_2, \\ R \geq \frac{b_2+l_1}{d_2}. \end{cases} \tag{B.38}$$

Substituting $R = \frac{b_1+b_2'}{d_1} + r_2$ into $\frac{b_2-b_2'}{r_2} + \frac{l_1+l_2}{R} = d_2$ gives

$$\frac{d_1}{r_2}R^2 - \left[d_1 - d_2 + \frac{b_1+b_2}{r_2}\right]R - (l_1+l_2) = 0,$$

which gives

$$R = \frac{(d_1-d_2)r_2 + (b_1+b_2) + \sqrt{((d_1-d_2)r_2 + b_1 + b_2)^2 + 4d_1r_2(l_1+l_2)}}{2d_1},$$

and

$$b_2' = \frac{(b_2-b_1) - (d_1+d_2)r_2 + \sqrt{((d_1-d_2)r_2 + b_1 + b_2)^2 + 4d_1r_2(l_1+l_2)}}{2}.$$

$b_2' \in [l_2, b_2)$ gives

$$d_2 \in \left(\frac{d_1(l_1+l_2)}{b_1+b_2+d_1r_2}, \ \frac{d_1(l_1+l_2)}{b_1+l_2+d_1r_2} + \frac{b_2-l_2}{r_2} \right].$$

Note that when $R_0^* = \frac{b_1+b_2}{d_1}+r_2$, we have $d_2 > \frac{(b_2+l_1)d_1}{b_1+b_2+r_2d_1} \in (\frac{d_1(l_1+l_2)}{b_1+b_2+d_1r_2}, \frac{d_1(l_1+l_2)}{b_1+l_2+d_1r_2}+\frac{b_2-l_2}{r_2})$. Hence we have $d_2 \in (\ \frac{(b_2+l_1)d_1}{b_1+b_2+r_2d_1},\ \frac{d_1(l_1+l_2)}{b_1+l_2+d_1r_2}+\frac{b_2-l_2}{r_2}\ ]$, *i.e.*,

$$d_2 < \frac{b_2+l_1}{r_2}, \text{ and } d_1 \in [\frac{(b_1+l_2)\left(d_2-\frac{b_2-l_2}{r_2}\right)}{b_2+l_1-r_2d_2}, \frac{d_2(b_1+b_2)}{b_2+l_1-r_2d_2}). \tag{B.39}$$

From $R \geq r_1+r_2$ and $R \geq \frac{b_2+l_1}{d_2}$, we have: when $d_2 < \frac{b_2+l_1}{r_1+r_2}$, $d_1 \leq \frac{r_2(l_1+l_2)}{\frac{b_2+l_1}{d_2}\left(\frac{b_2+l_1}{d_2}-r_2\right)} + \frac{b_1+b_2-d_2r_2}{\frac{b_2+l_1}{d_2}-r_2} = \frac{d_2(b_1+l_2)}{b_2+l_1-d_2r_2} + \frac{d_2(b_2-l_2)}{b_2+l_1}$, which is greater than $\frac{d_2(b_1+b_2)}{b_2+l_1-r_2d_2}$; and when $d_2 \geq \frac{b_2+l_1}{r_1+r_2}$, $d_1 \leq \frac{r_2(l_1+l_2)}{r_1(r_1+r_2)} + \frac{b_1+b_2-d_2r_2}{r_1} < \frac{d_2(b_1+b_2)}{b_2+l_1-r_2d_2}$. Combining it with Eq. (B.39) gives:

- ◦ when $d_2 < \frac{l_1+b_2}{r_2+r_1}$, $d_1 \in [\frac{(b_1+l_2)\left(d_2-\frac{b_2-l_2}{r_2}\right)}{b_2+l_1-r_2d_2}, \frac{d_2(b_1+b_2)}{b_2+l_1-r_2d_2})$;
- ◦ when $\frac{l_1+b_2}{r_1+r_2} \leq d_2 \leq \frac{b_2+l_1}{r_2}$, $d_1 \in [\frac{(b_1+l_2)\left(d_2-\frac{b_2-l_2}{r_2}\right)}{b_2+l_1-r_2d_2}, \frac{r_2(l_1+l_2)}{r_1(r_1+r_2)} + \frac{b_1+b_2-d_2r_2}{r_1})$.

• When $\lambda_8 > 0$, *i.e.*, $b_2' = l_2$, we have $\frac{b_2-b_2'}{r_2} + \frac{l_1+l_2}{R} = \frac{l_1+b_2}{R} + \left(\frac{1}{r_2}-\frac{1}{R}\right)(b_2-l_2) > \frac{b_2+l_1}{R}$. Then the constraints become

$$\begin{cases} \frac{b_1+l_2}{R-r_2} = d_1 \implies R = \frac{b_1+l_2}{d_1}+r_2 < \frac{b_2+b_1}{d_1}+r_2 = R_0^*, \\ R > r_1+r_2 \implies d_1 < \frac{b_1+l_2}{r_1} \\ \frac{b_2-l_2}{r_2}+\frac{l_1+l_2}{R} \leq d_2 \implies d_2 \geq \frac{b_2-l_2}{r_2}+\frac{(l_1+l_2)d_1}{b_1+l_2+r_2d_1} > \frac{(b_2+l_1)d_1}{b_1+b_2+r_2d_1s} \end{cases} \tag{B.40}$$

Hence, we have $d_1 < \frac{b_1+l_2}{r_1}$, and $d_2 \geq \frac{b_2-l_2}{r_2} + \frac{(l_1+l_2)d_1}{b_1+l_2+r_2d_1}$. Basic algebraic manipulation gives that it is equivalent to:

- ◦ when $d_2 \geq \frac{l_1+b_2}{r_2}$, $d_1 < \frac{b_1+l_2}{r_1}$;
- ◦ when $d_2 < \frac{l_1+b_2}{r_2}$, $d_1 \leq \min\left\{\frac{b_1+l_2}{r_1}, \frac{(b_1+l_2)\left(d_2-\frac{b_2-l_2}{r_2}\right)}{l_1+b_2-r_2d_2}\right\}$.

