Minimizing network bandwidth under latency constraints: The single node case

Jiayi Song, Roch Guérin
Washington University in St. Louis
(jiayisong,guerin)@wustl.edu

Henry Sariowan
Google, USA
hsariowan@google.com

***Abstract*—Much of today's 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. Of particular interest is the extent to which traffic (re)shaping can be of benefit. The paper focuses on the most basic network configuration, namely, a single link network, with extensions to more general, multi-node networks discussed in a companion paper. The main results are in the form of optimal solutions for different types of schedulers of varying complexity. They demonstrate how judicious traffic shaping can help lower complexity schedulers perform nearly as well as more complex ones.**

***Index Terms*—latency, bandwidth, optimization, shaping**

I. Introduction

The networks that connect datacenters are now on par with those from major Internet Service Providers (ISPs) [1]. Furthermore, 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 [2, 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, fine tuning of performance is now not only possible [3], but also highly desirable to minimize cost [4].

The paper assumes such an environment with datacenters inter-connected by links under the purview of an operator with both knowledge and control of individual traffic flows (or flow aggregates) traversing the network. Flows are assigned traffic contracts and corresponding latency bounds[1] with a central controller responsible for orchestrating how they are mapped to network resources [5]. In such a setting, the paper 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 type of scheduler used in the network affects the answer, and we consider options of varying complexity.

We note that this question is the dual of the traditional *call admission* problem that asks whether performance goals can be met *given* the available network capacity. The added complexity in this dual problem is in the exploration of possible configurations in handling individual flows (how to best map each flow to scheduling decisions).

This paper represents a first step in investigating this problem, and focuses on the most basic network configuration, namely, a *single node and link* (hop). Its contributions are in formulating optimal solutions for schedulers of different complexity in the single hop case, *i.e.,* a dynamic priority (service curve-based) scheduler, followed by 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. For the latter two schedulers, another question of interest is the improvements that may be feasible from reshaping flows on ingress. Reshaping adds an access delay component, but makes flows easier (smoother) to handle. The paper identifies optimal ingress traffic (re)shaping configurations that minimize the link bandwidth required to meet end-to-end deadlines (inclusive of shaping delays) under both schedulers.

The rest of this paper is structured as follows. Section II offers a brief review of related works. Section III introduces our one-hop "network" model and associated optimization, with Sections IV to VI devoted to deriving solutions for schedulers of different complexity. Section VII 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. Section VIII summarizes the paper's findings and identifies extensions. Due to lack of space, proofs and ancillary results are in [6].

II. Related Works

There is a vast literature on traffic engineering and scheduling in data centers, but it has been mainly focused on optimizing performance rather than minimizing cost (under performance constraints), as we do. From that perspective, works such as [5], [7], [8] and [9] are closest conceptually.

Reducing costs while meeting latency performance requirements is a goal we share with [7], 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, [7] is concerned with minimizing the number of servers needed to accommodate a workload with given deadlines, whereas our focus is on

The work of the first two authors was supported in part by a gift from Google and by NSF grant CNS 2006530.

[1]We note that the deterministic nature of both is largely meant to facilitate the validation of contractual agreements.

minimizing network bandwidth given a set of inter-datacenter flows and deadlines. As a result, [7] 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 [8], 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 [9], as is minimizing (network) cost while meeting target deadlines. Specifically, [9] relotheries 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 [9] that focuses on meeting long-term traffic volume targets on a single link (the ISP link whose cost is to be optimized).

Finally, we note that the theory of "network calculus" [10] involves a similar framework as that of the paper, but its applicability manifests itself mainly in the multi-node setting where it offers powerful tools to analyze end-to-end network performance (see [6] for a brief discussion of relevant results).

III. General Model Formulation

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

Consider a set of n flows, where flow $i, 1 \leq i \leq n$, is associated with an end-to-end packet-level deadline[2] 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 link "cost."

