Optimization-Based Network Flow Deadline Scheduling

Andrey Gushchin*, Shih-Hao Tseng†, and Ao Tang†
*Center for Applied Mathematics, Cornell University, Ithaca, New York 14853, U.S.A.
†School of Electrical and Computer Engineering, Cornell University, Ithaca, New York 14853, U.S.A.
Email: {avg36,st688}@cornell.edu, atang@ece.cornell.edu

***Abstract*—Many network flows nowadays, especially in a data center environment, have associated deadlines by which they must be fully transmitted. Nevertheless, traditional transport protocols such as TCP, focus on concepts like throughput and fairness, and do not aim to satisfy flow deadlines. Motivated by this limitation, several alternative transport designs and solutions have been recently proposed. These approaches generally achieve a better performance in terms of the number of satisfied deadlines and are usually built upon various heuristics. In contrast to these previous works, this article approaches the problem directly from an optimization perspective. We first prove that the problem belongs to the class of NP-hard problems that do not even admit a constant ratio approximation solution (unless P=NP), and formulate it as a mixed integer-linear optimization program. Then, using linear programming approximations, we further develop offline and online optimization-based rate control algorithms to approach the problem. Flow-level simulation results indicate that the proposed algorithms can be near-optimal, and hence they can be served as benchmarks against which other solutions to this problem can be evaluated. We additionally performed simulations incorporating such real network features as deployment delays and packet-level granularity to evaluate the performance of the proposed algorithms in a more realistic environment.**

I. Introduction

With the growing popularity of data center technology and real-time network applications, it becomes extremely important to provide network services in a timely fashion. In particular, many network flows are assigned with deadlines by which they must be completely transmitted or will lose value. However, traditional rate allocation approaches, for example TCP transport protocol, generally do not take flow deadlines into account and are not suitable for some modern network environments [1], [2].

The rising Software-Defined Networking (SDN) technology helps achieve a desired level of network efficiency by allowing a fine-grained control over network flows. Furthermore, SDN alleviates the problem of satisfying flow deadlines by facilitating implementation of algorithms designed for this problem in practice. In many cases a network controller has to deal with a large number of simultaneous flow requests demanding services, and sometimes it may not be possible to fulfill all service requests due to, for example, link capacity constraints. It is generally desirable, however, to meet as many requests as possible via a refined rate control of the flows. As we will show in this article, even with a complete control of flow rates of all time, the problem of maximizing the number of satisfied deadlines is extremely difficult in general.

Several approaches tackling the problem of satisfying flow deadlines have been recently proposed for data center network environments. Here we will provide a brief review of some of these works. For example, one of the goals of the deadline-aware control protocol proposed in [3] is to maximize the number of satisfied flow deadlines, and its rate control mechanism is based on a greedy approach. Another protocol for data centers is designed in [4], and the rate allocation there is implemented according to the Earliest Deadline First (EDF) rule [5]. In contrast, the data center transport protocol from [6] employs the Least Attained Service (LAS) scheduling mechanism. The main idea of the approach presented in [7] is to replicate short TCP flows, which in practice helps reduce flow completion time. Congestion control protocols from [2], [8], [9] emulate processor-sharing at each router, so that a router assigns a single rate to all flows that pass through it. Although these protocols do not try to maximize the number of satisfied deadlines explicitly, they do so in an implicit manner by reducing the flow completion times. Solution described in [1] tries to minimize the flow completion times by prioritizing small flows over large flows. A TCP-like protocol called DCTCP designed specifically for data centers is presented in [10]. DCTCP utilizes Explicit Congestion Notification and proposes a novel control scheme at the sources to decrease the latency of short flows and therefore potentially satisfy more flow deadlines compared to traditional TCPs. OpenTCP described in [11] is a TCP adaptation framework for SDN-based data centers that may lead to up to 59% reduction in flow completion times. Another approach for reducing the flow completion times in data center networks is described in [12] and it achieves this goal by utilizing multiple routing paths for each flow.

The aforementioned rate control approaches are generally based on relatively simple heuristics that allow to increase the number of satisfied flow deadlines compared to the state-of-the-art TCP protocol. In contrast to these previous works, in this article we take a principled approach and directly start from the optimization formulation that tries to satisfy as many flow deadlines as possible. There are several theoretical works originating from both scheduling theory ([13], [14]) and optimization ([15]) that provide solutions for some related problems.

The flow deadline scheduling problem studied in this article has two special features that make it both different and more challenging compared to the previously investigated ones,

 DOI: 10.1109/ICNP.2016.7784415

including those in [5], [13], [14] and [15]. First, processing of each flow (job in the context of scheduling theory) simultaneously requires resources of several specific links (processors or machines in the context of scheduling theory) belonging to the flow's routing path. Second, link bandwidth can be shared among different flows if their routing paths contain the same link or several links. These two features make the problem of satisfying network flow deadlines distinctly different from the problems studied in scheduling theory, where it is generally assumed that a job can be served by any set of available identical processors, and processing sharing is not allowed. Additionally, they significantly extend the space of possible scheduling solutions which complicates finding an optimal solution as demonstrated by Example 2.

