

On the Benefits of Traffic “Reprofiling” The Multiple Hops Case – Part I

Jiaming Qiu, *Member, IEEE*, Jiayi Song, Roch Guérin, *Fellow, ACM, IEEE*, and Henry Sariowan, *Member, IEEE*

Abstract—This paper considers networks where user traffic is regulated through deterministic *traffic profiles*, e.g., token buckets, and requires hard delay bounds. The network’s goal is to minimize the resources it needs to meet those bounds. The paper explores how *reprofiling*, i.e., proactively modifying how user traffic enters the network, can be of benefit. Reprofile produces “smoother” flows but introduces an up-front access delay that forces tighter network delays. The paper explores this trade-off and demonstrates that, unlike what holds in the single-hop case, reprofiling can be of benefit even when “optimal” schedulers are available at each hop.

Index Terms—latency, bandwidth, optimization, shaping, network calculus.

I. INTRODUCTION

Networks in domains as diverse as automotive, avionics, manufacturing, smart grids, and datacenters [1]–[7] serve users/applications with known traffic profiles and with expectations for delay guarantees, often in the form of hard end-to-end delay bounds. This has been reflected in standardization efforts such as the Time Sensitive Networking (TSN) and Deterministic Networking standards [8]–[11] that target hard delay bounds for deterministic traffic profiles under various schedulers. This is the environment this paper assumes.

Specifically, we consider networks where user flows are characterized using a *token bucket*, a common traffic regulator [12] that limits long-term transmission rates and flows’ burstiness, and where performance guarantees are in the form of deterministic end-to-end delay bounds. The problem we target is that of meeting those bounds while *using the least amount of overall network bandwidth* [1]. For ease of presentation, the results are derived under a *fluid* model. They hold with minor adjustments in a packet setting. Network topology and routing are assumed fixed.

The paper’s focuses on whether *reprofiling* flows, i.e., modifying on ingress the traffic profiles specified by users, can help reduce the bandwidth needed to meet delay bounds, inclusive of any reprofiling delay this introduces. The answer obviously

depends on the type of scheduler used in the network. The base one-hop case was investigated in [13] that established the benefits of reprofiling with First-In-First-Out (FIFO) and static priority schedulers, and confirmed the known result [14], [15] that reprofiling is unnecessary when an (optimal) Earliest Deadline-First (EDF) scheduler is used. This paper explores the extent to which this latter result remains true in a multi-hop network setting, with an extension considering the simpler FIFO and static priority schedulers left to a sequel (Part II).

We show that, in a multi-hop setting, reprofiling can help even with EDF-based schedulers. The intuition is that, while reprofiling has a cost (delay), it is incurred *once* (on ingress), but its benefits (from smoother traffic) accrue at *every hop* a flow traverses. Of note is the fact that the optimal solution often lies between the two extremes of “No Reprofile” (NR) and “Full Reprofile” (FR). NR keeps the traffic profile unchanged and allocates the entire delay budget to the network for maximum scheduling flexibility. Conversely, FR spends as much as possible of the delay budget on making flows smoother on ingress, at the cost of much tighter network delay budgets and, consequently, more limited scheduling flexibility.

The paper develops approaches for determining the best reprofiling solutions under a general service curve scheduler, SCED [16], and makes the following contributions:

- (i) Demonstrates the benefit of ingress reprofiling in meeting network delay bounds with less bandwidth;
- (ii) Develops and evaluates exact and approximate solutions for several scenarios.

Those contributions notwithstanding, the aspect of most interest is *establishing* that reprofiling can help even in the presence of powerful schedulers such as SCED.

The paper is structured as follows. Section II reviews network calculus and scheduling results we rely on. Section III presents our optimization framework. An exact but high complexity algorithm is introduced in Section IV, followed by a simpler yet effective approximate algorithm in Section V. The performance and run time of the two algorithms are compared in Section VI. Section VII illustrates the bandwidth benefits of reprofiling for several scenarios. Section VIII reviews related work. Finally, Section IX summarizes the paper’s contributions and discusses extensions. For clarity of presentation, proofs and ancillary results are relegated to appendices. For ease of reproducibility, both the reprofiling algorithms derived in the paper and the scenarios used to evaluate them are available at <https://github.com/qiujiaming315/traffic-reprofiling>.

J. Qiu and R. Guérin are with the Computer Science and Engineering department at Washington University in St. Louis, Saint Louis, MO 63130, USA, e-mail: {qiujiaming, guerin}@wustl.edu.

J. Song was with the Computer Science and Engineering department at Washington University in St. Louis and is now with ByteDance Inc., San Jose, CA 95110, USA, e-mail: jiayisong@wustl.edu.

H. Sariowan is with Google, Mountain View, CA 94043, USA, e-mail: hsariowan@google.com.

This work was supported by NSF grant CNS 2006530 and a gift from Google.

¹The formulation minimizes *total* bandwidth, but other objectives such as maximum or weighted sum of link bandwidth are possible. It can also be applied to the dual problem of maximizing the admitted traffic.

II. BACKGROUND

This section reviews relevant concepts from Network Calculus [17] (NC) and several results we rely on. Although the NC framework applies to packet and fluid models, for ease of exposition, we use a fluid model in the paper. The results can be extended to a packet model through standard approaches [17, Section 1.7], but do not yield further insight².

A. Network Calculus

1) *Arrival Curves*: They constrain the amount of traffic a flow is allowed to transmit over time. A flow with a cumulative arrival function $A(t)$ (i.e., the traffic sent between 0 and t) conforms to the arrival curve α if:

$$A(t + \tau) - A(t) \leq \alpha(\tau), \quad \forall \tau, t \geq 0.$$

In other words, α upper bounds the data sent in any time interval. The *token bucket* is such an arrival curve that bounds a flow's long-term rate r and burst size b :

$$\alpha(t) = \begin{cases} 0 & t = 0, \\ b + rt & t > 0. \end{cases}$$

We assume that, the traffic of every flow i is initially regulated by a token bucket $\alpha_i = (r_i, b_i)$ as it enters the network. We explore how modifying this original token bucket profile α_i can allow a network to offer given end-to-end latency guarantees to flows, and do so with less bandwidth.

2) *Service Curves*: They specify minimum service guarantees to flows, namely, if a flow with arrival function A is guaranteed a service curve β , then under relatively mild conditions³ that we assume hold, the amount of service $S(t)$ received by the flow by time t is such that there always exists a time $t - \tau, \tau \geq 0$ such that [17, Section 1.3]:

$$S(t) \geq A(t - \tau) + \beta(\tau).$$

3) *Delay and Buffer Bounds*: Given a flow with arrival and service curves α and β , its delay can be shown [17, Section 1.4] to be upper-bounded by the maximum horizontal distance $\Delta(\alpha, \beta)$ between α and β , where:

$$\Delta(\alpha, \beta) = \sup_{t \geq 0} \{ \inf_{\tau \geq 0 : \alpha(t) \leq \beta(t + \tau)} \tau \}. \quad (1)$$

Conversely, the maximum vertical distance $\Theta(\alpha, \beta)$ yields buffer bounds:

$$\Theta(\alpha, \beta) = \sup_{t \geq 0} \{ \alpha(t) - \beta(t) \}. \quad (2)$$

Another important result, known as *Pay Burst Only Once* (PBOO), that derives from those concepts is that the *end-to-end delay* of a flow traversing n hops and provided with service curve β_j at hop j can be upper-bounded using a *concatenated service curve* of the form [17, Section 3.1]:

$$\bigotimes_{j=1}^n \beta_j = \beta_1 \otimes \beta_2 \otimes \cdots \otimes \beta_n,$$

where \otimes is the min-plus convolution operator defined as:

$$(\beta_1 \otimes \beta_2)(t) = \inf_{0 \leq \tau \leq t} \{ \beta_1(\tau) + \beta_2(t - \tau) \}, \quad (3)$$

with $\Delta(\alpha, \bigotimes_{j=1}^n \beta_j)$ the end-to-end delay upper-bound.

Note that this upper-bound needs not be tight⁴. A service curve is a *lower bound* for the service a flow is guaranteed to receive, and its realization by the scheduler can deviate from it. This affects the flow's *departure curve* from the scheduler. In a multi-hop setting, this departure curve is the flow's arrival curve at the next hop and affects its delay there. Accurately characterizing departure curves is challenging for most schedulers, and hence so is tightening the upper bound.

4) *Traffic Shaping and Greedy Shaper*: A traffic shaper σ enforces conformance of incoming traffic with the arrival curve specified by σ . Shapers are typically realized as *greedy shapers* [17, Section 1.5.3] that release traffic at the earliest possible time that guarantees conformance with σ . Because the term shaper implies a non-work-conserving behavior, we opt to use the more general term “profiler” to account for the fact that, at least with sophisticated schedulers such as SCED, reprofiling can also be realized in a work-conserving manner⁵. We use the notation σ to indicate a flow's new (re)profile, with α denoting its original token bucket profile.

The goal of reprofiling is to make flows “smoother”, and a flow with initial token bucket profile α is reprofiled to $\sigma \leq \alpha$. Recall that a profiler/shaper enforces an arrival curve, so that a smaller profile imposes a corresponding reprofiling delay⁶. Given α and σ , this delay is given by $D = \Delta(\alpha, \sigma)$, where $\Delta(\cdot, \cdot)$ is as per Eq. (1). Ensuring that the flow's end-to-end deadline d is still met, inclusive of this reprofiling delay, therefore, calls for reducing the flow's network delay by a commensurate amount, i.e., from d down to $d - D$.

We note that, as articulated in [17, Corollary 1.5.1] and in the use of Eq. (1) to compute D , there is a close relation between greedy (re)profilers and service curves. The former upper bound the traffic a flow can transmit, while the latter give a lower bound on the service (transmission opportunities) the flow can receive. Hence, when, at a given hop, a flow is assigned a service curve equal to its profile, the flow experiences a delay of 0 (under a fluid model) at this hop.

In this paper, we explore whether making flows smoother outweighs the tighter network delays this forces. We also note that once a flow has been reprofiled according to σ , the same reprofiling can be applied at every hop without causing additional delays [17, Section 1.5.3]. This ensures that the benefits of reprofiling remain present at every hop.

5) *SCED Scheduling Policy*: Service Curve Earliest Deadline first (SCED) [16] is a scheduling policy that extends EDF and that, when schedulable (see Lemma 1), provides service curve guarantees to flows. SCED operates as follows: Given m flows with flow $i = 1, \dots, m$, having arrival curve $A_i(t)$ and service curve $\beta_i(t)$, a bit arriving at time t is assigned

⁴See [18, Section 1.2.3] for a pertinent discussion of this issue.

⁵Under SCED, the (re)profiling “delay” of a work-conserving profiler is simply integrated in the deadline SCED uses for the flow's transmissions.

⁶Under work-conserving operation, this delay is not necessarily incurred.

²See Appendix B-A for a detailed discussion.

³ β is continuous, $\beta(0) = 0$, and A is left-continuous.

deadline t' , where t' is the latest possible time that meets the flow's service curve guarantee and is computed as follows:

$$A_i(t) = (A_i \otimes \beta_i)(t').$$

Bits are scheduled for transmission in order of their deadlines.

Given a set of service curves, the link bandwidth C^* that ensures SCED's schedulability is characterized in Lemma 1.

Lemma 1. *Given a set of service curves $\beta_i(t), i = 1, \dots, m$, any scheduling mechanism requires a link bandwidth of at least:*

$$C^* = \sup_{t \geq 0} \frac{\sum_i \beta_i(t)}{t},$$

to guarantee those service curves. SCED realizes those service curves with a link bandwidth of exactly C^* .

The proof is in Appendix B-B. We note that $\sum_i \beta_i(t)$ represents an *aggregate service curve* that captures the minimum amount of service SCED needs to provide to guarantee service curves $\beta_i(t)$ to all flows $i = 1, \dots, m$.