Our cost metric is link bandwidth. In the simple case of one-hop (one link) networks illustrated in Fig. 1, we let R denote the link bandwidth, and define the vectors of the rates, burst sizes, and deadlines of the flows sharing the link as $\boldsymbol{r} = (r_1, r_2, \ldots, r_n)$, $\boldsymbol{b} = (b_1, b_2, \ldots, b_n)$, and $\boldsymbol{d} = (d_1, d_2, \ldots, d_n)$, respectively. 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:

$$\textbf{OPT} \quad \min \| R \|_1$$
$$\text{s.t} \quad D_i^*(\boldsymbol{r}, \boldsymbol{b}, R) \leq d_i, \quad \forall i, 1 \leq i \leq n.$$

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

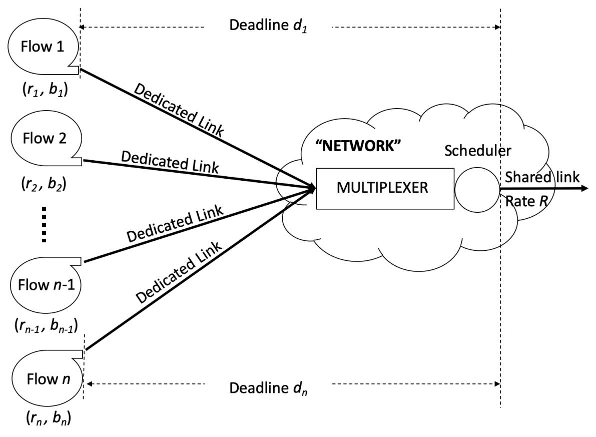


Fig. 1: A typical one-hop configuration with n flows.

The next three sections explore solutions to **OPT** for different schedulers. For simplicity of exposition and analysis, results are presented using a fluid model. [6] presents a solution for a static priority scheduler under a packet-based model, but the results do not contribute additional insight.

IV. 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 1. *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 $d_1 > d_2 > \ldots > d_n$ and $d_1 < \infty$. Consider a service-curve assignment Γ_{sc} that allocates flow i a service curve of*

$$SC_i(t) = \begin{cases} 0 & \textit{when } t < d_i, \\ b_i + r_i(t - d_i) & \textit{otherwise.} \end{cases} \tag{1}$$

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*

$$R^* = \max_{1 \leq h \leq n} \left\{ \sum_{i=1}^{n} r_i, \frac{\sum_{i=h}^{n} b_i + r_i(d_h - d_i)}{d_h} \right\}. \tag{2}$$

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

The optimality of Γ_{sc} is intuitive. Recall that a service curve is a lower bound on the service received by a flow. Eq. (1) 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. (1) under R^*, and consequently providing a solution to **OPT** for schedulers that support dynamic priorities.

Proposition 2. *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^*.*

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 [11].

Note also 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 [12], could be used instead. Unfortunately, it is easy to find scenarios where linear service curves perform worse.

In the next section, we consider simpler, static priority schedulers; first used alone (Section V-A) and then combined with an ingress (re)shaper (Section V-B). As is intuitive given their optimality and formally shown in [6], reshaping is of no benefit with the dynamic priority schedulers of this section.

V. Static Priorities

Though dynamic priorities are efficient and may be realizable [13], [14], they are expensive and not feasible in all environments. It is, therefore, of 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 3. *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 $d_i < d_j$.*

We note that while Proposition 3 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 necessarily be mapped to distinct priorities. More generally, in some scenarios, grouping flows with different deadlines in the same priority class can result in a lower bandwidth than mapping them to distinct classes[3]. Nevertheless, motivated by Proposition 3, we propose a simple assignment rule that strictly maps lower deadlines to higher priorities.

A. Static Priorities without (re)Shaping

From [10, 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 minimum link bandwidth $\widetilde{R}^*$ to ensure that flow h's deadline d_h is met for all h is given by:

$$\widetilde{R}^* = \max_{1 \leq h \leq n} \left\{ \sum_{i=1}^{n} r_i, \frac{\sum_{i=h}^{n} b_i}{d_h} + \sum_{i=h+1}^{n} r_i \right\} \tag{3}$$

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

$$R^* = \max_{1 \leq h \leq n} \left\{ \sum_{i=1}^{n} r_i, \frac{\sum_{i=h}^{n} b_i}{d_h} + \sum_{i=h+1}^{n} r_i \left(1 - \frac{d_i}{d_h}\right) \right\}. \tag{4}$$

Comparing Eqs. (3) and (4) 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

$$\frac{\sum_{i=h^*+1}^{n} d_i r_i}{\sum_{i=h^*}^{n} b_i\ + \sum_{i=h^*+1}^{n} r_i \left(d_{h^*} - d_i\right)}. \tag{5}$$

[3]We illustrate this in [6] with two flows and a static priority scheduler.

As Eq. (5) 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 $\frac{\sum_{i=h^*+1}^{n} r_i d_{h^*}}{\sum_{i=h^*}^{n} b_i}$. This is intuitive, as when flows have arbitrarily close deadlines, they should receive mostly equal service shares, which conflicts 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.