In summary, the local optimums are:

- $R = r_1 + r_2,$ when $d_1 \in [\frac{l_2+b_1}{r_1}, \frac{b_1+b_2}{r_1})$ and $d_2 \in [\max\left\{\frac{b_2+l_1}{r_1+r_2}, \frac{b_1+b_2-d_1r_1}{r_2} + \frac{l_1+l_2}{r_1+r_2}\right\}, d_1)$;
- $R = \frac{b_2+l_1}{d_2} \geq r_1 + r_2,$ when $d_2 < \frac{b_2+l_1}{r_1+r_2}$ and $d_1 \in [\frac{d_2(b_1+l_2)}{b_2+l_1-d_2r_2} + \frac{d_2(b_2-l_2)}{b_2+l_1}, \frac{d_2(b_1+b_2)}{b_2+l_1-d_2r_2})$;
- $R = \frac{(d_1-d_2)r_2+(b_1+b_2)+\sqrt{((d_1-d_2)r_2+b_1+b_2)^2+4d_1r_2(l_1+l_2)}}{2d_1} \geq \max\left\{r_1 + r_2, \frac{b_2+l_1}{d_2}, \frac{b_1+l_2}{d_1} + r_2\right\},$
  - when $d_2 < \frac{b_2+l_1}{r_1+r_2}$, and $d_1 \in [\frac{(b_1+l_2)\left(d_2-\frac{b_2-l_2}{r_2}\right)}{b_2+l_1-r_2d_2}, \frac{d_2(b_1+b_2)}{b_2+l_1-r_2d_2})$;
  - when $\frac{b_2+l_1}{r_1+r_2} \leq d_2 \leq \frac{b_2+l_1}{r_2}$, and $d_1 \in [\frac{(b_1+l_2)\left(d_2-\frac{b_2-l_2}{r_2}\right)}{b_2+l_1-r_2d_2}, \frac{r_2(l_1+l_2)}{r_1(r_1+r_2)} + \frac{b_1+b_2-d_2r_2}{r_1})$;
- $R = \frac{b_1+l_2}{d_1} + r_2 > \max\left\{r_1 + r_2, \frac{b_2+l_1}{d_2}\right\},$
  - when $d_2 < \frac{l_1+b_2}{r_2}$ and $d_1 < \min\left\{\frac{b_1+l_2}{r_1}, \frac{(b_1+l_2)\left(d_2-\frac{b_2-l_2}{r_2}\right)}{l_1+b_2-r_2d_2}\right\}$;
  - when $d_2 \geq \frac{l_1+b_2}{r_2}$ and $d_1 < \frac{b_1+l_2}{r_1}$.
- $R = R_0^*$, otherwise.

Next we characterize the global optimum R_1^*. We considering three cases: $d_2 \geq \frac{b_2+l_1}{r_2}$, $\frac{b_2+l_1}{r_1+r_2} < d_2 < \frac{b_2+l_1}{r_2}$, and $d_2 \leq \frac{b_2+l_1}{r_1+r_2}$.

- When $d_2 \geq \frac{b_2+l_1}{r_2}$, we consider whether $\frac{b_1+b_2}{r_1+r_2} + \frac{(l_1+l_2)r_2}{(r_1+r_2)^2} - \frac{l_2+b_1}{r_1} \geq 0$ or not.
  - When $\frac{b_1+b_2}{r_1+r_2} + \frac{r_2(l_1+l_2)}{(r_1+r_2)^2} - \frac{l_2+b_1}{r_1} \geq 0$, *i.e.*, $r_1r_2(r_1+r_2)\left(\frac{b_2-l_2}{r_2} - \frac{b_1-l_1}{r_1}\right) \geq r_2^2(l_1+l_2)$, basic algebraic manipulation gives that for all $d_2 \geq \frac{b_1+b_2}{r_1+r_2} + \frac{r_2(l_1+l_2)}{(r_1+r_2)^2}$, $R = r_1 + r_2$. As $\frac{b_1+b_2}{r_1+r_2} + \frac{r_2(l_1+l_2)}{(r_1+r_2)^2} - \frac{l_1+b_2}{r_2} = \frac{r_1r_2}{r_2(r_2+r_2)}\left(\frac{b_1-l_1}{r_1} - \frac{b_2-l_2}{r_2}\right) + \frac{(r_2^2-r_1^2-r_1r_2)(l_1+l_2)}{r_2(r_1+r_2)^2} \leq -r_1(r_1+r_2)(l_1+l_2) < 0$, we have $\frac{l_2+b_1}{r_2} > \frac{b_1+b_2}{r_1+r_2} + \frac{r_2(l_1+l_2)}{(r_1+r_2)^2}$. Therefore, we have $R_1^* = r_1 + r_2$.
  - When $\frac{b_1+b_2}{r_1+r_2} + \frac{(l_1+l_2)r_2}{(r_1+r_2)^2} - \frac{l_2+b_1}{r_1} < 0$, from the local optimums we have:
    - when $\frac{l_2+b_1}{r_1} \leq \frac{l_1+b_2}{r_2}$, $R_1^* = r_1 + r_2$;
    - when $\frac{l_2+b_1}{r_1} > \frac{l_1+b_2}{r_2}$, $R_1^* = r_1 + r_2$ when $d_1 \geq \frac{l_2+b_1}{r_1}$, while $R_1^* = \frac{b_1+l_2}{d_1} + r_2$ when $d_1 < \frac{l_2+b_1}{r_1}$.

- When $\frac{b_2+l_1}{r_1+r_2} < d_2 < \frac{b_2+l_1}{r_2}$, we separate it into three conditions: $d_1 \geq \frac{b_1+b_2}{r_1+r_2} + \frac{r_2(l_1+l_2)}{(r_1+r_2)^2}$, $d_1 \in (\frac{b_1+b_2-d_2r_2}{r_1} + \frac{r_2(l_1+l_2)}{r_1(r_1+r_2)}, \frac{b_1+b_2}{r_1+r_2} + \frac{r_2(l_1+l_2)}{(r_1+r_2)^2})$, and $d_1 < \frac{b_1+b_2-d_2r_2}{r_1} + \frac{r_2(l_1+l_2)}{r_1(r_1+r_2)}$.