The flexibility of EDF schedulers (and therefore SCED) notwithstanding, they are complex even if efficient implementations are increasingly available [19], [20]. However, they are optimal when it comes to minimizing bandwidth on individual links [13]–[15], and therefore represent a baseline when evaluating the potential benefits of reprofiling. To that end, we assume the availability of SCED schedulers capable of allocating dedicated service curves to flows at every hop.

B. Two-Slope Rate-Latency Service Curve

Although SCED can accommodate arbitrary service curves, for tractability we limit our investigation to a subset of piecewise-linear service curves we term *two-slope rate-latency service curves* (2SRLSC). Fig. 1a illustrates the generic structure

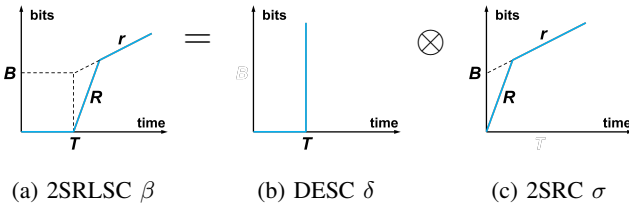


Fig. 1: 2SRLSC and its two components.

of the 2SRLSC $\beta = (T, R, B, r)$ allocated to a flow at a given hop. It consists of three segments and can be decomposed into the concatenation of a Delay Element Service Curve (DESC) δ , and a Two-Slope Reprofile Curve (2SRC) σ , i.e., $\beta = \delta \otimes \sigma$. There are two primary motivations for our choice.

- (i) The use of separate latency and rate components (DESC & 2SRC) acknowledges the efficacy of decoupling delay and rate guarantees, as embodied in the rate-latency service curves behind most practical schedulers [17, p. 21];
- (ii) The use of a 2SRC stems from our reliance on reprofiling. As each flow is reprofiled, assigning it a 2SRC equal to its profile ensures that its delay at each hop is just the delay T of the delay element. Further, as shown in Section II-B2, restricting reprofiling to only two slopes is not restrictive and can minimize the required bandwidth.

1) *Delay Element Service Curve*: A DESC δ is shown in Fig. 1b. It ensures that arriving traffic is transmitted before a *deadline* T . Operationally, it maps to an EDF scheduler (assuming schedulability), and enforces *local deadlines* for flows at every hop.

2) *Two-Slope Reprofile Curve*: A 2SRC $\sigma = (R, B, r)$, is defined by three parameters: a short-term rate $R \geq r$, a parameter B that determines the duration $t = B/(R - r)$ of transmission at R , and a long-term rate r . We note that σ can be realized through the concatenation of two token buckets $\alpha_1^* = (R, 0)$ and $\alpha_2^*(r, B)$, i.e., $\sigma = \alpha_1^* \otimes \alpha_2^*$. The next lemma formalizes the extent to which this is not a limitation.

Lemma 2. *Consider a flow with token bucket profile $\alpha = (r, b)$ that is reprofiled using $\sigma = \bigotimes_{j=1}^n \tilde{\alpha}_j$, where $\tilde{\alpha}_j = (r_j, b_j), j = 1, \dots, n$, are two-parameters token buckets, and σ is such that $\Delta(\alpha, \sigma) = D$. Let $\sigma^* = \alpha_1^* \otimes \alpha_2^*$, where α_1^* and α_2^* are token buckets with profiles $\alpha_1^* = (R^* = b/D, 0)$ and $\alpha_2^* = (r, B^* = b - rD)$, then $\Delta(\alpha, \sigma^*) = D$ and $\sigma(t) \geq \sigma^*(t), \forall t \geq 0$.*

The proof is in Appendix B-C, and the lemma establishes that, among all concatenations of token buckets that realize the (reprofiling) delay target D , σ^* does so while being the most frugal when it comes to bandwidth, i.e., $\sigma^*(t) \leq \sigma(t)$.

Combining Lemma 2 with Lemma 1 establishes that, when using SCED to realize a delay target of D for an arrival curve α , the 2SRC σ^* requires the least amount of bandwidth among all concave, piece-wise-linear service curves, i.e., service curves realized by a concatenation of token buckets and an EDF scheduler. Hence, σ^* is *optimal* among this family of service curves. Fig. 2 illustrates how σ^* realizes the (reprofiling) delay D for an arrival curve α .

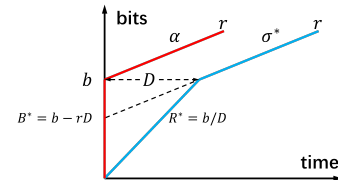


Fig. 2: Optimal reprofiler given reprofiling delay.

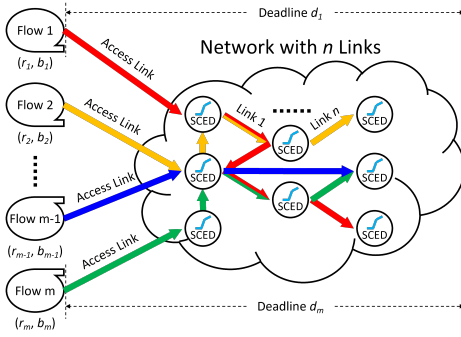
III. PROBLEM FORMULATION

A. Problem Setting

Consider a network with n links identified by their index $j, 1 \leq j \leq n$, and carrying m flows, as illustrated in Fig. 3. Flow $i, 1 \leq i \leq m$, is characterized as follows:

- The path of flow i from source to destination is assumed acyclic and specified by an ordered set of distinct link indices $\mathcal{P}_i = \{j_{i1}, j_{i2}, \dots, j_{ik}\}$ for a k hops path. The directed graph resulting from the union of the flow paths is allowed a general structure, i.e., either acyclic or cyclic.
- Traffic from flow i conforms to token bucket $\alpha_i = (r_i, b_i)$, and has an end-to-end packet-level latency target d_i . Together, α_i and d_i define the flow's profile (r_i, b_i, d_i) .

⁷Exclusive of propagation and processing delays.

Fig. 3: Network with m flows and n links.

Flows connect to the network through dedicated, high-speed access links and at each hop are assigned a 2SRLSC as defined in Section II-B. Further, network buffers are assumed large enough to ensure lossless operation. Buffer bounds can be readily derived from basic network calculus, and are presented in Section IV-C for a non-work-conserving setting.

B. Optimization Framework

1) *Inputs*: They consist of the vector triplet $(\mathbf{r}, \mathbf{b}, \mathbf{d})$ and the path matrix $\mathcal{P} = (\mathcal{P}_1, \mathcal{P}_2, \dots, \mathcal{P}_m)$, where $\mathbf{r} = (r_1, r_2, \dots, r_m)$, $\mathbf{b} = (b_1, b_2, \dots, b_m)$, and $\mathbf{d} = (d_1, d_2, \dots, d_m)$ specify the flows' token rates, burst sizes, and deadlines, respectively.

2) *Variables*: Given our reliance on SCED, link bandwidths depend on the 2SRLSCs assigned to the flows sharing a link, with their parameters the variables that capture our bandwidth minimization goal. Given a link j and denoting as \mathcal{F}_j the set of flows whose path traverses link j , the 2SRLSC β_{ij} of a flow in \mathcal{F}_j is specified through four parameters $(T_{ij}, R_{ij}, B_{ij}, r_i)$, where we have taken advantage of the fact that flows are regulated with a token rate that can, therefore, be assigned as the long-term rate of their 2SRLSCs. Hence, in addition to satisfying the stability condition $C_j \geq \sum_{i \in \mathcal{F}_j} r_i$, the bandwidth C_j needed on link j to accommodate the 2SRLSCs of the flows in \mathcal{F}_j depends only on the variables (T_{ij}, R_{ij}, B_{ij}) .

In general, the 2SRLSCs of flow i are captured in three vectors \mathbf{T}_i , \mathbf{R}_i , and \mathbf{B}_i that specify the variables (T_{ij}, R_{ij}, B_{ij}) for the links of path \mathcal{P}_i . The vectors for all m flows are then the variables⁸ that determine the vector of link bandwidths $\mathbf{C} = (C_1, C_2, \dots, C_n)$ the flows' 2SRLSCs require.

3) *Constraints*: The end-to-end latency bounds are the constraints of our optimization, which, recalling the results of Section II-A2, can be expressed as:

$$\Delta(\alpha_i, \bigotimes_{j \in \mathcal{P}_i} \beta_{ij}) \leq d_i, \quad \forall 1 \leq i \leq m, \quad (4)$$

In other words, for all flows, the sum over all hops of their reprofiling and scheduling delays (ingress reprofiling is assumed done at the first hop) is no larger than their end-to-end latency target.

⁸Section IV-A shows how the number of variables can be reduced.

4) *Objective Function*: Our goal is to meet latency targets while minimizing *total bandwidth* (again, other wide-sense increasing functions such as maximum link bandwidth or weighted sum of link bandwidth can be used). Formally, the optimization objective function (**OPT**) is:

$$\mathbf{OPT} : \min_{\mathbf{R}_i, \mathbf{T}_i, \mathbf{B}_i} \sum_{j=1}^n C_j, \quad (5)$$

subject to the constraints of Eq. (4).

C. Discussion

We note that **OPT** relies on the upper-bounds of Section II-A2 through Eq. (4). This has implications when it comes to attributing causes to the bandwidth improvements its solutions afford. Specifically, improvements come from three possible sources: (i) reprofiling; (ii) the assignments of distinct per-hop deadlines; **and** (iii) the tighter delay bounds that are themselves a consequence of reprofiling.

Our focus is on reprofiling, but local deadline assignments can also reduce bandwidth by leveraging heterogeneity across hops. Further, making flows smoother, as reprofiling does, reduces the possible “gap” between arrival and service curves. This allows tighter end-to-end bounds (see Section II-A3) as it restricts deviations between departure curves and service curves⁹. As a result, accurately quantifying how much of the improvements we observe come from reprofiling alone is challenging. Nevertheless, the results we present in Section VII provide evidence of the benefits of reprofiling, even if precisely assessing their relative contribution remains elusive.

IV. SOLVING PROBLEM **OPT**

The main challenge in solving **OPT** comes from the large number of variables associated with having individual 2SRLSCs for each flow at each hop. Next, We introduce steps to decrease this complexity by reducing the number of variables involved while preserving the solution's optimality.

A. *OPT* Reduction

We first establish that, for each flow, it is enough to consider the same 2SRC at each hop. We term this 2SRC the flow's *minimum reprofiler*, and show that for any optimal solution of **OPT**, it can be specified through a single parameter, the flow's *reprofiling delay*.

1) *Minimum Reprofiler*: This first lemma is a direct consequence of applying NC's concatenation result [17, Section 1.4.3] to 2SRLSCs, and will allow us (using Lemma 1) to reduce the number of variables of **OPT** by assuming that a flow is assigned the same 2SRC at every hop.

Lemma 3. Consider flow i assigned token bucket (r_i, b_i) and assigned on link $j \in \mathcal{P}_i$ a 2SRLSC β_{ij} with parameters

⁹Note that under the *linear* service curve of full reprofiling, departure and service curves are identical and the bound is tight.

T_{ij} , R_{ij} , B_{ij} , and $r_{ij} = r_i$. The concatenation of the flow's 2SRLSCs is readily found to be another 2SRLSC of the form:

$$\bigotimes_{j \in \mathcal{P}_i} \beta_{ij} = \beta \left\{ T_i = \sum_{j \in \mathcal{P}_i} T_{ij}, R_i = \min_{j \in \mathcal{P}_i} R_{ij}, B_i = \min_{j \in \mathcal{P}_i} B_{ij}, r_i \right\} \\ = \delta_{T_i} \otimes \sigma_i,$$

where δ_{T_i} is a delay element with parameter T_i and σ_i is a 2SRC with parameters R_i, B_i , and r_i .

The proof is in Appendix B-D. We term σ_i the flow's *minimum reprofiler*, and from Section II-A3, the flow's end-to-end delay upper bound is then of the form:

$$\Delta(\alpha_i, \bigotimes_{j \in \mathcal{P}_i} \beta_{ij}) = \sum_{j \in \mathcal{P}_i} T_{ij} + \Delta(\alpha_i, \sigma_i), \quad \forall 1 \leq i \leq m, \quad (6)$$

which is the sum of all the *local deadlines* plus the *reprofiling delay* from the minimum reprofiler σ_i ¹⁰.