B. Static Priorities with (re)Shaping

As shown, static priorities can result in a minimum required bandwidth significantly larger than R^*. This is because they are a blunt instrument when it comes to allocating transmission opportunities as a function of packet deadlines.

This is intrinsic to the static structure of the scheduler's decision, but can be mitigated by anticipating the extent to which flows may experience better deadlines than necessary and (re)shaping them before they enter the network. Such flows can absorb the added (ingress) reshaping delay, and reshaping them limits their impact on lower priority flows.

Consider a 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, a strict static-priority scheduler yields $\widetilde{R}^* = 11.14$. 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[4], but also entirely eliminates its burst. Under a fluid model, the required bandwidth to meet both flows' deadlines is now only 7.57 (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 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 without violating the flow's deadline d_i. In configurations with only two flows, it can be shown (see [6]) that reshaping profiles of the form (r_i, b_i'), *i.e.,* limited to the flow's burst, are sufficient. 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.

[4]Reshaping a burst of size $b_2 = 5$ down to 0 at a rate of $r_2 = 4$ results in a reshaping delay of $\frac{5}{4} = 1.25$, which consumes the entire end-to-end deadline d_2.

Specifically, Proposition 4 characterizes the worst case delays (ingress reshaping plus link scheduling delays) 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 again assigned to a priority class based on their deadline (shorter deadlines have higher priority). The optimization variables are the flows ingress reshaping parameters. Proposition 5 then characterizes the minimum bandwidth $\widetilde{R}_s^*$ that **OPT_S** can achieve, while Proposition 6 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 $\boldsymbol{b}'^* = (b_1'^*, b_2'^*, b_3'^*, ..., b_n'^*)$ 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 $i > n$. Proposition 4 gives flow i's worst-case end-to-end delay.

Proposition 4. *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*

$$D_i^* = max\left\{\frac{b_i + B_{i+1}'}{R - R_{i+1}},\ \frac{b_i - b_i'}{r_i} + \frac{B_{i+1}'}{R - R_{i+1}}\right\}. \quad (6)$$

Observe 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.

Combining Proposition 4 with **OPT**, and using the fact that flow 1 does not need to be reshaped gives optimization **OPT_S**. 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 $\sum_{i=1}^{n} r_i = R_1 \leq R$.

$$\begin{aligned} \textbf{OPT\_S} \quad & \min_{\boldsymbol{b}'} R \\ \text{s.t} \quad & \max\left\{\frac{b_i + B_{i+1}'}{R - R_{i+1}},\ \frac{b_i - b_i'}{r_i} + \frac{B_{i+1}'}{R - R_{i+1}}\right\} \leq d_i, \\ & \forall\ 1 \leq i \leq n, \\ & R_1 \leq R, \quad b_1' = b_1, \quad 0 \leq b_i' \leq b_i, \quad \forall\ 2 \leq i \leq n. \end{aligned} \quad (7)$$

The solution of **OPT_S** is given in Propositions 5 and 6. Proposition 5 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 5. *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 $2 \le i \le n$. Then we have $\widetilde{R}_s^* = \max\{R_1, \inf\{R \mid \forall s \in \mathbb{S}_n(R), s \ge 0\}\}$.

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 $R_1 \le \widetilde{R}_s^* \le \widetilde{R}^*$, we can use a binary search to compute $\widetilde{R}_s^*$ from the relation $\widetilde{R}_s^* = \max\{R_1, \inf\{R \mid \forall s \in \mathbb{S}_n(R), s \ge 0\}\}$ in Proposition 5.

Proposition 6 gives a constructive procedure for the optimal reshaping parameters given $\widetilde{R}_s^*$ and the original flow profiles.

Proposition 6. *Optimal reshaping parameters $b_i'^*, 2 \le i \le n$, satisfy*

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

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

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

VI. Basic FIFO with (re)Shaping

Next, we consider a simple first-in-first-out (fifo) scheduler. For conciseness, we directly assume that flows can be reshaped prior to entering the network. Given again a set of n flows with profiles $(r_i, b_i, d_i), 1 \le i \le n$, our goal is to find reshaping parameters (r_i', b_i') to minimize the link bandwidth required to meet all flows' deadlines. As in Section V-B, we assume that $r_i = r_i'$ and focus on identifying the best b_i' values.