Depending on the amount of information about the future flows that is available initially (at time zero), we consider two problem setups: offline and online. In the offline setup, all information about the upcoming flows, including their arrival times, sizes and deadlines, is revealed prior to the operation of the system. Therefore, a rate allocation can be precomputed in advance before the arrival of the flows. In the online setup, on the other hand, all information about a flow becomes available only upon its arrival. It is easy to see that an optimal rate control mechanism for the offline setup is at least as good as an optimal control for the online setup.

This paper is organized as follows. In the next section we introduce necessary notations, specify our assumptions and provide optimization problem formulation for the offline environment case. Further, section III contains our analysis of the formulated problem: we show that the problem is NP-hard and does not admit a constant ratio approximation in polynomial time (unless P=NP). After that, for both offline and online environments, we propose approximating algorithms that are based on linear programming relaxation, in section IV. We then perform flow-level and packet-level numerical simulations in section V and conclude in section VI.

II. Problem Formulation

We start this section by providing our assumptions and introducing necessary notations in subsection II-A. After that we proceed to the problem of satisfying flow deadlines in the offline setup when all information regarding the flows, including arrival times, sizes and deadlines, is available at the initial time $t = 0$. We first formulate this problem as an optimal control problem in subsection II-B, and then show in Proposition 1 of subsection II-C that it can also be posed as an optimization problem.

A. Problem Setup

We assume that the network is defined by a directed graph $G = (V, E)$ with V being the set of nodes and E denoting the set of edges. Every edge $e \in E$ is assigned with a nonnegative capacity $c(e) \geq 0$. Further, each network flow f_i is represented by a five-tuple $< a_i, z_i, r_i, s_i, d_i >$, where a_i and z_i are, resp., the source and the destination nodes of the flow f_i, r_i is the release or arrival time of the flow to the network, $s_i > 0$ is the flow size and d_i is its deadline. Release and deadline times for all flows are given on the absolute time scale, and

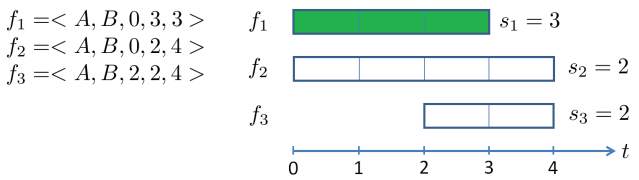


Fig. 1: Earliest Deadline First rule does not maximize the number of satisfied deadlines (Example 1).

for each flow f_i its deadline must be greater than its release time: $d_i > r_i$. Time interval $[r_i, d_i]$ between arrival time and deadline of flow f_i will be called the **lifespan** of this flow.

We examine the problem in which a single routing path is predefined for each source-destination pair, and hence the network operator can only perform the flow rate control while not being able to choose the routing paths. It is further assumed that the flow rate can be adjusted at any time, and the rate control decisions can be immediately deployed in the network. While this assumption may not be realistic in the real networks, it allows us to create a benchmark such that various ad hoc approaches to this problem can be compared with it. Therefore, for each source-destination pair of nodes $v_1 \in V$, $v_2 \in V$ such that $v_1 \neq v_2$, a single routing path $v_1 \to \cdots \to v_2$ is provided, and traffic for flow f_i can be routed using its corresponding single routing path $a_i \to \cdots \to z_i$. At any time t, we can control the sending rate $x_i(t)$ of each flow f_i if $t \in [r_i, d_i]$, such that all link capacity constraints are satisfied. Link capacity constraints require that the total amount of traffic that can be sent at any time t through a particular link is limited by this link bandwidth capacity. We say that the deadline of flow f_i is satisfied, if amount of traffic sent for this flow by its deadline is at least the size of the flow, i.e., if $\int_{r_i}^{d_i} x_i(t)dt \geq s_i$, and the goal is to satisfy as many flow deadlines as possible.

The offline environment implies that at time $t = 0$ all necessary information about the flows is available. In particular, the set of five-tuples $< a_i, z_i, r_i, s_i, d_i >$, $i = 1, \ldots, n$ is provided, where n is the total number of flows. We assume that n and each deadline time d_i are finite, and thus the problem has a finite horizon $T := \max_i d_i$.