  ◦ When $d_1 \geq \frac{b_1+b_2}{r_1+r_2} + \frac{r_2(l_1+l_2)}{(r_1+r_2)^2}$, we have $\frac{b_2+l_1}{r_1+r_2} > \frac{b_1+b_2-d_1r_1}{r_2} + \frac{l_1+l_2}{r_1+r_2} < \frac{b_2+l_1}{r_1+r_2} < d_2$. Therefore, we have $R_1^* = r_1 + r_2$.

  ◦ When $d_1 \in (\frac{b_1+b_2-d_2r_2}{r_1} + \frac{r_2(l_1+l_2)}{r_1(r_1+r_2)}, \frac{b_1+b_2}{r_1+r_2} + \frac{r_2(l_1+l_2)}{(r_1+r_2)^2})$, basic algebraic manipulation gives $d_1 > \frac{b_1+b_2-d_2r_2}{r_1} + \frac{r_2(l_1+l_2)}{r_1(r_1+r_2)} > \frac{b_2+l_1}{r_1+r_2}$. Therefore, we have $R_1^* = r_1 + r_2$.

  ◦ When $d_1 < \frac{b_1+b_2-d_2r_2}{r_1} + \frac{r_2(l_1+l_2)}{r_1(r_1+r_2)}$, from the local optimums we directly have:

    ◦ when $d_2 \geq \frac{l_1+b_2}{r_1+r_2} + \frac{r_1(b_2-l_2)}{r_2(r_1+r_2)}$, $R_1^* = \frac{b_1+l_2}{d_1} + r_2$.

    ◦ when $d_2 < \frac{l_1+b_2}{r_1+r_2} + \frac{r_1(b_2-l_2)}{r_2(r_1+r_2)}$, $R_1^* = \frac{(d_1-d_2)r_2+(b_1+b_2)+\sqrt{((d_1-d_2)r_2+b_1+b_2)^2+4d_1r_2(l_1+l_2)}}{2d_1}$

  when $d_1 \in [\frac{(b_1+l_2)\left(d_2-\frac{b_2-l_2}{r_2}\right)}{b_2+l_1-r_2d_2}, \frac{b_1+b_2-d_2r_2}{r_1} + \frac{r_2(l_1+l_2)}{r_1(r_1+r_2)})$, while $R_1^* = \frac{b_1+l_2}{d_1} + r_2$ when $d_1 < \frac{(b_1+l_2)\left(d_2-\frac{b_2-l_2}{r_2}\right)}{b_2+l_1-r_2d_2}$.

- When $d_2 \leq \frac{b_2+l_1}{r_1+r_2}$, we have $\frac{b_1+l_2}{r_1} > \frac{(b_2+l_2)\left(d_2-\frac{b_2-l_2}{r_2}\right)}{l_1+b_2-r_2d_2}$. Therefore, from local optimums we directly have:

  ◦ when $d_1 \in (d_2, \frac{(b_2+l_2)\left(d_2-\frac{b_2-l_2}{r_2}\right)}{l_1+b_2-r_2d_2}]$, $R_1^* = \frac{b_1+l_2}{d_1} + r_2$;

  ◦ when $d_1 \in [\frac{(b_2+l_2)\left(d_2-\frac{b_2-l_2}{r_2}\right)}{l_1+b_2-r_2d_2}, \frac{d_2(b_1+l_2)}{b_2+l_1-d_2r_2} + \frac{d_2(b_2-l_2)}{b_2+l_1})$,

  $$R_1^* = \frac{(d_1-d_2)r_2+(b_1+b_2)+\sqrt{((d_1-d_2)r_2+b_1+b_2)^2+4d_1r_2(l_1+l_2)}}{2d_1};$$

  ◦ when $d_1 \in (\frac{d_2(b_1+l_2)}{b_2+l_1-d_2r_2} + \frac{d_2(b_2-l_2)}{b_2+l_1}), \frac{d_2(b_1+b_2)}{b_2+l_1-d_2r_2})$, $R_1^* = \frac{b_2+l_1}{d_2}$.

Remark: The above analysis also shows: $R_1^* > R_0^*$ as long as $R_0^* = \frac{b_1+b_2}{d_1} + r_2 > \max\left\{\frac{b_2+l_1}{d_2}, r_1+r_2\right\}$.

In summary, the global optimum is:

- $R_1^* = r_1 + r_2$, when $d_1 \in [\frac{l_2+b_1}{r_1}, \frac{b_1+b_2}{r_1})$ and $d_2 \geq \max\left\{\frac{b_2+l_1}{r_1+r_2}, \frac{b_1+b_2-d_1r_1}{r_2} + \frac{l_1+l_2}{r_1+r_2}\right\}$;

- $R_1^* = \frac{b_2+l_1}{d_2} \geq r_1 + r_2$, when $d_2 < \frac{b_2+l_1}{r_1+r_2}$ and $d_1 \in [\frac{d_2(b_1+l_2)}{b_2+l_1-d_2r_2} + \frac{d_2(b_2-l_2)}{b_2+l_1}, \frac{d_2(b_1+b_2)}{b_2+l_1-d_2r_2})$;

- $R_1^* = \frac{l_2+b_1}{d_1} + r_2$, when $d_2 < \frac{l_1+b_2}{r_2}$ and $d_1 < \min\left\{\frac{b_1+l_2}{r_1}, \frac{(b_1+l_2)\left(d_2-\frac{b_2-l_2}{r_2}\right)}{l_1+b_2-r_2d_2}\right\}$; and when $d_2 \geq \frac{l_1+b_2}{r_2}$ and $d_1 < \frac{b_1+l_2}{r_1}$;
- $R_1^* = \frac{(d_1-d_2)r_2+(b_1+b_2)+\sqrt{((d_1-d_2)r_2+b_1+b_2)^2+4d_1r_2(l_1+l_2)}}{2d_1}$,
  - when $d_2 < \frac{b_2+l_1}{r_1+r_2}$, and $d_1 \in [\frac{(b_1+l_2)\left(d_2-\frac{b_2-l_2}{r_2}\right)}{b_2+l_1-r_2d_2}, \frac{d_2(b_1+l_2)}{b_2+l_1-d_2r_2} + \frac{d_2(b_2-l_2)}{b_2+l_1})$; and
  - when $\frac{b_2+l_1}{r_1+r_2} \leq d_2 \leq \frac{b_2+l_1}{r_2}$, and $d_1 \in [\frac{(b_1+l_2)\left(d_2-\frac{b_2-l_2}{r_2}\right)}{b_2+l_1-r_2d_2}, \frac{r_2(l_1+l_2)}{r_1(r_1+r_2)} + \frac{b_1+b_2-d_2r_2}{r_1})$.