We first proceed to characterize the worst case delay across n flows sharing a fifo scheduler and a link of bandwidth R, when the flows have traffic envelopes $(r_i, b_i), 1 \le i \le n$, and have been reshaped to $(r_i, b_i'), 1 \le i \le n$. Using this result, we then identify the reshaping parameters $b_i', 1 \le i \le n$, that minimize the link bandwidth required to ensure that all flows meet their deadlines. As for other configurations, we only state the results, with proofs found in [6].

Proposition 7. *Consider a system with n rate-controlled flows with traffic envelopes (r_i, b_i), where $1 \le i \le n$, which share a fifo link with bandwidth $R \ge R_1 = \sum_{j=1}^n r_j$. Assume that the system reshapes flow i's traffic envelope to (r_i, b_i'). Then the worst-case delay for flow i is*

$$\widehat{D}_i^* = \max\left\{\frac{b_i - b_i'}{r_i} + \frac{\sum_{j \ne i} b_j'}{R}, \frac{\sum_{j=1}^n b_j'}{R} + \frac{(b_i - b_i')R_1}{r_i R}\right\}. \tag{8}$$

With the result of Proposition 7 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 flows' deadlines $d_1 > d_2 > \ldots > d_n$, with $d_1 < \infty$. Specifically, combining Proposition 7 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, $\sum_{i=1}^n r_i = R_1 \le R$.

OPT_F $\min_{\boldsymbol{b}'} R$

s.t. $\forall\ 1 \le i \le n$

$$\max\left\{\frac{b_i - b_i'}{r_i} + \frac{\sum_{j \ne i} b_j'}{R}, \frac{\sum_{j=1}^n b_j'}{R} + \frac{(b_i - b_i')R_1}{r_i R}\right\} \le d_i,$$

$R_1 \le R, \quad 0 \le b_i' \le b_i.$

The solution of **OPT_F** is characterized in Propositions 8 and 9. As in the case of a static priority scheduler, Proposition 8 describes a numerical procedure to compute the optimal bandwidth $\widehat{R}_s^*$ given the flow profiles, while Proposition 9 specifies the optimal reshaping parameters $\widehat{\boldsymbol{b}}'^*$ given $\widehat{R}_s^*$ and the flow profiles.

Proposition 8. *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*

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

$$y_F(R) = \min_{\substack{P_1, P_2 \subseteq \mathbb{Z}_i \\ P_1 \bigcap P_2 = \emptyset \\ P_1 \bigcup P_2 \ne \emptyset}} \left\{\frac{\widehat{B}_i - \sum_{j \in P_1} \frac{RH_j}{R + r_j} - \sum_{j \in P_2}\left(b_j - \frac{r_j d_j R}{R_1}\right)}{\sum_{j \in P_1} \frac{r_j}{R + r_j} + \sum_{j \in P_2} \frac{r_j}{R_1}}\right\},$$

*and $Y_F(R) = \min_{1 \le i \le n-1}\left\{\widehat{B}_n, Rd_n, y_F(R)\right\}$. Then the optimal solution for **OPT_F** is*

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

Denoting as $\widehat{R}^*$ the minimum required bandwidth in a fifo system without reshaping, since $\max\left\{R_1, \frac{\widehat{B}_n R_1}{\sum_{i=1}^n r_i d_i}\right\} \le \widehat{R}_s^* \le \widehat{R}^* = \max\left\{R_1, \frac{\widehat{B}_n}{d_n}\right\}$, we can use a binary search to compute $\widehat{R}_s^*$ based on Proposition 8. Once $\widehat{R}_s^*$ is known, the optimal reshaping parameters can be obtained as stated in Proposition 9.

Proposition 9. *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 $2 \le i \le n$, where $\widehat{\boldsymbol{B}}'^*$ satisfy*

$$\begin{cases} \widehat{B}_i' = \max\left\{\sum_{j=1}^{i} T_j(\widehat{B}_n'^*, \widehat{R}_s^*), \widehat{B}_{i+1}'^* - b_{i+1}\right\}, i \ne n \\ \widehat{B}_n'^* = X_F(\widehat{R}_s^*), \end{cases} \tag{9}$$

Note that $\widehat{B}_i'^*, 1 \le i \le n-1$, depend on $\widehat{R}_s^*$, $\widehat{B}_n'^*$, $\widehat{B}_{i+1}'^*$ and flow profiles, while $\widehat{B}_n'^*$ relies only on $\widehat{R}_s^*$ and flow profiles. Hence, we can recursively characterize $\widehat{B}_i'^*$ from $\widehat{B}_n'^*$.