The next lemma formalizes that assigning flow i the same 2SRC as part of its 2SRLSC at each hop does not affect the optimal solution of **OPT**.

Lemma 4. *Given an optimal solution to **OPT**, the 2SRC of the 2SRLSC of every flow $i, 1 \leq i \leq m$, on link $j \in \mathcal{P}_i$, can be set to its minimum reprofiler σ_i .*

This is consistent with results from [17, Section 1.5.3]. The proof derives directly from Eq. (6), which states that given flow i 's local deadline assignments \mathbf{T}_i , ensuring that its end-to-end latency target d_i is met depends only on the values R_i and B_i of its minimum reprofiler. Further, as per Lemma 1, the link bandwidth required to accommodate a set of 2SRLSCs is a non-decreasing function of the R_{ij} and B_{ij} of the flows sharing the link. Given an optimal solution to **OPT**, changing the parameters R_{ij} and B_{ij} of flow i on link j to R_i and B_i (their minima) can, therefore, never increase the required link bandwidth. Hence an optimal solution exists where all flows are assigned their minimum reprofiler.

2) *Reprofiling Delay*: We next establish that for any given flow $i, 1 \leq i \leq m$, the variables R_i and B_i can be reduced to a single variable, namely, the flow's *reprofiling delay* D_i , using the optimal reprofiler σ_i^* introduced in Lemma 2.

Lemma 5. *Given an optimal solution to **OPT**, the minimum reprofiler σ_i of every flow $i, 1 \leq i \leq m$, can be set to its optimal reprofiler σ_i^* .*

According to Lemma 2, the optimal reprofiler σ_i^* ensures $\sigma_i(t) \geq \sigma_i^*(t), \forall t \geq 0$ among all reprofilers σ_i with a reprofiling delay of D_i . Hence, setting $R_i = b_i/D_i$ and $B_i = b_i - r_i D_i$ for all flows does not affect the optimality of the solution. Consequently, we have:

Proposition 6. *Solving **OPT** is equivalent to solving **OPT**⁻*

$$\mathbf{OPT}^- : \min_{\substack{\mathbf{T}_i, D_i \\ 1 \leq i \leq m}} \sum_{j=1}^n C_j, \quad (7)$$

subject to the constraints

$$\sum_{j \in \mathcal{P}_i} T_{ij} + D_i \leq d_i, \quad \forall 1 \leq i \leq m, \\ D_i \leq \frac{b_i}{r_i}, \quad \forall 1 \leq i \leq m, \quad (8)$$

where the second set of constraints ensure $R_i \geq r_i$ for all minimum reprofilers.

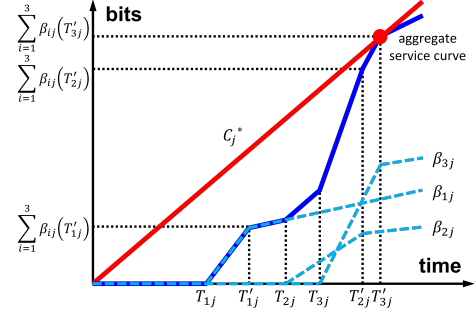


Fig. 4: Aggregate service curve and minimum required bandwidth.

B. Non-Linear Programs Formulation

Eq. (7) shows that solving **OPT**⁻ calls for computing the link bandwidths $C_j, 1 \leq j \leq n$, given the variables $\mathbf{T}_i, D_i, 1 \leq i \leq m$. Fig. 4 illustrates for a given link j with three flows, the minimum bandwidth C_j^* that satisfies the constraints imposed by the service curves β_{ij} 's of the flows traversing link j . The figure also hints at how C_j^* can be determined¹¹ from the resulting aggregate service curve (the solid, dark-blue line in the figure). This aggregate service curve is a composite piece-wise linear function constructed from the three segments of all the individual β_{ij} 's, namely,

$$\beta_{ij}(t) = \begin{cases} 0 & 0 \leq t < T_{ij}, \\ \frac{b_i}{D_i}(t - T_{ij}) & T_{ij} \leq t < T'_{ij}, \\ b_i + r_i(t - T'_{ij}) & t \geq T'_{ij}, \end{cases} \quad (9)$$

with the notation $T'_{ij} = T_{ij} + D_i$ used in Eq. (9) and Fig. 4.

Combining Lemma 1 with the expressions in Eq. (9) gives $C_j^* = \sup_{t \geq 0} \frac{\sum_{i \in \mathcal{F}_j} \beta_{ij}(t)}{t}$, which, as per the next lemma, can be further simplified into Eq. (10).

Lemma 7. *Given a set of flows \mathcal{F}_j sharing link j and assigned service curves in the form of 2SRLSCs as given in Eq. (9), the minimum bandwidth C_j^* required to meet the service curves is equal to:*

$$C_j^* = \max_{k \in \mathcal{F}_j} \left\{ \sum_{i \in \mathcal{F}_j} r_i, \frac{\sum_{i \in \mathcal{F}_j} \beta_{ij}(T'_{kj})}{T'_{kj}} \right\}. \quad (10)$$

Eq. (10) derives directly from the proof of Proposition 1 in [13]. Solving **OPT**⁻, therefore, conceptually calls for using Eqs. (10) and (9) to evaluate $\sum_{j=1}^n C_j$ for all combinations of

¹¹Note that C_j^* can only be realized at points associated with bandwidth decreases, and those correspond to the $T'_{ij} = T_{ij} + D_i$'s.

¹⁰This is consistent with an earlier similar finding in [21].

the variables T_{ij} and D_i that meet the constraints of Eq. (8). As alluded to, Fig. 4 suggests a more efficient approach.

From Eq. (10), we see that C_j^* has a closed-form functional expression *as long as the set of linear segments making up the aggregate service curve remains unchanged* (condition **ORD**). In other words, their slope and length can vary, but their number and respective positions should not. When this holds and given the functional expression of this aggregate service curve on each link, **OPT**[−] can then be formulated as a set of non-linear programs¹² (NLPs).

Those NLPs can be solved using standard solvers. Finalizing a solution to **OPT**[−] then calls for exploring how the problem's variables and constraints affect condition **ORD**. Specifically, **ORD** depends on the relative positions (*ordering*, hence the condition's name) of the inflection points of the link's aggregate service curve. For link j , these map to the variables T_{ij} and $T'_{ij} = T_{ij} + D_i, i \in \mathcal{F}_j$.

The number of feasible orderings of these variables is combinatorial in nature (it grows super-exponentially in m and n), and likely intractable even for a single hop. As a result, we rely on a standard randomized combinatorial search strategy to explore the space of feasible orderings¹³, and for each such ordering solve the associated NLP. Appendix B-E offers additional details on how we generate feasible orderings, and illustrates it with a simple three flows example.

C. Buffer Bounds

Given a solution to **OPT**[−], Eq. (2) can be used to obtain buffer upper bounds. We do so for a non-work-conserving setting¹⁴, where buffer requirements are split into (i) per flow ingress reprofiling buffers, (ii) per flow reprofiling buffers at each hop/link, and (iii) a per link scheduling buffer.

Under non-work-conserving operation, a flow's reprofiling buffer, at both ingress and at each hop, receives data from the upstream link or source, stores it until it conforms to the flow's (re)profile, before releasing it to the link/scheduler for transmission. The (link) scheduling buffer holds data from all flows waiting for transmission on the link. Bounds can be obtained separately for each flow's reprofiling buffer and for the link scheduling buffer. With those bounds in hand, an upper bound can be derived for a link's total buffer requirements simply by adding the reprofiling buffer bounds of the flows sharing the link and the link's scheduling buffer bound. Expressions for those individual bounds are given next.

The scheduling buffer bound for link j is given by:

$$\hat{\Theta}_j = \sup_{t \geq 0} \left\{ \sum_{i \in \mathcal{F}_j} \sigma_i - C_j t \right\}. \quad (11)$$

where $\sum_{i \in \mathcal{F}_j} \sigma_i$ represents the aggregate arrival curve of all flows sharing link j as they leave their respective reprofiler,

¹²The variables D_i 's and $T_{ij} + D_i$'s are in the denominator of Eqs. (9) and (10), respectively.

¹³We explore a logarithmic number of feasible orderings to achieve a reasonable trade-off between solution quality and computational efficiency.

¹⁴In a work-conserving setting buffer bounds are more challenging as tight departure curves from the upstream SCED schedulers are difficult to derive.

i.e., the flow's 2SRC. Conversely, $C_j t$ represents the link's service curve given its bandwidth C_j .

Similarly, we show in Appendix B-F that the reprofiling buffer of flow i at hop (link) j can be bounded based on the worst-case burst that can accumulate because of scheduling delays at the previous hop j' on its path \mathcal{P}_i :

$$\tilde{\Theta}_{ij} = \sup_{t \geq 0} \{ \sigma_i - \beta_{ij'} t \}, \quad (12)$$

Finally, we note that flow i 's ingress reprofiling buffer is readily bounded by its original token bucket burst size b_i .

D. A Representative Example

To showcase the operation of our NLP approach and the type of solutions to **OPT**[−] it produces, we introduce next a few representative examples. In spite of their limited scope, they highlight the diversity of possible outcomes, and consequently the challenges in generating insight into how individual parameters affect solutions.

For clarity of exposition, we focus on the minimalist configuration of Fig. 5 with just 2 hops and 2 flows. Flow 1 traverses links 1 and 2, while flow 2 is limited to link 2. The optimal solution, therefore, only requires determining how to reprofile flow 1 and split its residual deadline (after subtracting its reprofiling delay) across the two links. This is because flow 1 is reprofiled prior to reaching link 2, which, when combined with the optimality of EDF in the one-hop scenario [13], implies that minimizing the bandwidth of link 2 can be realized without reprofiling flow 2 (or further reprofiling flow 1).

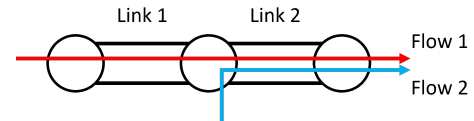


Fig. 5: Network with 2 hops and 2 flows.

Table I reports the structure (reprofiling delay and local deadline assignments) of optimal solutions for 5 experiments associated with different combinations of flow profiles. Because the focus is on the *structure* of the solution, we do not report the resulting bandwidth values, nor do we compare them to that of alternative approaches. This discussion is deferred to the investigation of Section VI.

Returning to Table I, the columns labeled “flow 1” and “flow 2” report the (r_i, b_i, d_i) , $i = 1, 2$, profiles of the two flows, with rates, burst sizes, and deadlines taking different values in each of the 5 experiments. The second group of columns labeled “Flow 1 Reprofiling and Deadlines” reports the reprofiling delay D_1 of flow 1 and its local deadlines T_{11}, T_{12} , on links 1 and 2, respectively. We also report D_1 relative to its maximum value¹⁵ $\hat{d}_1 = \min(d_1, b_1/r_1)$ to demonstrate how much reprofiling is applied to flow 1.

As alluded to, no clear trends emerge from the data, *e.g.*, tighter deadlines imply neither smaller nor larger relative reprofiling delays. Nevertheless, the optimal solutions share a few characteristics: (1) Optimal reprofiling delays can be

¹⁵The reprofiling rate R_1 cannot be less than r_1 .

TABLE I: Optimal Solutions for the 2-hop, 2-flow Network of Fig. 5

	Profile $(r_i, b_i, d_i), i = 1, 2$		Flow 1 Reprofilling and Deadlines			
	flow 1	flow 2	D_1	T_{11}	T_{12}	D_1/\hat{d}_1
Expt 1	(98.75, 88.18, 0.20)	(87.63, 33.56, 0.01)	0.10	0.09	0.01	50.68%
Expt 2	(16.84, 21.88, 2.00)	(57.37, 70.14, 1.00)	1.27	0	0.73	98.07%
Expt 3	(28.26, 71.05, 2.00)	(81.47, 48.07, 0.10)	0.93	0.97	0.10	46.29%
Expt 4	(60.39, 4.88, 0.20)	(86.24, 61.55, 0.10)	0.05	0.04	0.11	57.15%
Expt 5	(33.11, 6.19, 0.20)	(25.32, 88.41, 0.01)	0.08	0.11	0.01	44.56%

anywhere between full reprofilling ($D_i = \hat{d}_i$) and no reprofilling ($D_i = 0$), and (2) Local deadlines can vary significantly across hops, *i.e.*, they are typically not equal.