□

Solution for Sub-optimization 2

The Lagrangian function for sub-optimization 2 is

$$\begin{aligned} L_2(R, r_2', b_2', \boldsymbol{\lambda}) = {} & R + \lambda_1(R - r_1 - r_2') + \lambda_2\left(\frac{b_1+b_2}{r_1} - \frac{(R-r_1-r_2)(b_2-b_2')}{r_1(r_2'-r_2)} - d_1\right) \\ & + \lambda_3\left(\frac{b_2+l_1}{R} - d_2\right) + \lambda_4\left(\frac{b_2-b_2'}{r_2'} + \frac{l_1+l_2}{R} - d_2\right) \\ & + \lambda_5\left(\frac{b_1+b_2'}{r_1} - \frac{(R-r_2')(b_2-b_2')}{r_1(r_2'-r_2)}\right) + \lambda_6(r_1+r_2-R) \\ & + \lambda_7(r_2-r_2') + \lambda_8(l_2-b_2') + \lambda_9(b_2'-b_2) \end{aligned} \tag{B.41}$$

$$\nabla_{R,r_2',b_2'} L_2 = \begin{bmatrix} 1 + \lambda_1 - \frac{\lambda_2(b_2-b_2')}{r_1(r_2'-r_2)^2} - \frac{\lambda_3(b_2+l_1)}{R^2} - \frac{\lambda_4(l_1+l_2)}{R^2} - \frac{\lambda_5(b_2-b_2')}{r_1(r_2'-r_2)} - \lambda_6 \\ -\lambda_1 + \frac{\lambda_2(R-r_1-r_2)(b_2-b_2')}{r_1(r_2'-r_2)^2} - \frac{\lambda_4(b_2-b_2')}{r_2'^2} + \frac{\lambda_5(b_2-b_2')(R-r_2)}{r_1(r_2'-r_2)^2} - \lambda_7 \\ \frac{\lambda_2(R-r_1-r_2)}{r_1(r_2'-r_2)} - \frac{\lambda_4}{r_2'} + \lambda_5\left(\frac{1}{r_1} + \frac{R-r_2'}{r_1(r_2'-r_2)}\right) - \lambda_8 + \lambda_9 \end{bmatrix} \tag{B.42}$$

$$\mathrm{diag}(\triangle_{R,r_2',b_2'}L_1) = \begin{bmatrix} \frac{2\lambda_3(b_2+l_1)}{R^3} + \frac{2\lambda_4(l_1+l_2)}{R^3} \\ -\frac{2\lambda_2(R-r_1-r_2)(b_2-b_2')}{r_1(r_2'-r_2)^3} + \frac{2\lambda_4(b_2-b_2')}{r_2'^3} - \frac{2\lambda_5(b_2-b_2')(R-r_2)}{r_1(r_2'-r_2)^3} \\ 0 \end{bmatrix} \tag{B.43}$$

Similar as that for suboptimization 1, we have $\lambda_5^* = 0$. Substitute it into Eq. (B.43), we have

$$\mathrm{diag}(\triangle_{R,r_2',b_2'}L_1) = \begin{bmatrix} \frac{2\lambda_3(b_2+l_1)}{R^3} + \frac{2\lambda_4(l_1+l_2)}{R^3} \\ -\frac{2\lambda_2(R-r_1-r_2)(b_2-b_2')}{r_1(r_2'-r_2)^3} + \frac{2\lambda_4(b_2-b_2')}{r_2'^3} \\ 0 \end{bmatrix} \tag{B.44}$$

Next we consider KKT's necessary conditions for the optimization under $\lambda_5^* = 0$. Remember that when solving sub-optimization 2, we only consider $R < r_1 + r_2'$. Similar as before, we consider only $b_2' < b_2$. Therefore, we have:

$$
\begin{aligned}
&1+\lambda_1-\frac{\lambda_2(b_2-b_2')}{r_1(r_2'-r_2)^2}-\frac{\lambda_3(b_2+l_1)}{R^2}-\frac{\lambda_4(l_1+l_2)}{R^2}-\lambda_6=0, && \frac{\lambda_2(R-r_1-r_2)}{r_1(r_2'-r_2)}-\frac{\lambda_4}{r_2'}-\lambda_8+\lambda_9=0,\\
&\frac{\lambda_2(R-r_1-r_2)(b_2-b_2')}{r_1(r_2'-r_2)^2}-\frac{\lambda_4(b_2-b_2')}{r_2'^2}=\lambda_1+\lambda_7, && \lambda_i\geq 0,\ \text{for } i=1,...9,\\
&R-r_1-r_2'<0, && \lambda_1=0,\\
&\frac{b_1+b_2}{r_1}-\frac{(R-r_1-r_2)(b_2-b_2')}{r_1(r_2'-r_2)}-d_1\leq 0, && \lambda_2\left(\frac{b_1+b_2}{r_1}-\frac{(R-r_1-r_2)(b_2-b_2')}{r_1(r_2'-r_2)}-d_1\right)=0,\\
&\frac{b_2+l_1}{R}-d_2\leq 0, && \lambda_3\left(\frac{b_2+l_1}{R}-d_2\right)=0,\\
&\frac{b_2-b_2'}{r_2'}+\frac{l_1+l_2}{R}-d_2\leq 0, && \lambda_4\left(\frac{b_2-b_2'}{r_2'}+\frac{l_1+l_2}{R}-d_2\right)=0,\\
&\frac{b_1+b_2'}{r_1}-\frac{(R-r_2')(b_2-b_2')}{r_1(r_2'-r_2)}<0, && \lambda_5=0,\\
&r_1+r_2-R\leq 0, && \lambda_6(r_1+r_2-R)=0,\\
&r_2-r_2'\leq 0, && \lambda_7(r_2-r_2')=0,\\
&l_2-b_2'\leq 0, && \lambda_8(l_2-b_2')=0,\\
&b_2'-b_2<0 && \lambda_9=0
\end{aligned}
\tag{B.45}
$$