VII. Evaluation

This section explores the relative performance of the solutions of the previous sections. One aspect is the "cost of simplicity," as measured by the additional bandwidth simpler static priority or fifo schedulers require compared to a more complex (edf-based) dynamic priority scheduler. Also of interest is the magnitude of the improvements that (ingress) reshaping affords with static priority and fifo schedulers.

The evaluation initially focuses (Section VII-A) 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 VII-B) to more "general" scenarios involving multiple flows with different combinations of deadlines and traffic envelopes.

A. Basic Two-Flow Configurations

We first recall our earlier notation for the minimum bandwidth required in each configuration, namely, R^* (dynamic priority); $\widetilde{R}^*$ (static priority); $\widetilde{R}^*_s$ (static priority w/ reshaping); $\widehat{R}^*$ (fifo); and $\widehat{R}^*_s$ (fifo w/ reshaping).

We then proceed to specialize Eq. (2) to a configuration with only two flows, (r_1, b_1, d_1) and (r_2, b_2, d_2), to get

$$R^* = \max\left\{r_1 + r_2,\ \frac{b_2}{d_2},\ \frac{b_1 + b_2 - r_2 d_2}{d_1} + r_2\right\}, \tag{10}$$

We consider next the same two flows configuration with a static priority scheduler, for which Eq. (3) gives

$$\widetilde{R}^* = \max\left\{r_1 + r_2,\ \frac{b_2}{d_2},\ \frac{b_1 + b_2}{d_1} + r_2\right\}; \tag{11}$$

If (optimal) reshaping is introduced, specializing Proposition 5 to two flows, the minimum bandwidth $\widetilde{R}^*_s$ reduces to

$$\widetilde{R}^*_s = \begin{cases} \max\left\{r_1 + r_2,\ \dfrac{b_2}{d_2},\ \dfrac{b_1 + b_2 - r_2 d_2}{d_1} + r_2\right\} \\ \max\left\{r_1 + r_2,\ \dfrac{b_2}{d_2},\ \dfrac{b_1 + \max\{b_2 - r_2 d_2, 0\}}{d_1} + r_2\right\} \end{cases} \tag{12}$$

where the first expression holds when $\frac{b_2}{r_2} \geq \frac{b_1}{r_1}$ and the second otherwise.

Finally, similarly specializing the results of Propositions 8 and 9 to two flows, we find that the minimum required bandwidth $\widehat{R}^*$ under fifo without reshaping is

$$\widehat{R}^* = \max\left\{r_1 + r_2,\ \frac{b_1 + b_2}{d_2}\right\}; \tag{13}$$

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

$$\widehat{R}^*_s = \max\left\{r_1 + r_2, \frac{b_2}{d_2}, \frac{(b_1 + b_2)(r_2 + r_2)}{d_1 r_1 + d_2 r_2}, \frac{b_1 + b_2 - d_1 r_1 + \sqrt{(b_1 + b_2 - d_1 r_1)^2 + 4 r_1 d_2 b_2}}{2 d_2}\right\}. \tag{14}$$

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

1) The Impact of Scheduler Complexity: We first compare the bandwidth requirements of the three schedulers, *i.e.,* dynamic priority, static priority, and fifo where the latter two are combined with an optimal reshaper. The comparison is in the form of relative differences, *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.

From Eqs. (10) and (12), $R^* < \widetilde{R}^*_s$ iff $\frac{b_2}{r_2} < d_2 \leq d_1 < \frac{b_1}{r_1}$. Fig. 2a uses a "heatmap" to illustrate this difference for a 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 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[5].

To better characterize this range, we explore in [6] the supremum of $\frac{\widetilde{R}^*_s - R^*}{\widetilde{R}^*_s}$, and find that it 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 $R^* = r_1 + r_2$. This can be shown to yield a difference upper-bounded by 0.5. In other words, in the two-flow case, the (optimal) dynamic scheduler results in bandwidth savings of at most 50% over a static priority scheduler with (optimal) reshaping. This happens when the deadlines of the two flows are very close to each other. This is unlikely in practice.

Dynamic priority vs. fifo w/ optimal reshaping

Comparing $\widehat{R}^*_s$ with R^*, Eqs. (14) and (10) give that $\widehat{R}^*_s > R^*$ iff $d_1 - \frac{b_1}{r_1} < d_2 < \frac{b_1 + b_2 - d_1 r_1}{r_2}$. Fig. 2b illustrates the corresponding relative difference for the same previous two-flow combination. From the figure, fifo with shaping performs the most poorly relative to a dynamic priority scheduler when *both* d_1 and d_2 are small. This is again unlikely in practice.