V. A GREEDY REPROFILING ALGORITHM

The NLP formulation's ability to find an "optimal" solution to **OPT**⁺ notwithstanding, its complexity motivates the development of a heuristic that we describe next. It relies on a simple greedy algorithm (Greedy) that, at least on the small topologies the NLP formulation can handle, performs nearly as well, but at a much lower computational cost.

As just mentioned, optimal solutions often have reprofilling delays D_i anywhere in $[0, \hat{d}_i]$, where $\hat{d}_i = \min(d_i, b_i/r_i)$ is flow i 's maximum reprofilling delay, and per hop deadlines that vary across hops. In other words, those solutions select *intermediate* reprofilling configurations, *i.e.*, strictly between "no reprofilling" and "full reprofilling", and distribute the remaining delay budget unevenly across hops.

A. Overview

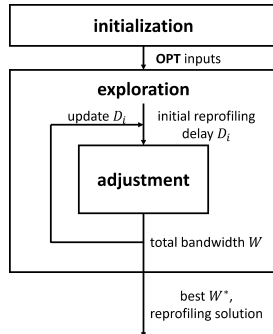


Fig. 6: Overview of Greedy.

The algorithm we present next captures both above aspects. It proceeds in two nested phases as shown in Fig. 6.

- (i) An iterative *exploration* of reprofilling configurations, with initial flows' reprofilling delays spanning values between 0 and their maximum value \hat{d}_i .
- (ii) For each reprofilling configuration, *adjustments* of local deadlines and reprofilling solutions at hops where flows' delay guarantees are strictly better than their target.

B. Exploration Phase

As its name indicates, this phase *explores* reprofilling configurations that span the range of possible reprofilling delays. The intent is to find the combination that yields the smallest overall

Algorithm 1 Exploration

Inputs: flow profiles $\mathbf{r} = (r_1, r_2, \dots, r_m)$,
 $\mathbf{b} = (b_1, b_2, \dots, b_m)$, $\mathbf{d} = (d_1, d_2, \dots, d_m)$
 path matrix $\mathbf{P} = (\mathcal{P}_1, \mathcal{P}_2, \dots, \mathcal{P}_m)$
 exploration parameters L, K
 adjustment threshold ϵ

Output: minimum total bandwidth W^*
 optimal reprofilling delays \mathbf{D}^*
 local deadline assignments \mathbf{T}^*

- 1: $\hat{d}_i = \min(d_i, b_i/r_i), \forall 1 \leq i \leq m$
- 2: $lr = 0, hr = 1$
- 3: **for** $l = 1$ to L **do**
- 4: **for** $k = 0$ to $K + 1$ **do**
- 5: $\gamma_k = lr + \frac{hr - lr}{K + 1} \cdot k$
- 6: $D_i = \gamma_k \cdot \hat{d}_i, \forall 1 \leq i \leq m$
- 7: $T_{ij} = (d_i - D_i)/|\mathcal{P}_i|, \forall 1 \leq i \leq m, j \in \mathcal{P}_i$
- 8: $W = \text{adjustment}((\mathbf{r}, \mathbf{b}, \mathbf{d}), \mathbf{P}, \mathbf{D}, \mathbf{T}, \epsilon)$
- 9: **if** $W < W^*$ **then**
- 10: $W^* = W, k^* = k$
- 11: $\mathbf{D}^* = \mathbf{D}, \mathbf{T}^* = \mathbf{T}$
- 12: **end if**
- 13: **end for**
- 14: $lr = \gamma_{\max(k^* - 1, 0)}, hr = \gamma_{\min(k^* + 1, K + 1)}$
- 15: **end for**
- 16: **return** $W^*, \mathbf{D}^*, \mathbf{T}^*$

bandwidth. It is detailed in Algorithm 1 where we refer to the line numbers on the left when describing the algorithm.

In considering possible reprofilling configurations, the choice of reprofilling delays can vary across flows, and each flow i boasts a different range $[0, \hat{d}_i]$. Exploring all possible combinations is clearly intractable. Our exploration seeks a trade-off between coverage and tractability. It proceeds iteratively, and, in iteration x , applies the *same* reprofilling ratio γ_x to all flows, and sets the reprofilling delay of flow i to $D_i(x) = \gamma_x \hat{d}_i$. This allows flows with different reprofilling ranges to be assigned different reprofilling delays, while enabling a systematic exploration of the underlying space, *i.e.*, from $D_i(x) = 0$ when $\gamma_x = 0$ to $D_i(x) = \hat{d}_i$ when $\gamma_x = 1$.

The relative simplicity of this approach notwithstanding, its coverage depends on the number of distinct values γ_x takes; with a fine-grain exploration requiring a large number of values. To bound the resulting computational cost, we rely on an iterative approach that progressively refines the range of reprofilling ratios it explores. It is illustrated in Fig. 7.

Turning next to the details of Algorithm 1, each iteration explores $K + 2$ reprofilling ratios $\gamma_k = \frac{k}{K + 1}, 0 \leq k \leq K + 1$,

uniformly distributed in that iteration's range (line 4). The first iteration (line 2) is shown in Fig. 7a and spans the full range from $D_i(0) = 0$ to $D_i(K+1) = d_i$. Subsequent iterations, for a total of L , explore increasingly narrower ranges of reprofiling ratios to progressively refine and improve the best result.

Specifically, Algorithm 1 iterates (line 3) by exploring new ranges centered at the reprofiling ratio that produced the smallest network bandwidth in the previous iteration (lines 9–10), and with lower and upper bounds set to the two adjacent values (line 13). As discussed next, given a reprofiling ratio, the required network bandwidth is obtained (line 8) following an adjustment phase¹⁶ that, when feasible, shifts delay allocations between initial reprofiling delays and local deadlines.

We note that implicit in the iterative approach of Algorithm 1 is the assumption that, in any iteration, the region where the minimum of the objective function (network bandwidth) is located can be bracketed with few samples. Fig. 27 of Appendix B-G offers evidence in support of this assumption.

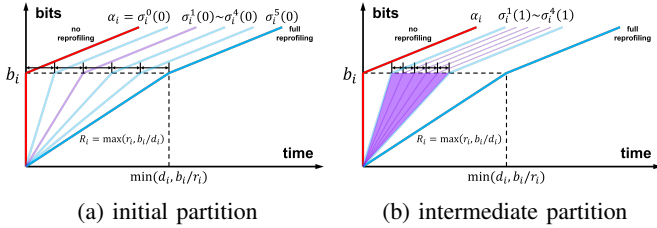


Fig. 7: Reprofile exploration for $K = 4$. $\sigma_i^k(l)$ specifies the k^{th} 2SRC of flow i at iteration l . $\sigma_i^2(0)$ is assumed to yield the best outcome in the first iteration.

C. Adjustment Phase

As shown in Fig. 6, the adjustment phase is invoked in each iteration of the exploration phase. For an initial assignment of reprofiling delays and local deadlines, it checks every link to, when possible, greedily *adjust* (increase) the reprofiling delays of flows with better deadlines than needed on that link. The adjustment is bandwidth-neutral for the link, but, because increasing reprofiling makes flows smoother, it can reduce bandwidth requirements on other links. Algorithm 2 details those adjustments, with Fig. 8 illustrating them on link j .

Recall that according to Eq. (10), the minimum required bandwidth C_j^* on link j depends on the values of the aggregate service curve at the inflection points T_{ij}' 's. As illustrated in Fig. 8a, only one of those inflection points is typically involved in determining C_j^* . This indicates the presence of “slack,” *i.e.*, more bandwidth than necessary, at the other inflection points. We denote the slack at inflection point T_{ij}' as s_{ij} , with:

$$s_{ij} = C_j^* T_{ij}' - \sum_{k \in \mathcal{F}_j} \beta_{kj}(T_{ij}'). \quad (13)$$

The purpose of the adjustment phase is to leverage this slack to increase the reprofiling delay of flows that can tolerate it. For a given flow i , this means increasing its reprofiling delay while correspondingly decreasing T_{ij}' (to keep T_{ij}' constant). The

Algorithm 2 Adjustment

Input: flow profiles $\mathbf{r} = (r_1, r_2, \dots, r_m)$,
 $\mathbf{b} = (b_1, b_2, \dots, b_m)$, $\mathbf{d} = (d_1, d_2, \dots, d_m)$
path matrix $\mathcal{P} = (\mathcal{P}_1, \mathcal{P}_2, \dots, \mathcal{P}_m)$
initial reprofiling delays $\mathbf{D} = (D_1, D_2, \dots, D_m)$
initial local deadlines $\mathbf{T} = \{T_{ij} : \forall 1 \leq i \leq m, j \in \mathcal{P}_i\}$
improvement threshold ϵ

Output: total bandwidth W after adjustment

- 1: $W = \infty, W' = \infty$
- 2: sort links in decreasing order of $|\bigcup_{i \in \mathcal{F}_j} \mathcal{P}_i|$
- 3: **while** $(W' - W)/W' > \epsilon$ **do**
- 4: **for** $j = 1$ to n **do**
- 5: $T_{ij}' = T_{ij} + D_i, \forall i \in \mathcal{F}_j$
- 6: compute C_j^* according to Eq. (10)
- 7: sort flows in \mathcal{F}_j in decreasing order of T_{ij}'
- 8: **for** $i \in \mathcal{F}_j$ **do**
- 9: /* varies T_{ij} in Eq. (13) to find T_{ij}^* that depletes
 slack at first preceding inflection point */
- 10: $T_{ij}^* = \max_{k \in \mathcal{F}_j, k > i} \{T_{ij}' : s_{kj} = 0\}$
- 11: $T_{ij} = \max(T_{ij}^*, 0, T_{ij}' - b_i/r_i)$
- 12: $D_i = T_{ij}' - T_{ij}$
- 13: **end for**
- 14: **end for**
- 15: update $C_j^*, \forall 1 \leq j \leq n$ according to Eq. (10)
- 16: $W' = W, W = \sum_{1 \leq j \leq n} C_j^*$
- 17: **end while**
- 18: **return** W

resulting impact and how slack is consumed in the process, is shown in Fig. 8b for flow 3. We note that increasing the reprofiling delay D_3 “raises” the aggregate service curve link j prior to T_{3j}' . Hence, the increase in D_3 can proceed until either one of the following three conditions is met (lines 9–10):

- (a) s_{kj} is depleted¹⁷ at some T_{kj}' , $k \in \mathcal{F}_j$ and $T_{kj}' < T_{ij}'$.
- (b) The flow's own local deadline T_{ij} decreases to 0.
- (c) The reprofiling delay D_i reaches its maximum value b_i/r_i .

Once adjustments have been finalized on a link, the updated profiles (2SRCs) are propagated to all the hops traversed by the affected flows. The resulting smoother profiles translate in potentially lower bandwidth requirements on those links.

We note that, except for the flow with the smallest T_{ij}' that can always be fully reprofiled, reprofiling a flow affects slack at other T_{ij}' 's. Hence, the order in which flows are examined influences reprofiling. We chose to reprofile flows in decreasing order of the T_{ij}' 's (line 7). This is because a larger T_{ij}' minimizes how many other flows are affected¹⁸.

The other ordering affecting the adjustment phase is the one in which links are visited. We visit links in decreasing order of $|\bigcup_{i \in \mathcal{F}_j} \mathcal{P}_i|$ (line 2) to prioritize links whose flows affect the most other links. This ordering notwithstanding, changes in a flow's profile affect slack on all the links it traverses, including previously visited links. Hence, irrespective of the ordering chosen, new reprofiling opportunities may emerge. Realizing them requires revisiting links. To that end, we

¹⁷This is the case in Fig. 8b where D_3 is increased until $s_{2j} = 0$.