B. Optimal Control Formulation

Any rate control mechanism for every time $t \in [0, T]$ assigns a certain rate to each flow f_i. Let $x_i(t)$ denote the assigned rate to flow f_i at time $t \in [0, T]$. We will consider only rate controls satisfying $x_i(t) = 0 \; \forall t \in [0, r_i) \cup (d_i, T]$ for $i = 1, \ldots, n$. Therefore, a flow's rate is zero before its arrival time and after its deadline. For each control mechanism and flow f_i, $i = 1, \ldots, n$, we propose a utility function

$$U_i = \begin{cases} 1, & \text{if } \int_{r_i}^{d_i} x_i(t)dt \geq s_i; \\ 0, & \text{otherwise.} \end{cases}$$

Here we assume that the integral is well-defined. Then, the objective to be maximized by the optimal rate control mech-

anism is defined as the number of satisfied deadlines:

$$\sum_{i=1}^{n} U_i. \tag{1}$$

From the scheduling theory it is known that the Earliest Deadline First (EDF) rule which at any moment of time selects the task with the earliest deadline, is optimal for the case of a single machine with non-sharable resources, i.e., when only a single job can be processed by the machine at any moment of time [5]. Optimality, however, is defined here in the sense of feasibility: if a schedule satisfying all deadlines (i.e., when $\sum_{i=1}^{n} U_i = n$) exists, then the schedule produced by the EDF rule also satisfies all the deadlines. It is easy to see that the EDF approach does not necessary maximize objective (1) as the following example shows.

Example 1 (EDF rule does not maximize the number of satisfied deadlines). *Suppose that the network consists of a single unit capacity link (A, B) and there are three flows with arrival times $r_1 = r_2 = 0$, $r_3 = 2$, with sizes $s_1 = 3$, $s_2 = s_3 = 2$ units, and deadlines $d_1 = 3$, $d_2 = d_3 = 4$ (see Fig. 1). Then, the EDF rule would provide the whole link capacity to flow f_1 at time $t = 0$, and two remaining flows f_2 and f_3 would miss their deadlines. Thus, the value of objective function* (1) *achieved by the EDF rule is equal to one. This is, obviously, not an optimal schedule in terms of objective function* (1)*, since it is possible to satisfy the deadlines of flows f_2 and f_3.*

Because the rate of each flow f_i can be changed at any moment of time $t \in [r_i, d_i]$, the problem of maximizing total utility (1) is an optimal control problem. The state of the system is described by a set of n state variables $S_i(t)$. Each state variable $S_i(t)$ is associated with a corresponding single flow f_i and is defined as follows:

$$S_i(t) = s_i - \int_0^t x_i(t)dt,$$

i.e., it is equal to the residual of a flow that remains to be sent. Rate variables $x_i(t)$ are the control variables. The step function $\delta(x)$ is defined as

$$\delta(x) = \begin{cases} 1, & \text{if } x \geq 0; \\ 0, & \text{if } x < 0. \end{cases}$$

We can now formulate the optimal control problem whose optimal solution maximizes the number of satisfied flow deadlines:

$$\max \sum_{i=1}^{n} \delta(-S_i(T)),$$

$$\forall i, \forall t : \dot{S}_i(t) = -x_i(t), \tag{2a}$$

$$\forall i : S_i(0) = s_i, \tag{2b}$$

$$\forall i, \forall t \in [0, r_i) \cup (d_i, T] : x_i(t) = 0, \tag{2c}$$

$$\forall e \in E, \forall t : \sum_{i:\ e \in f_i} x_i(t) \leq c(e), \tag{2d}$$

$$\forall i, \forall t : x_i(t) \geq 0. \tag{2e}$$

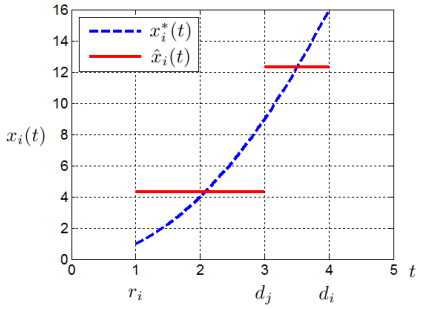


Fig. 2: Example illustrating that for any rate control $x^*(t)$ there exists an equivalent piecewise constant control $\hat{x}(t)$.

Condition (2a) describes the evolution of the system's state variables, and (2b) defines the initial state of the system. Further, condition (2c) requires that the flow's traffic can be sent only during its lifespan. Link capacity constraints have to be satisfied as required by (2d), where notation $e \in f_i$ means that flow f_i's routing path traverses link e, and each flow's rate must be nonnegative by (2e).

C. Optimization Problem Formulation

Notice that the functional of optimal control problem (2) consists only of the endpoint cost part, and does not contain Lagrangian. Furthermore, it is defined as a sum of nonconvex binary-valued functions δ which makes it difficult to find an optimal control for problem (2).