From the conditions in Eq. (B.45), we have:

- when $d_2<\frac{b_2+l_1}{r_1+r_2}$, and $d_1\in[\frac{d_2(b_2-l_2)}{b_2+l_1}+\frac{b_1+l_2}{r_1},\frac{d_2(b_1+b_2)}{b_2+l_1-d_2r_2})$, $R_2^*=\frac{b_2+l_1}{d_2}<R_0^*$;
- otherwise, $R_2^*=R_0^*$.

Proof. When $R=r_1+r_2$, from $\frac{b_1+b_2}{r_1}-\frac{(R-r_1-r_2)(b_2-b_2')}{r_1(r_2'-r_2)}-d_1\leq 0$ we have $\frac{b_1+b_2}{d_1}\leq r_1$, which gives $\frac{b_1+b_2}{d_1}+r_2\leq r_1+r_2$. From Corollary 43, we know that adding shapers cannot decrease the minimum required bandwidth. Therefore, we consider only $R>r_1+r_2$ afterwards. Combine $R>r_1+r_2$ with $R<r_1+r_2'$, we have $r_2'>r_2$. Hence, $\lambda_6=\lambda_7=0$.

As $\lambda_1=0$, we have $\frac{\lambda_2(R-r_1-r_2)(b_2-b'-2)}{r_1(r_2'-r_2)^2}-\frac{\lambda_4(b_2-b_2')}{\lambda_2'^2}=0$. Substitute it to $\frac{\lambda_2(R-r_1-r_2)}{r_1(r_2'-r_2)}-\frac{\lambda_4}{r_2'}-\lambda_8+\lambda_9=0$, we have $-\frac{\lambda_4 r_2}{r_2'^2}-\lambda_8+\lambda_9=0$. As $\lambda_9=0$, *i.e.*, $b_2'\neq b_2$, we have $\lambda_2=\lambda_4=\lambda_8=0$.

As $\lambda_1 = \lambda_6 = 0$, from $1 + \lambda_1 - \frac{\lambda_2(b_2-b_2')}{r_1(r_2'-r_2)^2} - \frac{\lambda_3(b_2+l_1)}{R^2} - \frac{\lambda_4(l_1+l_2)}{R^2} - \lambda_6 = 0$, we have $\lambda_3 > 0$, *i.e.,* $R = \frac{b_2+l_1}{d_2}$. Note that when $\lambda_2 = \lambda_4 = 0$ and $\lambda_3 > 0$, from Eq. (B.43), we have $\mathrm{diag}(\triangle_{R,r_2',b_2'} L_1) \geq 0$. Hence, $R = \frac{b_2+l_1}{d_2}$ is a local optimum.

Hence, the constraints of the optimization reduces to

$$\begin{cases} r_2' \in (R - r_1, R), \qquad b_2' \in [l_2, b_2) \\ \\ \frac{b_1+b_2}{r_1} - \frac{(R-r_1-r_2)(b_2-b_2')}{r_1(r_2'-r_2)} \leq d_1 \implies \frac{b_2-b_2'}{r_2'-r_2} \geq \frac{b_1+b_2-d_1r_1}{R-r_1-r_2}, \\ \\ \frac{b_2-b_2'}{r_2'} + \frac{l_1+l_2}{R} \leq d_2 = \frac{b_2+l_1}{R} \implies b_2 - b_2' \leq \frac{r_2'(b_2-l_2)}{R}, \\ \\ \frac{b_1+b_2'}{r_1} - \frac{(R-r_2')(b_2-b_2')}{r_1(r_2'-r_2)} < 0 \implies \frac{b_2-b_2'}{r_2'-r_2} > \frac{b_1+b_2}{R-r_2}, \end{cases} \tag{B.46}$$

Basic algebraic manipulation gives $\frac{b_1+b_2}{R-r_2} > \frac{b_1+b_2-d_1r_1}{R-r_1-r_2}$ iff $R > \frac{b_1+b_2}{d_1} + r_2 = R_0^*$. Hence, we only consider the case where $\frac{b_1+b_2}{R-r_2} < \frac{b_1+b_2-d_1r_1}{R-r_1-r_2}$. Under such a condition, we have $b_2 - b_2' \in [\frac{(b_1+b_2-d_1r_1)(r_2'-r_2)}{R-r_1-r_2}, \frac{r_2'(b_2-l_2)}{R}]$.

Next we configure the conditions where:

$$\exists\ r_2' \in (R - r_1, R),\ \text{s.t.}\ S := [\frac{(b_1 + b_2 - d_1r_1)(r_2' - r_2)}{R - r_1 - r_2}, \frac{r_2'(b_2 - l_2)}{R}] \cap (0, b_2 - l_2] \neq \emptyset.$$

When $R^* < R_0^*$, it has $\frac{b_1+b_2}{d_1} + r_2 > r_1 + r_2$, *i.e.,* $b_1 + b_2 - d_1r_1 > 0$. Hence we have $\frac{(b_1+b_2-d_1r_1)(r_2'-r_2)}{R-r_1-r_2} > 0$. From $r_2' < R$, we have $\frac{r_2'(b_2-l_2)}{R} < b_2 - l_2$. Therefore, $S \neq \emptyset$ is equivalent to

$$\exists\ r_2' \in (R - r_1, R)\ \text{s.t.}\ \frac{(b_1 + b_2 - d_1r_1)(r_2' - r_2)}{R - r_1 - r_2} \leq \frac{r_2'(b_2 - l_2)}{R}.$$

Define

$$g(x) = \frac{x(b_2 - l_2)}{R} - \frac{(b_1 + b_2 - d_1 r_1)(x - r_2)}{R - r_1 - r_2}.$$