¹⁸Because reprofiling only affects slack at smaller T_{ij}' 's.

¹⁶See Algorithm 2

repeat the adjustment process (visiting all network links) until improvements fall below a threshold ϵ (line 3). As the required network bandwidth is lower-bounded, this process converges. Across all our experiments with a threshold of $\epsilon = 0.1\%$, it terminated in 4 iterations or less 90% of the time.

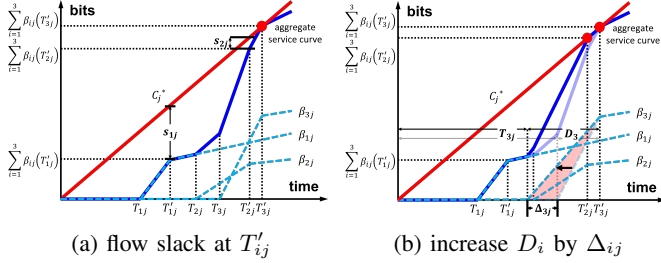


Fig. 8: Greedy flow reprofiling adjustment.

D. Discussion

Recall that each iteration of the exploration phase starts with the *same* reprofiling ratio γ_x for all flows, and splits any remaining delay equally among links. We experimented with alternatives, including assigning different initial reprofiling delays to flows and heterogeneous initial local deadlines across hops. The results were almost statistically identical (slightly worse), and had a longer run time.

Running Greedy calls for selecting L (exploration iterations) and K (reprofiling configurations). The algorithm terminates after L iterations or when an iteration yields an improvement below a configured threshold. In practice, choosing $L = 2$ and $K = 4$ seems to realize a reasonable performance vs. computations compromise. Fig. 28 of Appendix B-G offers evidence in support of this conclusion. We also note that because flows from the same traffic class (same end-to-end deadline) and following the same path can be aggregated, Greedy's run time complexity is $O(N_C \times N_P)$, where N_C and N_P are the number of traffic classes and paths, respectively.

Finally, while Greedy, like the NLP formulation, combines reprofiling and unequal deadlines across hops, its primary driver is reprofiling, with unequal deadline assignments a side effect. Hence, its close approximation of the NLP formulation, as shown next, hints that reprofiling is the dominant factor in decreasing bandwidth. Fig. 29 of Appendix B-G offers evidence in support of this claim. We also acknowledge that the evidence of Greedy's close approximation of NLP is only empirical. Deriving tight approximation bounds remains elusive, in part because the "optimal" solution to **OPT**⁻ relies on the combination of multiple locally optimal solutions for each NLP instance under consideration. This makes applying traditional approximation techniques challenging.

VI. NLP FORMULATION VS. GREEDY

In this section, we compare the cost and performance (bandwidth reduction) of NLP and Greedy. As NLP's complexity grows rapidly with the number of links and flows, our comparison is limited to the simple tandem topology of Fig. 9.

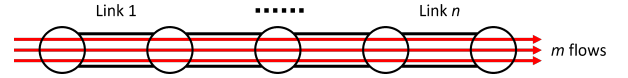


Fig. 9: tandem topology

In comparing the two algorithms, we vary the number m of flows and the number n of hops each flow traverses in the topology of Fig. 9. We also select from a range of traffic profiles, and use different values of *target* per-hop deadlines to construct end-to-end deadlines. The latter means that, as n increases, so do end-to-end deadlines. Figs. 35 and 36 in Appendix B-I investigate an alternative where end-to-end deadlines are fixed, with target per-hop deadlines then decreasing as n increases. Results were qualitatively similar.

Specifically, we consider three per-hop deadlines with a dynamic range of 100, i.e., $d_h \in \{0.01, 0.1, 1\}$, so that end-to-end deadlines are of the form $d = n \cdot d_h$, as the number of hops vary. The value d_h of a flow is randomly chosen from these three values, and its traffic profile (r, b) is set by randomly selecting values in the range $[1, 100]$ for both r and b .

In our first set of experiments, we fix the number of flows to $m = 3$ and vary the number of links n from 2 to 10. The results are shown in Fig. 10, where each data point is based on 150 samples. We use Octeract¹⁹ as our NLP solver. Fig. 10a reports the average relative reduction in overall bandwidth that the NLP formulation affords over Greedy, while Fig. 10b gives the corresponding average run times on an Intel(R) Xeon(R) CPU E5-1620 v3 @ 3.50GHz server. The error bars report the 95% confidence interval for both over a set of 150 experiments. As expected from its "optimality," the NLP formulation outperforms Greedy across all our experiments, but the gap was small (below 3%), and differences in run time quickly exceeded three orders of magnitude as n increased.

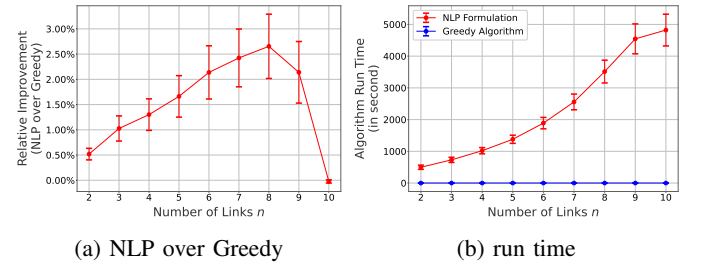


Fig. 10: NLP vs. Greedy ($m = 3$ flows, # links, n , varies).

Fig. 10a also highlights that while the gap between NLP and Greedy first increases with n , it eventually drops to 0. This is expected and an artifact of keeping per-hop deadlines fixed. Specifically, as n grows, so do end-to-end deadlines until they reach a value for which reprofiling all flows to their average rate (a reprofiling delay of $D = b/r$) becomes feasible. Both NLP and Greedy readily discover this solution.

Our next set of experiments uses a similar setup as the first, but varies the number of flows m from 2 to 5 while keeping the number of hops at $n = 2$. The smaller range of the latter is because the complexity of the NLP formulation grows faster

¹⁹ Accessible at <https://octeract.ggf/>

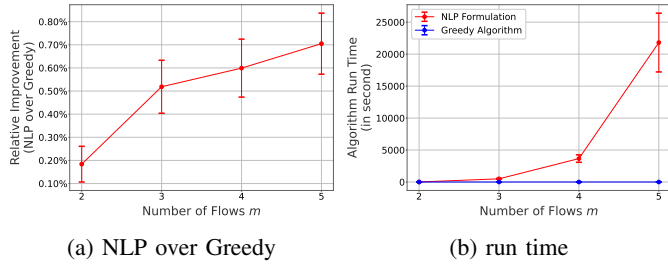


Fig. 11: NLP vs. Greedy ($n = 2$ links, # flows, m , varies).

(factorial) in the number of flows than links. The results are reported in Fig. 11 that parallels Fig. 10 with mostly similar conclusions. Greedy remains within about 0.5% of NLP on average across experiments, and it achieves this with even greater savings in run time, namely, improvements of more than six orders of magnitude over NLP when m is just 5.

For calibration purposes, we also compare in Appendix B-H the NLP-based solution with two baselines. The first, “Full Reprofile” or FR, reprofiles flows as much as possible with a reprofiling delay set to $D = \hat{d} = \min(d, b/r)$. When the deadline d is small, *i.e.*, $d \leq b/r$, it is fully dedicated to reprofiling and the network delay is zero²⁰. When the deadline is larger, *i.e.*, $d > b/r$, the residual delay budget, $d - b/r$, is split evenly across hops. The second, “No Reprofile” or NR, forfeits reprofiling altogether, *i.e.*, sets $D = 0$, and splits the full d evenly across hops. As Greedy includes FR and NR among its initial solutions, it never performs worse than either, and often does better because of its ability to explore intermediate reprofiling configurations.

The conclusion from our two sets of experiments is that while the NLP formulation generates better solutions, its computational complexity makes it impractical for anything but very small configurations. In contrast, Greedy scales well and produces solution that perform nearly as well. From examining the solutions produced by the two approaches, we note though that they usually differ significantly in the configurations they produce. As pointed out in Section V-D, this is not unexpected given their structural differences, and hints at a relatively “flat” solution space around optimal solutions.

VII. EVALUATION

In Appendix B-I we conduct a comprehensive evaluation of the benefits of reprofiling using synthetic topologies and flow profiles whose parameters we vary systematically. The investigation shows that reprofiling yields consistent improvements, often in the double digits in relative bandwidth decreases. However, except for a few intuitive findings, *e.g.*, that full reprofiling is increasingly the best solution as the number of hops a flow traverses increases²¹, it does not offer much insight into the structure of optimal reprofiling decisions. As a result, we concentrate on reporting the benefits of reprofiling (Greedy) for two more realistic configurations.

²⁰Recall that we are assuming a fluid model.

²¹As discussed in Appendix B-I this is true for both fixed hop-by-hop and end-to-end deadlines, albeit for different reasons.

The first configuration is a vehicle network operating according to the TSN standard. The second mimics a network connecting multiple datacenters and the traffic flows between them. For both, the benefits of Greedy’s reprofiling decisions are compared to the two baselines, FR and NR.

In reporting results, the focus is on the bandwidth improvements that Greedy yields over both baselines, with, as before, error bars giving the 95% confidence intervals. Of interest is that Greedy’s outcome often falls between the two extremes of NR and FR. This demonstrates not only the benefits of reprofiling, but more importantly that they involve a trade-off between making flows smoother and preserving scheduling flexibility in the network. Because of space constraints, results on buffer bounds are relegated to Appendices B-J and B-K.

A. Application-Derived Topologies and Flow Profiles

Exploring how reprofiling performs in a more realistic setting calls for identifying representative network topologies and creating a traffic mix composed of flows with profiles derived from actual application traffic. We consider two application scenarios; one reflective of a TSN deployment, and the other that arises in an inter-datacenter network setting.

1) Time Sensitive Networking (TSN):

a) *Representative Network Topology:* We first consider a TSN-inspired network setting. The Orion Crew Exploration Vehicle (CEV) network [22] is the instantiation we use for our evaluation. Fig. 3 of [23] illustrates the network that consists of 31 end devices and 13 switches connected by 47 bidirectional links, each viewed as two independent unidirectional links for bandwidth accounting. Every end device is a possible source or destination for traffic flows. We note that SCED is not a scheduler included in the TSN standard, so that our focus is on evaluating the benefits of reprofiling (for SCED) in a setting where hard delay bounds are relevant.

Paths between sources and destinations (S-D pairs) are minimum hop paths. When multiple choices exist, one is randomly selected as the default. In addition, when generating a set of paths, we omit all instances of single hop paths. This is to avoid bias and noise when comparing Greedy to FR and NR, respectively²².

b) *Flow Profiles:* The TSN standard specifies several traffic classes, including a Control-Data Traffic (CDT) class with stringent latency requirements, and classes A and B with looser latency targets. The traffic characteristics of those three classes are given in Table 2 of [24] through their maximum frame size, minimum frame inter-arrival time and maximum end-to-end delay target. We use that information to create traffic profiles (r, b, d) for flows from all three classes.

End-to-end deadlines d for classes CDT, A, and B are set to 0.1, 2, and 50ms, respectively, directly from Table 2 of [24].

Specifying the tuples (r, b) requires some intermediate steps. We first specify average flow rates for all three classes. We do so by relying on the frame sizes of Table 2 of [24] (128, 256, and 256 bytes for classes CDT, A, and B respectively), and selecting frame inter-arrival times randomly from among a discrete set of values that range from the minimum

²²Recall that the EDF policy of SCED is optimal on a single link.

values of [24] ($500\mu s$, $125\mu s$ and $250\mu s$ for classes CDT, A, and B respectively) up to maximum values a couple of orders of magnitudes larger. Once a flow's average rate is set, we rely on the approach of [25] to determine suitable token bucket parameters (r , b). We assume that frames arrive according to a Poisson process, and choose the flow's token rate r to be 10% higher than the average flow rate, with b then set equal to 25 frame sizes. This combination ensures a 99th percentile of the token bucket access delay of approximately zero.

c) *Scenarios*: Following again the setup of [24], we sample applications and their associated flow profiles from a mixture of the three traffic classes, CDT, A, and B, using a sampling ratio of 1:4:4. Further, because in a TSN setting flows can be either unicast, multicast, or broadcast, we select each transmission mode equally for each flow. A flow's source is first chosen randomly from the 31 end devices in the Orion CEV network. In the case of unicast and multicast, the destinations are also selected randomly from the remaining nodes, with, in the multicast case, the number of distinct destinations itself random between 2 and 29. Hence, an application maps to, on average, 15.5 traffic flows or distinct S-D pairs.