Nevertheless, as shown in the following proposition, an optimal rate control can be found by solving an optimization problem instead of the control problem (2).

Proposition 1. *Let $x_i^*(t), i = 1, \ldots, n$, be an arbitrary feasible rate control of the problem* (2)*. Then, there exists another feasible rate control $\hat{x}_i(t), i = 1, \ldots, n$ with the same objective value, in which each $\hat{x}_i(t)$ is a piecewise constant function that can change value only at the release or deadline times of flows.*

Proof: We prove this statement by showing how to obtain an feasible piecewise constant rate control $\hat{x}_i(t)$ from the given arbitrary feasible rate control $x_i^*(t)$. We divide the whole time interval $[0, T]$ into subintervals $\Delta_j := [t_j, t_{j+1})$, where each t_k is either an arrival time or a deadline of a flow, and $j = 1, \ldots, J$ such that $t_1 = 0$ and $t_{J+1} = T$. Notice, that t_j and t_{j+1} are not necessary arrival and deadline times of the same flow. It is assumed that the last subinterval $\Delta_J = [t_J, t_{J+1}]$ contains both endpoints. In example from Fig. 2 the whole interval $[0, 4]$ is divided into three subintervals $[0, 1)$, $[1, 3)$ and $[3, 4]$.

Then, we define the rates of flows on each time subinterval in a following way:

$$\hat{x}_i(\Delta_j) := \frac{1}{(t_{j+1} - t_j)} \int_{t_j}^{t_{j+1}} x_i^*(t)dt, \tag{3}$$

where $i = 1, \ldots, n$. It is easy to see that the rate control defined by (3) achieves the same value of objective function (1) as the rate control x_i^*, $i = 1, \ldots, n$. It only remains to notice that this rate control also satisfies link capacity constraints at

any time $t \in [0, T]$. Indeed, if it is supposed that there is a link $e \in E$ capacity constraint violation by $\hat{x}_i(t)$ rate control at some interval Δ_j, then

$$\sum_{i:\, e \in f_i} \hat{x}_i(\Delta_j) > c(e),$$

i.e.,

$$\sum_{i:\, e \in f_i} \int_{t_j}^{t_{j+1}} x_i^*(t)dt > c(e) \cdot (t_{j+1} - t_j),$$

which implies that the given rate control mechanism defined by $x_i^*(t)$, $i = 1, \ldots, n$, also violates the capacity constraint for link e during interval Δ_j. ■

In the example from Fig. 2 it is shown how the given rate control function $x_i^*(t)$ for flow f_i can be replaced by a piecewise constant function $\hat{x}_i(t)$.

Proposition 1 allows us to consider only piecewise constant rate control functions in order to obtain an optimal solution to problem (2). Moreover, since in the offline environment all information including arrival and deadline times is available, optimal control problem (2) can be formulated as an optimization problem.

For every flow f_i, $i = 1, \ldots, n$, its lifespan $[r_i, d_i]$ can be represented as a union of intervals Δ_j considered in the proof of Proposition 1. In each such interval Δ_j, the rate of flow f_i is constant, and is denoted by $x_i(\Delta_j)$. Let $|\Delta_j|$ denote the length of interval Δ_j:

$$|\Delta_j| := (t_{j+1} - t_j).$$

The optimization problem whose objective is the number of satisfied deadlines is formulated as follows: [1]

$$\max \sum_{i=1}^{n} \delta\Big(\sum_{\Delta_j \in f_i} (x_i(\Delta_j) \cdot |\Delta_j|) - s_i \Big), \tag{4a}$$

$$\forall e \in E,\ \forall \Delta_j : \sum_{i:\, e \in f_i} x_i(\Delta_j) \le c(e), \tag{4b}$$

$$\forall i,\ \forall \Delta_j \in f_i :\ x_i(\Delta_j) \ge 0, \tag{4c}$$

$$\forall i,\ \forall \Delta_j \notin f_i :\ x_i(\Delta_j) = 0. \tag{4d}$$

Here $\sum_{\Delta_j \in f_i} (x_i(\Delta_j) \cdot |\Delta_j|)$ is the total amount of traffic that is sent for flow f_i, notation $\Delta_j \in f_i$ means that time interval Δ_j belongs to the lifespan $[r_i, d_i]$ of this flow, and $\Delta_j \notin f_i$ implies that Δ_j is not from the lifespan of f_i. Further, $\sum_{\Delta_j \in f_i} (x_i(\Delta_j) \cdot |\Delta_j|) - s_i$ is non-negative if and only if the flow f_i's deadline is satisfied. Constraint (4b) is a link capacity constraint, (4c) requires that all flow rates are non-negative, and the rate must be equal to zero beyond the lifespan of each flow according to (4d).