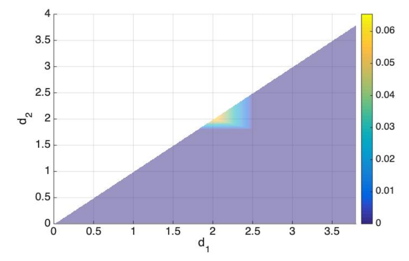
To explore the source and possible magnitude of this difference, we note that the supremum of $\frac{\widehat{R}^*_s - R^*}{\widehat{R}^*_s}$ is achieved when $0 < d_2 < \frac{b_1 + b_2 + r_2 d_1 - \sqrt{(b_1 + b_2 + r_2 d_1)^2 - 4 r_2 b_2 d_1}}{2 r_2}$, with Eq. (14) defaulting to $\widehat{R}^*_s = \frac{b_1 + b_2 - d_1 r_1 + \sqrt{(b_1 + b_2 - d_1 r_1)^2 + 4 r_1 d_2 b_2}}{2 d_2}$ and Eq. (10) to $R^* = \frac{b_2}{d_2}$. In [6] we show that the supremum of the relative difference $\frac{\widehat{R}^*_s - R^*}{\widehat{R}^*_s}$ is achieved as $d_1 \to 0$, 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 yield a 100% improvement over a basic fifo scheduler that reshapes flows optimally.

Fifo vs. static priority both w/ optimal reshaping

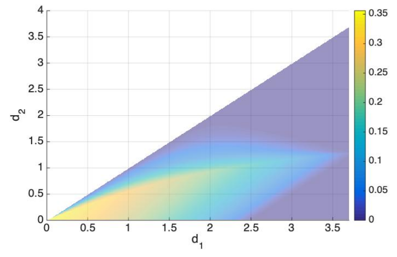
Comparing Eqs. (14) and (12) gives that $\widehat{R}^*_s > \widetilde{R}^*_s$ iff $\max\left\{\frac{b_2}{r_2}, \frac{(b_1 + b_2)(r_1 + r_2)}{r_2(b_1/d_1 + r_2)}\right\} < d_1 < \frac{b_1}{r_1}$. Fig. 2c illustrates their difference for the same two-flow combination as before.

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

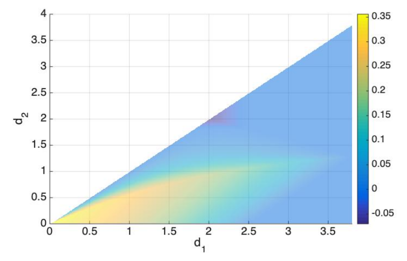
[5]When d_2 and d_1 are close but small, an edf dynamic priority scheduler behaves like a static priority one as the very large bandwidth called for by small deadlines ensures that data from either class is transmitted before dynamic priorities can affect transmissions order. Conversely, when d_2 and d_1 are close but large, both schedulers meet the deadlines with a bandwidth equal to the sum of the flows' average rates.



(a) Dynamic priority vs. static priority + reshaping



(b) Dynamic priority vs. fifo + reshaping



(c) fifo + reshaping vs. static priority + reshaping

Fig. 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).

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.

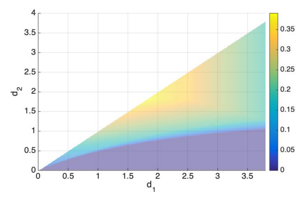
Fig. 2c also reveals that a small region exists ($d_1 \approx d_2$ with both of intermediate value) where fifo outperforms static priority. As alluded to in the discussion following Proposition 3, this is because a *strict* mapping of deadlines to priorities is not always optimal. For instance, two otherwise identical flows that differ infinitesimally in their deadlines should be treated "identically." Having the two flows share a common fifo queue is then a better fit than assigning them to two distinct priorities.

To better understand differences in performance between the two schemes, we characterize the supremum and the infimum of their relative difference. As again discussed in [6], the supremum is of the form $\frac{b_1}{b_1+b_2}$, which goes to 1 when $\frac{b_1}{b_2} \to \infty$, *i.e.,* a 100% penalty for fifo with reshaping over static priorities with reshaping. Conversely, the infimum is of the form $\frac{r_1}{r_1+r_2} - \frac{b_1}{b_1+b_2}$, 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 $\frac{b_1}{b_2} \to \infty$, the infimum is -1, *i.e.,* a maximum penalty of 100% but now for static priority with reshaping over fifo with reshaping.