As $g(x)$ is linear w.r.t x, $S \neq \emptyset$ is equivalent to at least one of $g(R)$ and $g(R - r_1)$ is non-negative, which gives $d_1 > \frac{b_1 + l_2}{r_1}$ and $\frac{b_2 + l_1}{d_2} > \frac{r_1(b_2 - l_2)}{d_1 r_1 - b_1 - l_2}$ through basic algebraic manipulations. Combine it with $\frac{b_1 + b_2}{d_1} + r_2 > \frac{b_2 + l_1}{d_2} \geq r_1 + r_2$, we have:

$$\circ\ d_2 < \frac{b_2 + l_1}{r_1 + r_2},\ \text{and}\ d_1 \in [\frac{d_2(b_2 - l_2)}{b_2 + l_1} + \frac{b_1 + l_2}{r_1}, \frac{d_2(b_1 + b_2)}{b_2 + l_1 - d_2 r_2}).$$

□

B.7 Proofs for Multiple-node cases

B.7.1 Proofs for Algorithm 2

Proposition 49. *For $1 \leq i \leq n$, Algorithm 2 results in $0 \leq b'_i \leq b_i$, $0 \leq \hat{d}'_i \leq \hat{d}_i$, and keeps the link deadline ordering.*

Proof. We first show that Algorithm 2 keeps the link deadline ordering. Remember that reshaping flow i to b'_i results a reshaping delay of $\frac{b_i - b'_i}{r_i}$. Therefore, $\hat{d}'_i = hatd_i - \frac{b_i - b'_i}{r_i}$ is the link deadline after reshaping. Combining it with $b'_i \geq \max\left\{0, b_i - r_i\left(\hat{d}_i - \hat{d}'_{i+1}\right)\right\} \geq b_i - r_i\left(\hat{d}_i - \hat{d}'_{i+1}\right)$ gives $\hat{d}'_{i+1} \leq \hat{d}'_i$. Thus, Algorithm 2 keeps the link deadline ordering.

Next we show that $0 \leq \hat{d}'_i \leq \hat{d}_i$. From $\hat{d}'_i = \hat{d}_i - \frac{b_i - b'_i}{r_i}$, $\hat{d}'_i \leq \hat{d}_i$. From $\hat{d}'_{i+1} \leq \hat{d}'_i$, $\hat{d}'_i \geq \hat{d}'_{n+1} = 0$.

Finally we show $0 \leq b'_i \leq b_i$. From $b'_i \geq \max\left\{0, b_i - r_i\left(\hat{d}_i - \hat{d}'_{i+1}\right)\right\}$ we have $b'_i \geq 0$. Next we show $b'_i \leq b_i$. When $R^* = \sum_{j \geq i}$, it has $b'_i = \left[b_i - r_i \hat{d}_i\right]^+ \leq b_i$, whereas when $b_i \leq r_i \hat{d}_i$, it has $b'_i = 0$. Below consider the case when $R^* > \sum_{j \geq i} r_j$ and $b_i > r_i \hat{d}_i$. Define

$\frac{r_i}{R-\sum_{j\geq i} r_j}\left[\frac{b_i - r_i\hat{d}_i}{r_i}\left(R^* - \sum_{j>i} r_j\right) + \sum_{j>i}\left(b'_j - d'_j r_j\right)\right] = x_i$, which by basic algebraic manipulation is equivalent to $R^* = \frac{x_i + \sum_{j>i}\left(b'_j - \hat{d}'_j r_j\right) + \sum_{j>i} r_j\left(\hat{d}_i - \frac{b_i - x_i}{r_i}\right)}{\hat{d}_i - \frac{b_i - x_i}{r_i}}$, whose r.h.s decreases with x_i. When $x_i = b_i$, the r.h.s defaults to $\frac{b_i + \sum_{j>i}\left(b'_j - \hat{d}'_j r_j\right) + \sum_{j>i} r_j \hat{d}_i}{\hat{d}_i} = \frac{b_i + \sum_{j>i}\left(b_j - \hat{d}_j r_j\right) + \sum_{j>i} r_j \hat{d}_i}{\hat{d}_i}$, which from Eq 3.3 is no larger than R^*. Thus it has $x_i \leq b_i$. □

Proposition 50. *Reshaping according to Algorithm 2 will not increase the hop's minimum required bandwidth.*

Proof. From Algorithm 2 it has $b'_i \geq \frac{r_i}{R-\sum_{j\geq i} r_j}\left[\frac{b_i - r_i\hat{d}_i}{r_i}\left(R^* - \sum_{j>i} r_j\right) + \sum_{j>i}\left(b'_j - d'_j r_j\right)\right]$, which from basic algebraic manipulation is equivalent to $R^*\left(\hat{d}_i - \frac{b_i - b'_i}{r_i}\right) \geq b'_i + \sum_{j>i}\left(b'_j - r_j\hat{d}'_j\right) + \left(\hat{d}_i - \frac{b_i - b'_i}{r_i}\right)\sum_{j>i} r_j$. Note that its r.h.s is the minimum required amount of data for the system to process before time $\hat{d}_i - \frac{b_i - b'_i}{r_i}$ without violating flow i's deadline. Therefore, at any flow's reshaped deadline, the sum of all flows' shifted arrival curve will not increase the bandwidth. As the minimum required bandwidth, if strictly greater then $\sum_{j\geq i} r_j$, must be achieved at a specific flow's deadline, we know that reshaping according to Algorithm 2 will not increase the minimum required bandwidth. □

B.7.2 Proofs for Proposition 21

Proposition 21. *Consider a network with k nodes and n flows, where flow $1 \leq i \leq n$ has a token-bucket arrival curve $\mathcal{AC}_i(t) = b_i + r_i t$, an end-to-end deadline of d_i, and a path of $\boldsymbol{M}_i = [m_{i,1}, ..., m_{i,k_i}]$, where M_i contains unique nodes $1 \leq m_{i,j} \leq k$ for all $1 \leq j \leq k_i$, and there guarantees to be a link between $m_{i,j}$ and $m_{i,j+1}$ for all $1 \leq j \leq k_i - 1$. Suppose a per-hop reshaping mechanism reshapes flow i to $b'_{i,j}$ and assigns a link deadline of $d_{i,j}$ at node $m_{i,j}$. W.l.o.g, suppose $b'_{i,j} \geq b'_{i,j+1}$ for all $1 \leq j \leq k_i - 1$. Then under EDF, the mechanism that*

- *allocates flow i an ingress reshaper of b'_{i,k_i} at every hop along $\boldsymbol{M}_i$; and*
- *allocates a per-hop deadline of $d_{i,j} - \frac{b'_{i,j-1}-b'_{i,j}}{r_i^{(2)}}$ at node $m_{i,j}$, where $1 \leq j \leq k_i$ and for simplicity define $b'_{i,0} = 0$,*

performs no worse than the corresponding per-hop reshaping mechanism.