III. Analysis

In this section we describe some features of optimization problem (4). In particular, in Proposition 2 (subsection III-A) we prove that this optimization problem belongs to the class of NP-hard problems for which no polynomial algorithm with a constant approximation ratio exists (unless P=NP). After that, in subsection III-B we provide an example illustrating that link bandwidth sharing among different flows may lead to a more optimal rate control. Finally, a Linear Programming Approximation (LPA) of optimization (4) is proposed in subsection III-C. The optimal objective value of LPA establishes an upper bound on the maximum possible number of flow deadlines that can be satisfied. Moreover, when all flow deadlines can be satisfied simultaneously, an optimal solution to this LPA relaxation also satisfies them.

A. NP-hardness

Although Proposition 1 allowed us to formulate the original control problem as an optimization problem, finding an optimal (and even an approximating) schedule is an NP-hard problem as shown in the following proposition.

Proposition 2. *The problem of finding an optimal piecewise constant rate control for the offline environment is NP-hard, and cannot be approximated to a constant factor in polynomial time (unless P=NP).*

Proof: We prove this fact by reducing the Maximum Independent Set (MIS) problem to the offline rate control problem. Since MIS is known to be NP-hard with no polynomial time constant factor approximation (unless P=NP), the proposition follows.

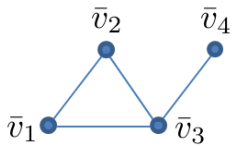
Given an instance of the MIS problem represented by an undirected graph $\bar{G} = (\bar{V}, \bar{E})$, we construct the directed graph $G = (V, E)$ from $\bar{G}$ for the corresponding instance of the offline rate control problem as follows. Let $\bar{V} = \{\bar{v}_1, \ldots, \bar{v}_{|\bar{V}|}\}$. For each vertex $\bar{v}_i \in \bar{V}$, we construct two vertices $a_i, z_i \in V$ and make them a source-destination pair (a_i, z_i) connected by a directed edge $a_i z_i \in E$. After transforming the set $\bar{V}$, we obtain $|\bar{V}|$ source-destination pairs. Each pair has only one feasible path, and the path has at least one edge that is not shared with any path connecting some other source-destination pair. We call those edges non-shared and we will maintain the property throughout the rest of the construction.

For each $\bar{v}_i \bar{v}_j \in \bar{E}$, we construct two vertices $b_{ij}, c_{ij} \in V$ and a shared directed edge $b_{ij} c_{ij} \in E$. From the unique path connecting (a_i, z_i), we select a non-shared edge $\alpha_i \zeta_i$, replace it with two non-shared directed edges $\alpha_i b_{ij}, c_{ij} \zeta_i$ and a shared edge $b_{ij} c_{ij}$, then we remove $\alpha_i \zeta_i$ from E. Similarly, we direct the unique path for (a_j, z_j) to go through $b_{ij} c_{ij}$. After the redirection, the property mentioned previously is maintained, and the unique paths for (a_i, z_i) and (a_j, z_j) share an edge $b_{ij} c_{ij}$.

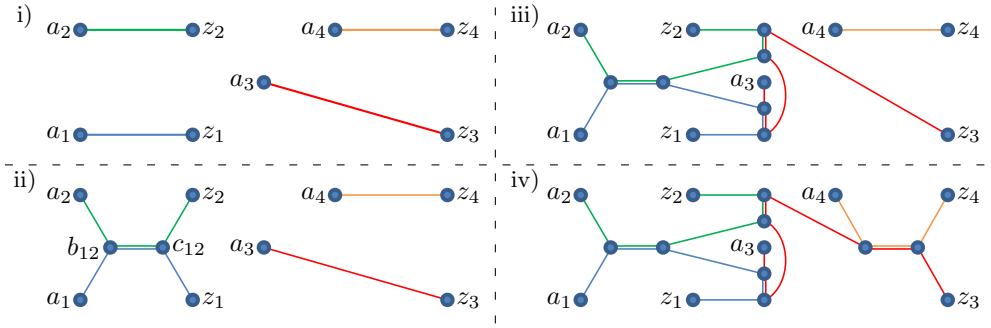
We finish the construction by letting all the edges in E have unit capacity and assigning

$$\forall i : r_i = 0,\ s_i = 1,\ d_i = 1,$$

[1] Although optimization (4) is not a Mixed Integer Linear Program (MILP), it can be easily reformulated in this form using ideas from [16], for example.

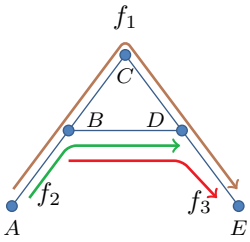


(a) Graph $\bar{G}$ of the MIS problem.



(b) Construction of graph G for the offline rate control problem. Paths for source-destination pairs (a_1, z_1), (a_2, z_2), (a_3, z_3) and (a_4, z_4) are shown in blue, green, red and orange, respectively.