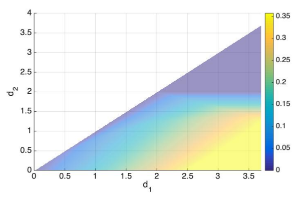
In other words, when used with reshaping, both fifo and static priority can end-up requiring twice as much bandwidth as the other. Addressing this issue calls for determining when flows should be grouped in the same priority class rather than assigned to separate classes. An optimal grouping can be identified in simple scenarios with two or three flows (see [6]), but a general solution remains elusive. However, as we shall see in Section VII-B, the simple strict priority assignment on which we rely appears to perform reasonably well across a broad range of flow configurations.

2) 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 evaluating the resulting relative differences in the minimum bandwidth required to meet flow deadlines under both schedulers without and with reshaping, *i.e.,* $\frac{\widetilde{R}^* - \widetilde{R}^*_s}{\widetilde{R}^*}$ and $\frac{\widehat{R}^* - \widehat{R}^*_s}{\widehat{R}^*}$.

For a static priority scheduler, Eqs. (11) and (12) 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.,* (re)shaping decreases the required bandwidth only when the larger deadline, d_1, is not too large and the smaller deadline, d_2, not too small. This is intuitive. When d_1 is large, the low-priority flow 1 can meet its deadline without any mitigation of the impact of flow 2. Conversely, a small d_2 offers little to no opportunity for reshaping flow 2, as the resulting added delay would need to be compensated by an even higher link bandwidth. This is illustrated in Fig. 3a for the same two-flow combination as in Fig. 2. The 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%.



(a) Static prio. w/o vs. w/ shaping



(b) fifo w/o vs. w/ shaping

Fig. 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).

Similarly, Eqs. (13) and (14) indicate that $\widehat{R}^*_s < \widehat{R}^*$ iff $d_2 < \frac{b_1+b_2}{r_1+r_2}$, *i.e.,* (re)shaping decreases the required bandwidth of a fifo scheduler 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. Fig. 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 with static priority, 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 appears that, in general, a fifo scheduler stands to benefit more from reshaping than a static priority one.

B. Relative Performance – Multiple Flows

This section extends the investigation of Section VII-A to scenarios with more than two flows, with flows assigned to *ten* different deadline classes. The dynamic range of deadlines is set to 10 with minimum and maximum deadlines of 0.1 and 1, respectively, and we consider three different patterns for how the 10 deadlines are spread across that range. Specifically,

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 assignments:

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 assignments:

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) $\boldsymbol{d}_{34} = (1, 0.97, 0.95, 0.93, 0.9, 0.88, 0.85, 0.82, \quad 0.6, \quad 0.1)$.

A total of $1,000$ experiments are then performed for each of the resulting eight deadline assignments, where an experiment consists of randomly selecting a traffic envelope for each deadline class[6]. Traffic envelopes are selected by independently drawing ten (aggregate) flow burst sizes b_1 to b_{10} 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 maps to a rate value beyond which even a fifo scheduler without reshaping performs as well as the optimal solution, so that there are no differences across mechanisms.

The results of the experiments are summarized in Table I, which gives the mean, standard deviation, and the 95% confidence interval of the mean 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.

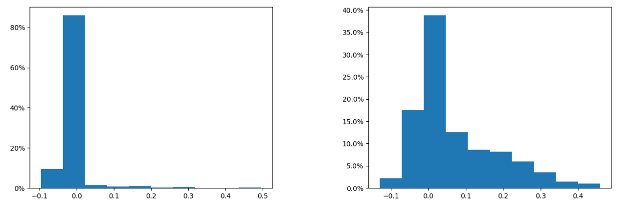
TABLE I: Relative bandwidth savings.

Comparisons	Scenario	Mean	Std. Dev.	95% Conf. Intvl.
R^* vs. $\widetilde{R}_s^*$	$\boldsymbol{d}_{11}$	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. $\widehat{R}_s^*$	$\boldsymbol{d}_{11}$	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. $\widehat{R}_s^*$	$\boldsymbol{d}_{11}$	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]

[6]In practice, this envelope corresponds to the aggregate of all flows assigned to the corresponding deadline class.

The first conclusion from the data in Table I is that while a dynamic priority scheduler affords some benefits, they are on average smaller than the maximum values of Section VII-A. In particular, average improvements over static priority with reshaping were often around 1% and did not exceed just over 6% across all configurations. Those values were a little higher for fifo with reshaping, where they reached 12%, but those are also much less than the maximum values of Section VII-A.