Each experiment consists of 1000 instances of a given scenario. Results are reported across experiments involving an increasing number of applications (from 10 to 200 in steps of 10), and therefore a correspondingly increasing expected number of flows (S-D pairs). For scalability, traffic flows from the same class and S-D pair are aggregated.

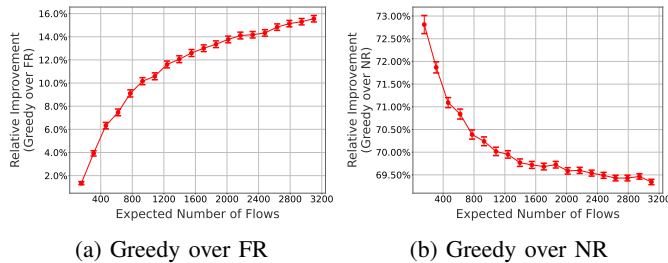


Fig. 12: Greedy's bandwidth improvements for Orion CEV.

d) *Results*: Figs. 12a and 12b report, as a function of the expected number of flows, the bandwidth improvements of Greedy over the two baselines, FR and NR, respectively. The figures show that on the Orion CEV network, Greedy yields bandwidth improvements of up to 16% over FR and 73% over NR.

Having significantly larger improvements over NR than FR is consistent with our intuition from small configurations. Consider a simple scenario with a single flow crossing two hops. In this basic configuration, both FR and Greedy produce the same solution, while NR requires twice as much bandwidth²³. Hence, in this single flow setting, Greedy generates an improvement of 0% over FR and of 50% over NR.

We also note that Greedy's improvements over FR increase with the number of flows. This is again intuitive, as more flows means more deadline combinations, and correspondingly

²³Each hop receives only half the total delay budget, and the original flow profile is assumed at both. Note that avoiding this penalty was the motivation behind the RCSD policies of [21] that essentially emulate FR.

scheduling flexibility that Greedy can leverage as it seeks to balance the benefits of smoother flows and scheduling flexibility. In contrast, FR largely forfeits this option by allocating the full delay budget to making flows smoother. Hence, Greedy outperforms FR as its ability to leverage scheduling flexibility increases with the number flows.

Analyzing the improvements of Greedy over NR as the number of flows varies is more challenging. It involves a complex trade-off between how the number of flows affects the "error" of NR's end-to-end delay bounds and its ability to leverage greater scheduling flexibility, all of which depend on the exact combinations of flow profiles. Hence, the main conclusion of Fig. 12b is that Greedy consistently outperforms NR, irrespective of the number of flows.

Finally, as the number of flows increases, the improvements of Greedy over both FR and NR eventually stabilize. This is because flows associated with the same S-D pair (and therefore path) and from the deadline class can be aggregated. Hence, given that the number of S-D pairs is bounded, once the number of flows is large enough, traffic mixes on all links converge to a steady-state configuration that in turn yields a certain level of improvements.

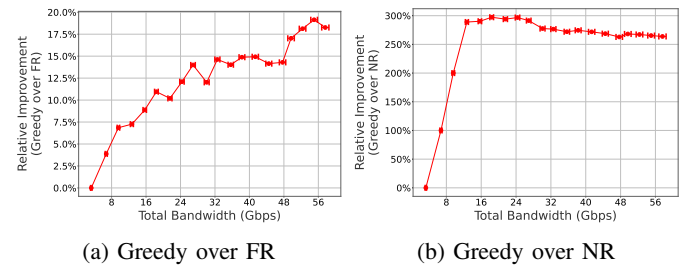


Fig. 13: Greedy's improvements in number of flows accommodated for Orion CEV.

Fig. 13 demonstrates the benefit of reprofiling from a different perspective. It shows Greedy's improvements in number of flows accommodated compared to FR and NR, *assuming that the network bandwidth is fixed*. In other words, it captures the additional traffic (number of flows) a *given* network can carry from relying on Greedy's reprofiling solution. The figure shows that reprofiling allows about 20% more flows than FR and 300% more flows than NR, which is consistent with the bandwidth improvements results from Fig. 12. Specifically, when the number of flows across classes is large enough, the bandwidth required by an "average" flow can be approximated by dividing the total network bandwidth by the number of flows it carries. Under such an assumption, a relative bandwidth improvement of x can then be shown²⁴ to yield a relative improvement of $\frac{x}{1-x}$ in the number of flows carried.

Fig. 14 offers additional insight into how Greedy's improvements are realized. It plots for a 3100 flows configuration, the distribution (over the 1000 instances of the experiment) of the relative *per link* improvements of Greedy over FR across the 47 directional links of the Orion CEV network. It shows that while most links have a lower bandwidth than under FR, this is not true for all links. This is because differences in traffic

²⁴See Appendix B-L for details.

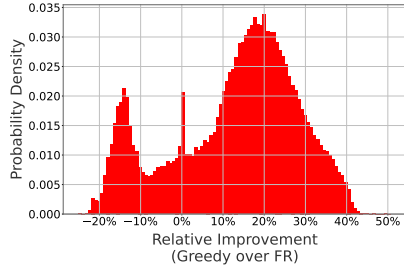


Fig. 14: Distribution of Greedy's bandwidth improvements over FR across the links of Orion CEV (3100 flows).

mix at each hop can affect how much the relative benefits of greater scheduling flexibility matter. In some cases, a (locally) sub-optimal performance on a given link affords larger savings on other links, or savings on multiple links.

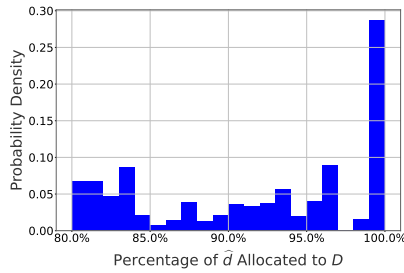


Fig. 15: Distribution of Greedy's reprofiling ratio D/\hat{d} on Orion CEV (3100 flows).

Fig. 15 offers another perspective. It reports the distribution across flows of the fraction of the maximum reprofiling delay \hat{d} used. The figure shows that, while some flows are fully reprofired (the mode at 100%), Greedy also uses intermediate solutions that preserve some scheduling flexibility, which helps it outperform FR. This confirms the existence of a “sweet spot” in the trade-off between tighter network delays (because of the added reprofiling delay), and the benefits of smoother traffic.

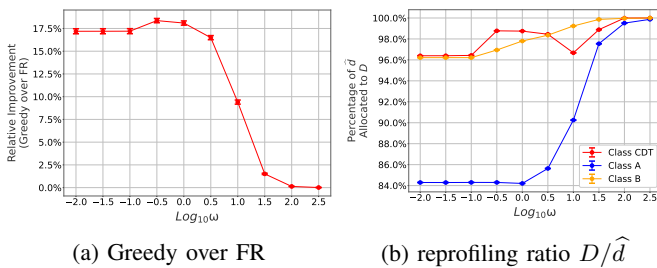


Fig. 16: Reprofiling's behavior on Orion CEV (3100 flows) as a function of deadline scaling (ω).

Although our choice of deadlines was inspired by Table 2 of [24], it is of interest to explore the sensitivity of the results to different choices. We do so using again the 3100 flows configuration but scaling the deadlines of the three traffic classes by a common factor $\omega \in [10^{-2}, 10^{2.5}]$.

Fig. 16a reports the relative improvements in bandwidth of Greedy over FR as deadlines vary (the comparison to NR is omitted as it consistently under-performs FR). The y -axis is

the relative bandwidth reduction of Greedy over FR, while the x -axis is the \log_{10} of the deadlines' scaling factor ω . The figure shows that, as deadlines increase, Greedy's improvements over FR eventually decrease. This is expected since large deadlines allow all flows to be reprofiled at their token rate, *i.e.*, $R = r$, in which case Greedy and FR converge to the same solution. In contrast, tighter deadlines result in increasing improvements that, for the Orion CEV network, stabilize²⁵ at about 17%.

Fig. 16b investigates possible causes for those improvements. It plots the reprofiling ratio D/\hat{d} for the three traffic classes, CDT, A, and B, as ω varies. As expected, class CDT has a consistently high reprofiling ratio. This is because it has the smallest deadline target and, as mentioned in Section V-C, the flow with the smallest T'_{ij} (sum of local deadline and reprofiling delay) can always be fully reprofired during Greedy's adjustment phase. A similar phenomenon is observed for class B, albeit for a different reason. Since class CDT and class A have smaller deadlines and burst sizes no larger than those of class B, class B can keep trading local deadlines for reprofiling without depleting the slack of the other two classes in the adjustment phase²⁶.

The situation is different for class A. When ω is large, CDT's reprofiling rate is set to $R = r$, which eliminates all burstiness. This changes as ω decreases. R gets larger and CDT's burstiness starts to increase. It then becomes possible for class A to use CDT's bandwidth once CDT's initial burst subsides. Realizing this opportunity calls for a smaller reprofiling ratio to reintroduce local deadlines that allow class A to wait for CDT's additional bandwidth to become available.

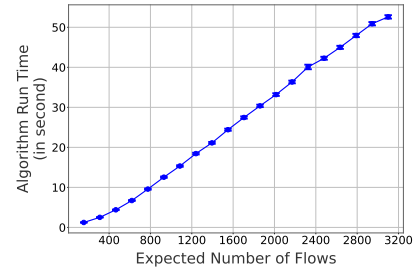


Fig. 17: Greedy's run time on Orion CEV.

Finally, Fig. 17 is intended to demonstrate the scalability of Greedy. Recall that, as mentioned in Section VII-A1c, flows sharing the same path and traffic class are aggregated. As a result, the number of (aggregate) flow combinations that Greedy needs to consider is upper-bounded by the number of distinct paths in the network times the number of traffic classes (end-to-end deadline classes). Although this upper-bound is not reached with the largest (3100) number of flows we consider, the corresponding run time of about 50 secs obtained for the Orion CEV network is reasonable especially in light of the low-end server on which the computations run.

2) Inter-Datacenter Network:

a) *Representative Network Topology*: This scenario targets a topology representative of networks connecting large

²⁵Note though that once deadlines reach 0, the required bandwidth is infinite, so that no improvements are then possible.

²⁶Recall that the adjustment phase considers the largest deadline flows first.

datacenters, and mimics the North-American network connecting the (edge) datacenters of a major cloud provider [26]. The resulting topology (US-Topo) is shown in Fig. 18 and consists of 11 nodes and 23 bidirectional links. Sources and destinations are selected from the 11 nodes and, as with the Orion CEV network, we use minimum hop routing and only multi-hop paths are considered.

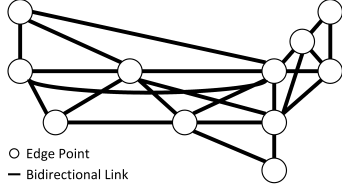


Fig. 18: US-Topo network topology (from [26]).

b) Flow Profiles: Creating realistic traffic mixes is challenging and few studies provide the necessary information. Other works targeting datacenter network optimization [27]–[29] have faced a similar problem, and [29] introduced a set of candidate flow profiles in its own evaluation. Those profiles unfortunately focus on *intra* datacenter traffic, so that most deadlines can be met by guaranteeing a flow its long-term rate, *i.e.*, flows can be reprofiled down to their average rate. This does not affect the evaluation of [29] that is focuses on the benefits of fine-grained routing, but is a poor fit for our investigation of the benefits of reprofiling. As a result, we opt to extract profiles from a study of the traffic traversing segments of a large network connecting datacenters [30].