Proof. For flow i, as the per-hop mechanism results in a reshaping delay of $\frac{b'_{i,j-1}-b'_{i,j}}{r_i}$ at node $m_{i,j}$, it leaves a per-hop link deadline of $d_{i,j} - \frac{b'_{i,j-1}-b'_{i,j}}{r_i}$ at node $m_{i,j}$, and yields an aggregate reshaping delay of $\sum_{1\leq j\leq k_i} \frac{b'_{i,j-1}-b'_{i,j}}{r_i} = \frac{b'_{i,0}-b'_{i,k_i}}{r_i} = \frac{b_i-b'_{i,k_i}}{r_i}$ along the path. Therefore, both the per-hop and ingress reshaping mechanisms assign the same per-hop link deadline for each flow. Note that compared to per-hop reshaping, at each hop ingress reshaping generates a smaller (point-wise) arrival curve for each flow, which in turn results in a smaller minimum required bandwidth. Thus, the ingress reshaping algorithm proposed in Proposition 21 dominates the corresponding per-hop reshaping mechanism. □

B.7.3 Proofs for Algorithm 5

In this part we show the correctness of Algorithm 5. Specifically, from Lemma 51, checking only points inside $\mathbb{T}$ is enough to figure out whether there is any room for reshaping. Combining Lemma 52 with Proposition 10 guarantees that we can use binary search to characterize the minimum reshaper.

Lemma 51. *Define* $\mathcal{AC}_i(t) = \begin{cases} \min\left\{r_i^{(1)}t, r_i^{(2)}t + b_i\right\}, & \text{when } t \geq 0 \\ 0, & \text{else} \end{cases}$. *Consider function* $\mathcal{F}(t) = \sum_{1\leq i\leq n} \mathcal{AC}_i(t - \hat{d}_i)$. *Suppose* $\sup_{t\geq 0} \frac{\mathcal{F}(t)}{t} = R^*$. *Call t_0 a critical point if* $\frac{\mathcal{F}(t_0)}{t_0} = R^*$.

Define $t_i = \begin{cases} \dfrac{b_i}{r_i^{(1)} - r_i^{(2)}}, & \text{if } r_i^{(1)} > r_i^{(2)} \\ 0, & \text{otherwise} \end{cases}$, *where* $r_i^{(1)} \geq r_i^{(2)} > 0$. *Then all the critical points belong to* $\left\{\hat{d}_i + t_i \mid 1 \leq i \leq n\right\}$.

Proof. As $\mathcal{F}(t)$ is piece-wise linear and continuous, the critical points can only be the turning points of $\mathcal{F}(t)$, *i.e.,* $\hat{d}_i$ or $\hat{d}_i + t_i$. When $r_i^{(1)} = r_i^{(2)}$, from $t_i = 0$ it has $\hat{d}_i = \hat{d}_i + t_i$. Afterwards we consider the case when $r_i^{(1)} \neq r_i^{(2)}$.

Observe first that a critical point t_0 satisfies $\lim_{\epsilon \to 0} \frac{\mathcal{F}(t_0) - \mathcal{F}(t_0 - \epsilon)}{\epsilon} \geq R^*$, as otherwise there exists $\epsilon_0 > 0$ such that $\frac{\mathcal{F}(t_0 - \epsilon_0)}{t_0 - \epsilon_0} > R^*$. Specifically, suppose $\lim_{\epsilon \to 0} \frac{\mathcal{F}(t_0) - \mathcal{F}(t_0 - \epsilon)}{\epsilon} < R^*$. Then basic algebraic manipulation shows that $\frac{\mathcal{F}(t)}{t} \geq \frac{\mathcal{F}(t_0 - \epsilon_0)}{t_0 - \epsilon_0}$ iff $\lim_{\epsilon \to 0} \frac{\mathcal{F}(t_0) - \mathcal{F}(t_0 - \epsilon)}{\epsilon} \geq R^*$, which contradicts to the assumption.

Based on the above observation, we then show the Lemma. We show that $\hat{d}_i$ cannot be a critical point by contradiction. Suppose $\hat{d}_i$ is a critical point, *i.e.,* $\frac{\mathcal{F}(\hat{d}_i)}{\hat{d}_i} = R^*$. Then by definition of $\mathcal{F}$ we have $\lim_{\epsilon \to 0} \frac{\mathcal{F}(\hat{d}_i + \epsilon) - \mathcal{F}(\hat{d}_i)}{\epsilon} > R^*$. For $0 < \epsilon_0 < t_i$, we have $\frac{\mathcal{F}(\hat{d}_i + \epsilon_0)}{\hat{d}_i + \epsilon_0)} = \frac{\mathcal{F}(\hat{d}_i) + \epsilon_0 \lim_{\epsilon \to 0} \frac{\mathcal{F}(\hat{d}_i + \epsilon) - \mathcal{F}(\hat{d}_i)}{\epsilon}}{\hat{d}_i + \epsilon_0} > \frac{\mathcal{F}(\hat{d}_i) + \epsilon_0 R^*}{\hat{d}_i + \epsilon_0}$. Since $\hat{d}_i + \epsilon_0$ is not a turning point of $\mathcal{F}$, we have $\frac{\mathcal{F}(\hat{d}_i + \epsilon_0)}{\hat{d}_i + \epsilon_0)} < R^*$. However, from basic algebraic manipulation, $\frac{\mathcal{F}(\hat{d}_i + \epsilon_0)}{\hat{d}_i + \epsilon_0)} < \frac{\mathcal{F}(\hat{d}_i)}{\hat{d}_i}$ iff $\epsilon_0 < 0$. □

Lemma 52. *For* $1 \leq i \leq n$, *define* $\mathcal{AC}_i(t \mid b_i) = \begin{cases} \min\left\{r_i^{(1)} t, r_i^{(2)} t + b_i\right\}, & \text{when } t \geq 0 \\ 0, & \text{else} \end{cases}$.