Table I 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 this is not consistent across configurations and a few scenarios exist where a fifo scheduler outperforms static priority. Fig. 4 illustrates this by plotting the relative "penalty" of fifo + reshaping over static priority + reshaping for two of the tri-modal deadline distributions, d_{32} and d_{33}.



(a) Tri-modal assignment $\boldsymbol{d}_{32}$ (b) Tri-modal assignment $\boldsymbol{d}_{33}$

Fig. 4: Relative bandwidth difference between fifo + shaping and static priority + shaping.

The two deadline assignments differ 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) of one deadline each. In contrast, assignment d_{33} has two small upper modes (large and intermediate) with a single deadline, and a large lower mode consisting of six relatively low value deadlines.

TABLE II: Relative benefits of reshaping flows.

Comparisons	Scenario	Mean	Std. Dev.	95% Conf. Intvl.
$\widetilde{R}_s^*$ vs. $\widetilde{R}^*$	$\boldsymbol{d}_{11}$	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. $\widehat{R}^*$	$\boldsymbol{d}_{1}$	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]

Recall that priorities map strictly to deadlines (smaller deadlines have higher priority). As a result, even if reshaping mitigates the impact of priority, multiple closely grouped deadlines is a poor fit for a priority scheme, especially when the number of other flows for which it can be beneficial

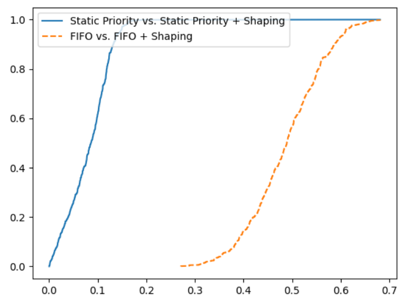


Fig. 5: CDF of relative bandwidth reduction from reshaping under bi-modal assignment $\boldsymbol{d}_{21}$.

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 from 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.

Towards gaining a better understanding of the extent reshaping may be behind the unexpected good performance of fifo, Table II reports its impact for both static priority and fifo. Specifically, as Table I, 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 II 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 $\approx 16\%$ 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. 5 through the full cumulative distribution function (cdf) of those benefits for both static priority and fifo under the deadline distribution $\boldsymbol{d}_{21}$ (bi-modal, with two similar modes of five deadlines at the two ends of the deadline range).

VIII. Conclusion and Future Work

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

The paper 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, the paper 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. 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.

There are many directions in which to extend the paper's results. The first is to tackle the issue illustrated in Section VII, and determine how to best group flows with different deadlines when relying on a static priority scheduler. Another, which we are currently pursuing, is to consider a multi-hop setting. This entails significant added complexity with exact solutions likely intractable. However, it represents an essential next step to making the results more broadly applicable.

References

[1] *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.

[2] A. Van Bemten and W. Kellerer, "Network calculus: A comprehensive guide," Technical University Munich, Technical Report 201603, October 2016.

[3] M. Howard, "Survey: SDN deployed by 78 percent of global service providers by the end of 2018," January 2019, downloaded on 4/01/21 from https://technology.ihs.com/610557/survey-sdn-deployed-by-78-percent-of-global-service-providers-by-the-end-of-2018.

[4] L. Luo, H. Yu, K. T. Foerster, M. Noormohammadpour, and S. Schmid, "Inter-datacenter bulk transfers: Trends and challenges," *IEEE Network*, vol. 34, no. 5, pp. 240–246, July 2020.

[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: ACM, August 2015.

[6] J. Song, R. Guérin, and H. Sariowan, "Minimizing network bandwidth under latency constraints: The single node case," 2021, available at http://arxiv.org/abs/2104.02222.

[7] T. Zhu, M. A. Kozuch, and M. Harchol-Balter, "WorkloadCompactor: Reducing datacenter cost while providing tail latency SLO guarantees," in *Proc. SoCC*, Santa Clara, CA, September 2017.

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

[9] 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*, vol. 26, no. 3, pp. 1193–1206, June 2018.

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

[11] L. Georgiadis, R. Guerin, and A. Parekh, "Optimal multiplexing on a single link: delay and buffer requirements," *IEEE Transactions on Information Theory*, vol. 43, no. 5, pp. 1518–1535, September 1997.

[12] 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*, vol. 1, no. 3, pp. 344–357, June 1993.

[13] 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*, Florianopolis, Brazil, August 2016.

[14] N. K. 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*, Santa Clara, CA, February 2020.