We selected three applications from [30]: Web (W), Cache read and replacement (C), and Hadoop (H). Fig. 4 of [30] reports their traffic contributions to inter-datacenter links, with Figs. 6 and 7 reporting the distributions of their flow sizes and durations. Because those distributions are reported separately, additional assumptions are needed to create complete profiles. Specifically, we assumed perfectly positively correlated flow sizes and durations, *i.e.*, larger flows last longer than smaller flows, and sample flow size and duration pairs jointly from their respective distributions. This then readily yields cumulative distribution functions (CDFs) for the average (token) rates r of the three applications (see Fig. 19).

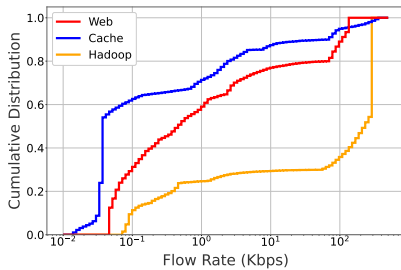


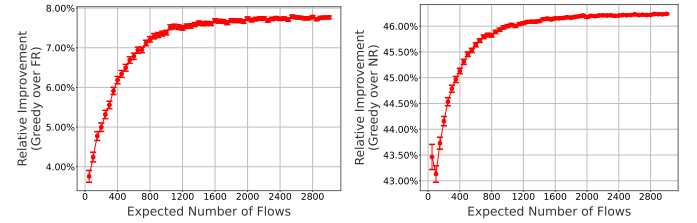
Fig. 19: CDF of flow rate r for Web, Cache, and Hadoop applications derived from [30].

In choosing bucket sizes, we assume that the smallest flow sizes of each application, $S_W = 0.15\text{Kbytes}$, $S_C = 0.4\text{Kbytes}$, and $S_H = 0.3\text{Kbytes}$, correspond to the transmission of an isolated burst. In determining overall burstiness, we rely on

the statement from [30] that flows from these applications tend to be internally bursty, with Cache significantly burstier than Hadoop, and Web somewhere in between. To limit access delay, token bucket sizes are configured to accommodate multiple successive bursts. Specifically, b is drawn randomly from $[0, 20S_C]$, $[0, 10S_W]$, and $[0, 2S_H]$, for Cache, Web, and Hadoop, respectively.

Finally, we set the end-to-end deadlines of Web, Cache, and Hadoop to 10ms, 50ms, and 200ms, respectively. This is motivated by discussions in [30] regarding the services relying on them, *i.e.*, Web search, user data query, and offline analysis (*e.g.*, data mining).

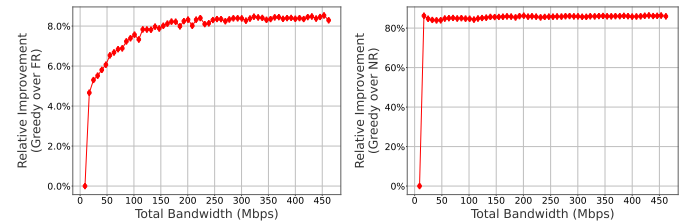
c) Scenarios: We specify possible traffic matrices by selecting a traffic mix of Web, Cache, and Hadoop in the ratio 3:9:1, which seeks to capture their relative usage across services²⁷. The number of flows is then varied from 50 to 3000 in steps of 50, with again, for a given experiment (number of flows), 1000 random instances generated for each result. Traffic from the same application and S-D pair is again aggregated for scalability.



(a) Greedy over FR

(b) Greedy over NR

Fig. 20: Greedy's bandwidth improvements for US-Topo.



(a) Greedy over FR

(b) Greedy over NR

Fig. 21: Greedy's improvements in number of flows accommodated for US-Topo.

d) Results: Fig. 20 mirrors Fig. 12 and reports the relative bandwidth improvements of Greedy over FR and NR. The results are qualitatively similar to those of the Orion CEV network. Bandwidth improvements are a little smaller but still meaningful (about 8% over FR and 46% over NR). Similarly, Fig. 21 evaluates the number of flows the network can accommodate, and Greedy can allow 8% and 90% more flows than FR and NR, respectively.

Fig. 22 duplicates Fig. 14 for US-Topo and plots the distribution of link bandwidth improvements of Greedy over FR for a 3000 flows configuration. The shape of the distribution differs from that of the Orion CEV network, but the results

²⁷Loosely inspired from Table 4 of [30].

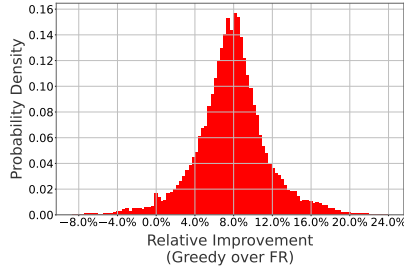


Fig. 22: Distribution of Greedy's bandwidth improvements over FR across the links of US-Topo (3000 flows).

confirm that some links see bandwidth increases (negative improvements) in exchange for larger decreases on other links. How those improvements are realized is reported in Fig. 23 that parallels Fig. 15 and plots the distribution of reprofiling ratios for the 3000 flows configuration. As with the Orion CEV network, intermediate reprofiling solutions are common.

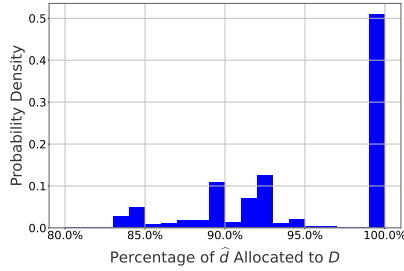


Fig. 23: Distribution of Greedy's reprofiling ratio D/\hat{d} on US-Topo (3000 flows).

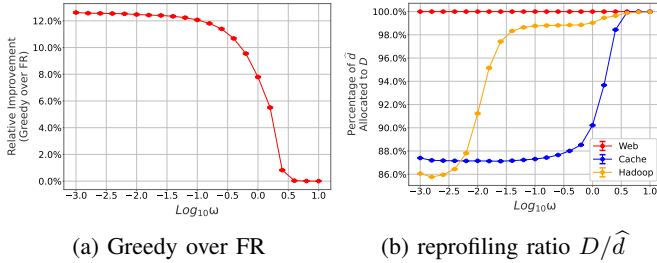


Fig. 24: Reprofiling's behavior on US-Topo (3000 flows) as a function of deadline scaling (ω).

Fig. 24 explores the impact of reprofiling as deadlines vary. As before the deadlines from each class are scaled by a factor ω now spanning the range $[0.001, 10]$. The relative bandwidth improvement of Greedy over FR and the reprofiling ratios of the three applications are reported in Figs. 24a and 24b, respectively, with results largely consistent with those of Fig. 16. A minor difference is that, unlike Class B in Orion CEV, the reprofiling ratio of the largest deadline application, Hadoop, drops to about 86% as ω decreases. This is because, unlike Orion CEV classes CDT and A, Web and Cache have larger burst sizes relative to Hadoop. Hence, as ω decreases so that Hadoop full reprofiling rate exceeds its token rate r , it can start exploiting scheduling flexibility to lower its required bandwidth by taking advantage of bandwidth that becomes

available when the initial bursts of Web and Cache clear. This is similar to what happens to class A in Orion CEV.

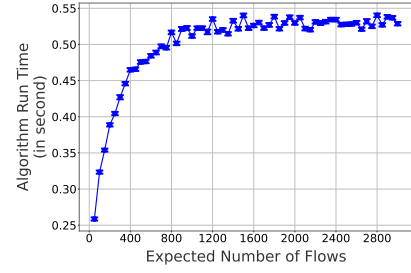


Fig. 25: Greedy's run time on US-Topo.

Finally, Fig. 25 duplicates Fig. 17 in reporting Greedy's average run time as we increase the expected number of flows on the US-Topo network. As with Orion CEV, since flows are aggregated the run time converges to an upper-bound once all paths have been sampled with a stable mix of applications. This happens quickly in the small configuration of US-Topo.

VIII. RELATED WORK

A. Time Sensitive Networking (TSN)

As alluded to, the TSN (and DetNet) standard targets hard delay guarantees for traffic with deterministic traffic profiles. Recent surveys [31], [32] offer comprehensive reviews of both the standard and various scheduling and shaping solutions developed under its guise.

The optimization framework that underlies many of those studies have connections to the problem we address. Their objective function is typically in the form of either minimizing delay or maximizing the amount of traffic that can be accommodated given network resources. In contrast, we primarily target the dual problem of minimizing network resources given a set of flows. More importantly, like most prior scheduling work, traffic profiles are assumed fixed and the impact of reprofiling is not considered.

B. Packet-level shaping and scheduling

Delay guarantees in packet networks have received much attention, and we only sample a few directly relevant works.

An early work [33] introduced latency-two-rates (L2R) service curves similar to the 2SRLSC of this paper. However, it focused on minimizing the resources allocated to *individual* flows [33, Section IV.A], rather than the network-wide, multi-flow perspective of this paper. More importantly, the aspect of flow reprofiling was altogether absent.

Another early example closer to this paper is [21], [34] that relied on *rate-controlled service disciplines* (RCSDs) [35]. The papers demonstrated that by selecting shaping parameters derived from the service curves of *rate proportional processor sharing* (RPPS) disciplines [36], [37], RCSDs could outperform those disciplines and afford a larger schedulability region. The papers, however, explored neither the reprofiling (reshaping) of flows, nor how to allocate per-hop deadlines.

[38] investigated splitting a delay budget between shaping and network delays to minimize bandwidth. The focus was, however, on a single flow and single link.

Finally, [39], and [40] relied on linear programming (LP) and mixed-integer linear programming (MILP) to compute tight worst-case delay bounds under blind and FIFO scheduling, respectively. Flow reprofiling was again absent.

C. Datacenter solutions

Several papers [27]–[29], [41] proposed solutions for predictable and differentiated performance in datacenters. They vary in their type of schedulers and traffic profile models, but typically ignore the potential impact of reprofiling.

WorkloadCompactor [41] is the exception, and while it relies on static priority schedulers, it comes closest to this paper in exploring reprofiling. It investigates how to adjust token bucket parameters to guarantee tail latency while minimizing the number of servers required by a set of workloads. Its reprofiling is, however, limited to a family of feasible regulators²⁸ that do not delay workload requests. In contrast, the reprofiling options we consider can result in added access delays that must then be offset by lower network delays.

D. (Re)shaping bulk transfers

Bulk data transfers must often complete by a deadline, and various approaches [42]–[48] have been proposed to meet this goal while minimizing bandwidth costs. The main difference with this paper is the focus on *transfer* rather than *packet* deadlines. The latter are relevant to real-time or interactive applications. The two perspectives are, however, complementary, as tight packet deadlines often result in relatively low link bandwidth utilization that can then accommodate the coarser bandwidth shaping schedules of bulk data transfers.

E. Deadline assignments

Another set of relevant works [49]–[51] considered how to split end-to-end deadlines across hops, albeit under very different models from ours. In [49], local deadlines are dynamically adjusted based on actual delays incurred at upstream hops. Statically partitioning end-to-end deadlines across hops, as in this paper, is explored in [50], [51]. However, as the notion of arrival curves is absent from both, their solutions are neither applicable nor do they address reprofiling.

IX. CONCLUSION

The paper demonstrated that reprofiling flows can help a network offer delay guarantees with less bandwidth. This highlighted the existence of a trade-off between the added delay this reprofiling entails and the benefits it affords the network from smoother traffic. This trade-off exists in part because the reprofiling delay penalty is incurred once, while its benefits accrue at every hops the flows traverse.

There are many directions in which to extend the work. Simpler schedulers, *e.g.*, static priority or FIFO, are clearly of interest, as they are known to benefit from reprofiling even in the single hop case [13]. Such an extension would, however, likely also include reliance on TSN's inter-leaved shapers [12]

rather than per-flow greedy shapers to minimize in-network complexity. Another natural extension is to consider statistical guarantees. Hard delay bounds are arguably extreme statistics that may not be suitable for all environments. In particular, while they are appropriate in the context of the Orion CEV network, the inter-datacenter networks of Section VII-A2 may benefit from instead considering lower order statistics. Exploring the extent to which reprofiling can still be beneficial when focusing on such metrics is, therefore, also of interest.