Fig. 3: Example showing how an instance of the offline rate control problem can be obtained from a given instance of the Maximum Independent Set problem.



$f_1 =< A, E, 0, 0.7, 1 >$
$f_2 =< A, D, 0, 0.3, 1 >$
$f_3 =< B, E, 0, 0.1, 1 >$

Fig. 4: Link bandwidth sharing enables satisfying more flow deadlines (Example 2).

to form the flows f_i for all i.

An independent set is equivalent to a feasible offline rate control by the assignment: vertex $\bar{v}_i$ belongs to the independent set if and only if the deadline of f_i is satisfied. Therefore, the MIS problem can be reduced (in polynomial time) to the offline rate control problem, which proves the proposition. ■

Fig. 3 illustrates the process of instance transformation in Proposition 2.

Proposition 2 implies that unless P=NP, it is not possible to provide a polynomial time algorithm with a constant approximation factor for a general case of the offline rate control problem (4). Along with Proposition 1, we can deduce that problem (2) also has no constant-ratio approximation unless P=NP. It does not necessarily mean, however, that no algorithm exists with a high average performance in practice. In particular, in section IV we propose an optimization-based algorithm called ILPA for the offline rate control problem and further demonstrate in section V that its excellent performance.

B. Link Bandwidth Sharing

Another question of interest is whether link bandwidth sharing among the flows may allow to achieve a higher objective value compared to schedules when the bandwidth sharing is not permitted. It can be easily seen that the answer is yes if capacities can be different for different links. It is also true, however, even when all link capacities are equal as can be demonstrated by the next example.

Example 2 (bandwidth sharing allows to satisfy more flow deadlines). *Consider the network topology that consists of five unit capacity links as illustrated in Fig. 4. Further, three flows f_1, f_2 and f_3 arrive to the network at time $t = 0$ and have equal deadline $t = 1$. If link bandwidth sharing is allowed, it is possible to satisfy all three deadlines, and if it is not permitted, only two deadlines (at most) can be satisfied.*

C. Linear Programming Approximation (LPA)

A linear optimization problem approximating the original NP-hard problem (4) can be formulated as follows.

$$\max \sum_{i=1}^{n} \sum_{\Delta_j \in f_i} \frac{x_i(\Delta_j) \cdot |\Delta_j|}{s_i}, \tag{5a}$$

$$\forall e \in E,\ \forall \Delta_j : \sum_{i:\ e \in f_i} x_i(\Delta_j) \le c(e), \tag{5b}$$

$$\forall i : \sum_{\Delta_j \in f_i} x_i(\Delta_j) \cdot |\Delta_j| \le s_i, \tag{5c}$$

$$\forall i,\ \forall \Delta_j \in f_i : x_i(\Delta_j) \ge 0, \tag{5d}$$

$$\forall i,\ \forall \Delta_j \notin f_i : x_i(\Delta_j) = 0. \tag{5e}$$

In contrast to problem (4), objective (5a) of this optimization problem is a linear function of the variables, and favors flows of smaller size. Link capacity (5b) and non-negativity (5d), (5e) constraints remain unchanged. A new set of constraints (5c) was added requiring that an amount of flow that can be sent for each flow f_i is limited by its demand s_i. In the rest of this article optimization problem (5) will be referred as LPA.

Notice that LPA (5) is a relaxation of the NP-hard problem (4), and while an optimal solution to LPA (5) may not necessary maximize the number of satisfied deadlines, it has two important properties. First, if it is possible to satisfy all flow deadlines, then an LPA optimal solution does it. Second, the optimal value of objective function (5a) provides an upper bound on the total number of flow deadlines that can be satisfied, i.e., an upper bound on the optimal objective value

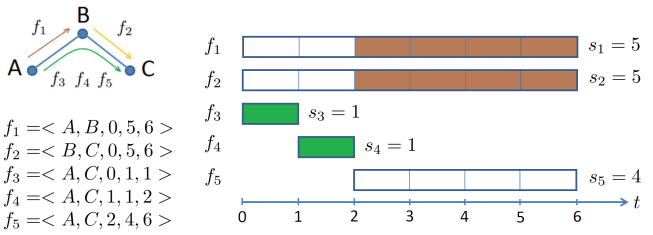


Fig. 5: Illustration of a drawback of the LPA approach (Example 3).

of problem (4). We emphasize here the difference between the optimal solutions to optimizations (4) and (5). Optimal objective value of optimization (5) is always greater or equal to optimal objective value of (4). On the other hand, an optimal solution to (5), compared to an optimal solution to (4), may achieve a lower value in terms of objective (4a), i.e., it may satisfy fewer flow deadlines as can be observed in Fig. 6. These properties are summarized in the following proposition.