Then for all $t \geq 0$, $\mathcal{AC}_i\left(t - \hat{d}_i' + \frac{b_i - b}{r_i^{(2)}} \mid b\right) + \sum_{j \neq i} \mathcal{AC}_j\left(t - \hat{d}_j' \mid b_j'\right)$ *decreases with* b.

Proof. Since $\sum_{j\neq i}\mathcal{AC}_j\left(t-\hat{d}'_j \mid b'_j\right)$ is independent with b, below we show

$$\mathcal{AC}_i\left(t-\hat{d}'_i+\frac{b_i-b}{r_i^{(2)}} \mid b\right)=\begin{cases} 0, & \text{when } t\leq \hat{d}'_i-\frac{b_i-b}{r_i^{(2)}} \\ r_i^{(1)}\left(t-\hat{d}'_i+\frac{b_i-b}{r_i^{(2)}}\right), & \text{when } \hat{d}'_i-\frac{b_i-b}{r_i^{(2)}}<t\leq \frac{b}{r^{(1)}-r^{(2)}}+\hat{d}'_i-\frac{b_i-b}{r_i^{(2)}} \\ b+r_i^{(2)}\left(t-\hat{d}'_i+\frac{b_i-b}{r_i^{(2)}}\right), & \text{otherwise} \end{cases}$$

decreases with b. As $\mathcal{AC}_i(t)=0$ when $t\leq \hat{d}'_i-\frac{b_i-b}{r_i^{(2)}}$, next we consider only $t>\hat{d}'_i-\frac{b_i-b}{r_i^{(2)}}$.

Define $\Delta\mathcal{AC}_i=\mathcal{AC}_i\left(t-\hat{d}'_i+\frac{b_i-b-\Delta b}{r_i^{(2)}} \mid b+\Delta b\right)-\mathcal{AC}_i\left(t-\hat{d}'_i+\frac{b_i-b}{r_i^{(2)}} \mid b\right)$, where $\Delta b>0$. By definition $\mathcal{AC}_i\left(t-\hat{d}'_i+\frac{b_i-b}{r_i^{(2)}} \mid b\right)$ decreases with b iff $\Delta\mathcal{AC}_i\leq 0$ for all $\Delta b>0$.

When $t\leq \frac{b}{r^{(1)}-r^{(2)}}+\hat{d}'_i-\frac{b_i-b}{r_i^{(2)}}$ or $t\geq \frac{b+\Delta b}{r^{(1)}-r^{(2)}}+\hat{d}'_i-\frac{b_i-b-\Delta b}{r_i^{(2)}}$, both $\mathcal{AC}_i\left(t-\hat{d}'_i+\frac{b_i-b-\Delta b}{r_i^{(2)}} \mid b+\Delta b\right)$ and $\mathcal{AC}_i\left(t-\hat{d}'_i+\frac{b_i-b}{r_i^{(2)}} \mid b\right)$ go into the same condition of $\mathcal{AC}_i$. From $b+r_i^{(2)}\left(t-\hat{d}'_i+\frac{b_i-b}{r_i^{(2)}}\right)=r_i^{(2)}\left(t-\hat{d}'_i+\frac{b_i}{r_i^{(2)}}\right)$ and $r_i^{(1)}\left(t-\hat{d}'_i+\frac{b_i-b}{r_i^{(2)}}\right)$ decrease with b, we have $\Delta\mathcal{AC}_i\leq 0$.

When $t\in\left(\frac{b}{r^{(1)}-r^{(2)}}+\hat{d}'_i-\frac{b_i-b}{r_i^{(2)}},\ \frac{b+\Delta b}{r^{(1)}-r^{(2)}}+\hat{d}'_i-\frac{b_i-b-\Delta b}{r_i^{(2)}}\right)$, basic algebraic manipulation gives that $\Delta\mathcal{AC}_i=\left(r_i^{(2)}-r_i^{(1)}\right)\left(t-\hat{d}'_i+\frac{b_i}{r_i^{(2)}}\right)+\frac{r_i^{(1)}}{r_i^{(2)}}b$. Thus, $\Delta\mathcal{AC}_i\leq 0$ iff $t\geq \frac{r_i^{(1)}b}{r_i^{(2)}\left(r_i^{(1)}-r_i^{(2)}\right)}+\hat{d}'_i-\frac{b_i}{r_i^{(2)}}$, *i.e.*, $t\geq \frac{b}{r_i^{(1)}-r_i^{(2)}}+\hat{d}'_i-\frac{b_i-b}{r_i^{(2)}}$. Therefore, when $t\in\left(\frac{b}{r^{(1)}-r^{(2)}}+\hat{d}'_i-\frac{b_i-b}{r_i^{(2)}},\ \frac{b+\Delta b}{r^{(1)}-r^{(2)}}+\hat{d}'_i-\frac{b_i-b-\Delta b}{r_i^{(2)}}\right)$, $\Delta\mathcal{AC}_i<0$, *i.e.,* decreases with b.

As $\mathcal{AC}_i\left(t-\hat{d}'_i+\frac{b_i-b}{r_i^{(2)}}\right)$ is continuous with b, combining both cases gives that $\Delta\mathcal{AC}_i\leq 0$ for all $Deltab>0$, *i.e.,* $\mathcal{AC}_i\left(t-\hat{d}'_i+\frac{b_i-b}{r_i^{(2)}} \mid b+\Delta b\right)+\sum_{j\neq i}\mathcal{AC}_j\left(t-\hat{d}'_j \mid b\right)$ decreases with b □