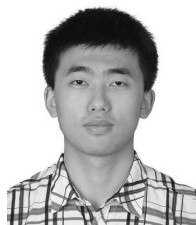
REFERENCES

- [1] M. Ashjaei, L. L. Bello, M. Daneshmand, G. Patti, S. Saponara, and S. Mubeen, "Time-sensitive networking in automotive embedded systems: State of the art and research opportunities," *Journal of Systems Architecture*, vol. 117, p. 102137, 2021. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S1383762121001028>
- [2] *Avionics Full Duplex Switched Ethernet (AFDX) Network*, Airlines Electronic Engineering Committee, Aircraft Data Network Part 7, ARINC Specification 664, Aeronautical Radio, Annapolis, MD, USA, 2002.
- [3] C. Zunino, A. Valenzano, R. Obermaier, and S. Petersen, "Factory communications at the dawn of the fourth industrial revolution," *Computer Standards & Interfaces*, vol. 71, p. 103433, 2020. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0920548919300868>
- [4] T. Docquier, Y.-Q. Song, V. Chevrier, L. Pontnau, and A. Ahmed-Nacer, "Performance evaluation methodologies for smart grid substation communication networks: A survey," *Computer Communications*, vol. 198, pp. 228–246, 2023. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0140366422004285>
- [5] (2022) AWS global network. [Online]. Available: <https://aws.amazon.com/about-aws/global-infrastructure/global-network/>
- [6] (2022) Google cloud networking overview. [Online]. Available: <https://cloud.google.com/blog/topics/developers-practitioners/google-cloud-networking-overview>
- [7] (2021, January) Microsoft global network. [Online]. Available: <https://docs.microsoft.com/en-us/azure/networking/microsoft-global-network>
- [8] J. Farkas, L. L. Bello, and C. Gunther, "Time-sensitive networking standards," *IEEE Communications Standards Magazine*, vol. 2, no. 2, 2018.
- [9] G. Parsons, "The rise of time-sensitive networking (TSN) in automobiles, industrial automation, and aviation," In *Compliance - Electronic Design, Testing & Standards*, January 2022, <https://incompliancemag.com/article/the-rise-of-time-sensitive-networking-tsn-in-automobiles-industrial-automation-and-aviation/>.
- [10] Y. Seol, D. Hyeon, J. Min, M. Kim, and J. Paek, "Timely survey of time-sensitive networking: Past and future directions," *IEEE Access*, vol. 9, pp. 142 506–142 527, 2021.
- [11] N. Finn, P. Thubert, B. Varga, and J. Farkas, "Deterministic Networking Architecture," RFC 8655, October 2019. [Online]. Available: <https://www.rfc-editor.org/info/rfc8655>
- [12] J.-Y. Le Boudec, "A theory of traffic regulators for deterministic networks with application to interleaved regulators," *IEEE/ACM Transactions on Networking*, vol. 26, no. 6, 2018.
- [13] J. Song, J. Qiu, R. Guérin, and H. Sariowan, "On the benefits of traffic 'reprofiling' the single hop case," *IEEE/ACM Transactions on Networking*, vol. TBD, no. TBD, February 2024, extended version available at <https://arxiv.org/abs/2104.02222>
- [14] L. Georgiadis, R. Guérin, and A. K. Parekh, "Optimal multiplexing on a single link: delay and buffer requirements," *IEEE Transactions on Information Theory*, vol. 43, no. 5, 1997.
- [15] J. Liebeherr, D. E. Wrege, and D. Ferrari, "Exact admission control for networks with a bounded delay service," *IEEE/ACM Transactions on Networking*, vol. 4, no. 6, 1996.
- [16] H. Sariowan, R. L. Cruz, and G. C. Polyzos, "SCED: A generalized scheduling policy for guaranteeing quality-of-service," *IEEE/ACM Transactions on Networking*, vol. 7, no. 5, 1999.
- [17] J.-Y. Le Boudec and P. Thiran, *Network Calculus: A Theory of Deterministic Queuing Systems for the Internet*. Springer, 2001, available at <https://leboudec.github.io/netcal/>

²⁸Regulators above the *r-b* curve using the terminology of [41].

- [18] A. Bouillard and G. Stea, "Worst-case analysis of tandem queueing systems using network calculus," in *Quantitative Assessments of Distributed Systems*, ser. Book Series on Performance Engineering, D. Bruneo and S. Distefano, Eds. John Wiley & Sons, 2016, ch. 15. [Online]. Available: <https://hal.inria.fr/hal-01272090>
- [19] 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*, 2020.
- [20] 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*, 2016.
- [21] 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*, vol. 4, no. 4, 1996.
- [22] M. Paulitsch, E. Schmidt, B. Gstöttenbauer, C. Scherrer, and H. Kantz, "Time-triggered communication (industrial applications)," *Time-Triggered Communication*, 2011.
- [23] D. Tămaş-Selicean and P. Pop, "Optimization of TTEthernet networks to support best-effort traffic," in *Proc. IEEE ETFA*, 2014.
- [24] S. Thangamuthu, N. Concer, P. J. Cuijpers, and J. J. Lukkien, "Analysis of ethernet-switch traffic shapers for in-vehicle networking applications," in *2015 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2015.
- [25] C. Li, J. Liu, C. Lu, R. Guérin, and C. D. Gill, "Impact of distributed rate limiting on load distribution in a latency-sensitive messaging service," in *Proc. CLOUD*, 2021.
- [26] (2022) Google cloud global locations. [Online]. Available: <https://cloud.google.com/about/locations#network>
- [27] M. P. Grosvenor, M. Schwarzkopf, I. Gog, R. N. M. Watson, A. W. Moore, S. Hand, and J. Crowcroft, "Queues Don't matter when you can JUMP them!" in *Proc. USENIX NSDI*, 2015.
- [28] K. Jang, J. Sherry, H. Ballani, and T. Moncaster, "Silo: Predictable message latency in the cloud," in *Proc. ACM SIGCOMM*, 2015.
- [29] A. Van Bemten, N. Derić, A. Varasteh, S. Schmid, C. Mas-Machuca, A. Blenk, and W. Kellerer, "Chameleon: Predictable latency and high utilization with queue-aware and adaptive source routing," in *Proc. ACM CoNEXT*, 2020.
- [30] A. Roy, H. Zeng, J. Bagga, G. Porter, and A. C. Snoeren, "Inside the social network's (datacenter) network," in *Proc. ACM SIGCOMM*, 2015.
- [31] T. Stüber, L. Osswald, S. Lindner, and M. Menth, "A survey of scheduling in time-sensitive networking (TSN)," 2022. [Online]. Available: <https://arxiv.org/abs/2211.10954>
- [32] J. Walrand, "A concise tutorial on traffic shaping and scheduling in time-sensitive networks," *IEEE Communications Surveys & Tutorials*, pp. 1–1, May 2023.
- [33] J. B. Schmitt, "On the allocation of network service curves for bandwidth/delay-decoupled scheduling disciplines," in *Proc. IEEE GLOBECOM*, 2002.
- [34] L. Georgiadis, R. Guérin, V. Peris, and K. N. Sivarajan, "The effect of traffic shaping in efficiently providing end-to-end performance guarantees," *Telecommunications Systems*, vol. 5, no. 1, 1996.
- [35] H. Zhang, "Providing end-to-end performance guarantees using non-work-conserving disciplines," *Computer Communications*, vol. 18, no. 10, 1995.
- [36] 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, 1993.
- [37] —, "A generalized processor sharing approach to flow control in integrated services networks: the multiple node case," *IEEE/ACM Transactions on Networking*, vol. 2, no. 2, 1994.
- [38] P. Giacomazzi, L. Musumeci, G. Saddemi, and G. Verticale, "Optimal selection of token bucket parameters for the admission of aggregate flows in IP networks," in *Proc. IEEE GLOBECOM*, 2006.
- [39] A. Bouillard, L. Jouhet, and E. Thierry, "Tight performance bounds in the worst-case analysis of feed-forward networks," in *Proc. IEEE INFOCOM*. IEEE, 2010, pp. 1–9.
- [40] A. Bouillard and G. Stea, "Exact worst-case delay in fifo-multiplexing feed-forward networks," *IEEE/ACM Transactions on Networking*, vol. 23, no. 5, pp. 1387–1400, 2014.
- [41] T. Zhu, M. A. Kozuch, and M. Harchol-Balter, "WorkloadCompactor: Reducing datacenter cost while providing tail latency SLO guarantees," in *Proc. SoCC*, 2017.
- [42] M. Marcon, M. Dischinger, K. P. Gummadi, and A. Vahdat, "The local and global effects of traffic shaping in the internet," in *Proc. COMSNETS*, 2011.
- [43] T. Nandagopal and K. P. Puttaswamy, "Lowering inter-datacenter bandwidth costs via bulk data scheduling," in *Proc. CCGRID*, 2012.
- [44] H. Zhang, K. Chen, W. Bai, D. Han, C. Tian, H. Wang, H. Guan, and M. Zhang, "Guaranteeing deadlines for inter-datacenter transfers," in *Proc. EuroSys*, 2015.
- [45] V. Jalaparti, I. Bliznets, S. Kandula, B. Lucier, and I. Menache, "Dynamic pricing and traffic engineering for timely inter-datacenter transfers," in *Proc. ACM SIGCOMM*, 2016.
- [46] 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, 2018.
- [47] Z. Yang, Y. Cui, X. Wang, Y. Liu, M. Li, S. Xiao, and C. Li, "Cost-efficient scheduling of bulk transfers in inter-datacenter WANs," *IEEE/ACM Transactions on Networking*, vol. 27, no. 5, 2019.
- [48] R. Singh, S. Agarwal, M. Calder, and P. Bahl, "Cost-effective cloud edge traffic engineering with cascara," in *Proc. USENIX NSDI*, 2021.
- [49] C. Li and E. W. Knightly, "Coordinated multihop scheduling: A framework for end-to-end services," *IEEE/ACM Transactions on Networking*, vol. 10, no. 6, 2002.
- [50] D. Marinca, P. Minet, and L. George, "Analysis of deadline assignment methods in distributed real-time systems," *Computer Communications*, vol. 27, no. 15, 2004.
- [51] S. Hong, T. Chantem, and X. S. Hu, "Meeting end-to-end deadlines through distributed local deadline assignments," in *Proc. IEEE RTSS*, 2011.

Jiaming Qiu received his B.S. in 2020 and is currently a Ph.D. candidate, both in Computer Science from Washington University in St. Louis. His research interests include network performance evaluation and optimization, edge computing, and machine learning.



Jiayi Song received her Ph.D. in computer science from Washington University in St. Louis. Prior to her doctoral studies, she obtained her bachelor's degree from Wuhan University, majoring in both applied mathematics and economics. Currently, she is with Bytedance Inc.'s Technical Infrastructures team.



Roch Guérin (F IEEE '01 / F ACM '06) received his Ph.D. from Caltech, worked at the IBM T. J. Watson Research Center and then the University of Pennsylvania. He is currently with the Department of Comp. Sci. & Eng. of Washington University in St. Louis, as the Harold B. and Adelaide G. Welge Professor and department chair. Dr. Guérin was the Editor-in-Chief for the IEEE/ACM Transactions on Networking from 2009 till 2013 and served as chair of ACM SIGCOMM. In 2010 he received the IEEE INFOCOM Achievement Award for "Pioneering Contributions to the Theory and Practice of QoS in Networks".



Henry Sariowan (SM IEEE, '03) received his Ph.D. from the University of California, San Diego, M.S. from Columbia University, New York, and B.S. from Sepuluh Nopember Institute of Technology, Surabaya, Indonesia, all in electrical engineering. He is currently with Google's Global Networking team, part of Google Cloud. Dr. Sariowan previously worked with several technology companies in San Diego, CA and published papers in the areas of network quality of service and video transmission.