Proposition 3. *An optimal objective value of LPA* (5) *provides an upper bound on the maximum possible number of flow deadlines that can be satisfied. If an optimal solution to* (4) *satisfies all flow deadlines, then so does an optimal solution of LPA* (5)*.*

Therefore, there is a similarity between the EDF rule and LPA (5): both approaches satisfy all flow deadlines when it is possible. The difference, however, is that the EDF rule is guaranteed to do this only in the case of a single link (or single source-destination pair), while relaxation (5) satisfies all flow deadlines in a general network. On the other hand, LPA (5) is an approach for the offline environment, i.e., it requires all information about the future flows, while the EDF rule does not have such requirement.

IV. Algorithmic Solutions

This section contains our optimization-based rate control algorithms for offline and online setups. Although LPA algorithm satisfies all flow deadlines when it is possible, its performance may be not optimal in the case when all deadlines cannot be satisfied simultaneously. Therefore, in subsection IV-A we propose Iterative LPA (ILPA), an iterative algorithm for the offline setup that generally outperforms LPA. Next, in subsection IV-B we investigate the case of the online environment for which we design an approximating online algorithm called OLPA.

A. Offline Setup: ILPA

The LPA approach proposed in the previous section has a drawback: in an optimal solution to (5), in an arbitrary interval $\Delta_j = [t_j, t_{j+1})$ traffic can be sent for flows whose deadlines cannot be satisfied anymore. It can be illustrated with the following example.

Example 3 (drawback of LPA). *As shown in Fig. 5, the network consists of two unit capacity links, there are three source-destination pairs (A, B), (B, C) and (A, C), and five flows $f_1 - f_5$. For this example, an optimal solution to LPA* (5) *will first finish flows f_3 and f_4, and then at time $t = 2$ will start sending traffic for flows f_1 and f_2, although it is clear at time $t = 2$ that the deadlines of these flows cannot be satisfied anymore. An optimal rate control solution would instead send traffic for flow f_5 in the time interval $[2, 6]$ to satisfy in total three flow deadlines instead of two deadlines satisfied by LPA.*

Algorithm 1: Iterative LPA for offline environment

input : $\mathcal{S}$, Δ_j $(j = 1, \dots, J)$;
output: rates $x_i(\Delta_j)$ $\forall f_i$, $\forall \Delta_j \in f_i$;

$\mathcal{S}_0 := \mathcal{S}$;
for $j = 1$ **to** J **do**
  $\mathcal{S}_j := \varnothing$;
  for each flow $f_i \in \mathcal{S}_0$ **do**
    find g_i^j using (6);
    if $d_i > t_j$ **and** (7) holds **and** $g_i^j > 0$ **then**
      $\mathcal{S}_j := \mathcal{S}_j \cup f_i$;
    end
  end
  $n_j := |\mathcal{S}_j|$;
  find $x_i(\Delta_j)$ for each $f_i \in \mathcal{S}_j$ by solving (8);
  $x_i(\Delta_j) = 0$ for each $f_i \notin \mathcal{S}_j$;
  $\mathcal{S}_0 := \mathcal{S}_j$;
end

To overcome this drawback we propose iterative Algorithm 1 that at each time t_j maintains a set of flows f_i with positive remaining size, with $d_i > t_j$ and whose deadlines it is still possible to satisfy. To formalize the algorithm, some additional notations must be introduced. First, we will denote by $\mathcal{S}$ the whole set of n flows. Next, let c_i denote the bottleneck link capacity of flow f_i defined as the minimum capacity over all links belonging to the routing path of this flow:

$$c_i := \min_{e \in f_i} c(e).$$

We suppose that the lifespan $[r_i, d_i]$ of flow f_i is a union of $k_i \geq 1$ intervals:

$$[r_i, d_i] = \Delta_l \cup \Delta_{l+1} \cup \dots \cup \bar{\Delta}_{l+k_i-1},$$

here $\bar{\Delta}_{l+k_i-1} = [t_{l+k_i-1}, t_{l+k_i}]$, $t_l = r_i$ and $t_{l+k_i} = d_i$. Further, by g_i^j we denote the remaining size of flow f_i at time t_j defined as

$$g_i^j := \begin{cases} s_i, & \text{if } j \leq l \\ s_i - \sum\limits_{p=l}^{j-1} x_i(\Delta_p) \cdot |\Delta_p|, & \text{if } l < j < l + k_i \\ s_i - \sum\limits_{p=l}^{l+k_i-1} x_i(\Delta_p) \cdot |\Delta_p|, & \text{if } j \geq l + k_i. \end{cases} \quad (6)$$

where the sums in the second and third cases ($j > l$) correspond to the amount of traffic that has been sent for this flow up to the time t_j. Then, it is still possible to satisfy the deadline of flow f_i at time t_j, $j < l + k_i$, if and only if

$$g_i^j \leq c_i \cdot \sum_{p=\max(j,l)}^{l+k_i-1} |\Delta_p|. \quad (7)$$

Algorithm 1 at each time t_j selects flows f_i with $d_i > t_j$, with positive remaining size $g_i^j > 0$ and whose deadlines can

still be satisfied (lines 6-8). Only for such flows linear program (8) is solved, and the flow rate assignment on the interval Δ_j is defined by its optimal solution (lines 11-12). Notice that although a solution to optimization (8) contains rate assignments for all intervals Δ_p where $j \le p \le J$, Algorithm 1 at step j uses the part of this solution corresponding to the nearest interval Δ_j. Rate assignment for intervals Δ_p, $p > j$ is determined at the further iterations of the algorithm.

$$\max \sum_{i=1}^{n_j} \sum_{\substack{\Delta_p \in f_i, \\ p \ge j}} \frac{x_i(\Delta_p) \cdot |\Delta_p|}{g_i^j}, \tag{8a}$$

$$\forall e \in E,\ \forall \Delta_p,\ p \ge j : \sum_{i:\ e \in f_i} x_i(\Delta_p) \le c(e), \tag{8b}$$

$$\forall i : \sum_{\substack{\Delta_p \in f_i, \\ p \ge j}} x_i(\Delta_p) \cdot |\Delta_p| \le g_i^j, \tag{8c}$$

$$\forall i,\ \forall \Delta_p \in f_i,\ p \ge j :\ \ x_i(\Delta_p) \ge 0, \tag{8d}$$

$$\forall i,\ \forall \Delta_p \notin f_i,\ p \ge j :\ \ x_i(\Delta_p) = 0. \tag{8e}$$

The main difference between linear program (8) and LPA (5) is that remaining flow size g_i^j is used in optimization (8) instead of the total flow size s_i.

B. Online Setup: OLPA

We now consider the online environment when it is assumed that no information about a flow f_i is available prior to its arrival time r_i. Instead, all details about a flow become available at the flow's arrival time r_i.

Algorithm 2 for online environment (Online LPA or OLPA) maintains a set of flows $\mathcal{S}$ whose deadlines it is still possible to satisfy, and invokes Algorithm 1 every time a new flow (or several flows) arrive. Therefore, between two subsequent flow arrival times t_k and t_{k+1} the flow rate assignment is obtained by applying Algorithm 1 to the flows with arrival time less or equal to t_k. Then, when a set of new flows F ($|F| \ge 1$) arrives at time t_{k+1}, it is added to the flow set $\mathcal{S}$ (line 11), and sizes of all flows are updated (line 6). Here $g_i(t)$ denotes the remaining size of a flow f_i at time t, i.e.,

$$g_i(t) = s_i - \int_{t^*}^{t} x_i(\tau) d\tau. \tag{9}$$

Further, from the set $\mathcal{S}$, flows whose deadlines cannot be satisfied anymore, flows with $d_i \le t$, or of zero remaining size are removed from this set (lines 7 - 9). It is not possible to satisfy the deadline of a flow f_i at time t, if

$$g_i(t) > c_i \cdot (d_i - t). \tag{10}$$

After that, a set of subintervals Δ_j dividing interval $[t_{k+1}, \max\limits_{f_i \in \mathcal{S}} d_i]$ is obtained, and Algorithm 1 is invoked for the flow set $\mathcal{S}$ and intervals Δ_j.

To demonstrate the operation of Algorithm 2, we apply it to the example shown in Fig. 5. At time $t = 0$ a set of flows

Algorithm 2: OLPA: LPA for online environment

```
1  S := ∅;
2  t* := 0;
3  while 1 > 0 do
4      if set of flows F arrives at time t then
5          for each flow f_i ∈ S do
6              s_i := g_i(t) defined by (9);
7              if d_i ≤ t or (10) holds or s_i = 0 then
8                  S := S \ f_i;
9              end
10         end
11         S := S ∪ F;
12         n := |S|;
13         obtain intervals Δ_j (j = 1, ..., J_t) for S;
14         invoke Algorithm 1 for S, Δ_j;
15         use rate assignment obtained by Algorithm 1;
16         t* := t;
17     end
18 end
```

$F = \{f_1, f_2, f_3\}$ arrives and it is added to the empty set $\mathcal{S}$: $\mathcal{S} = \{f_1, f_2, f_3\}$, $n = 3$. At this step remaining size of each flow from $\mathcal{S}$ is equal to its initial size. The horizon T of the problem at time $t = 0$ is equal to $\max\limits_{f_i \in \mathcal{S}} d_i$, i.e., $T = 6$. Time interval $[0, T]$ is divided into two subintervals $\Delta_1 = [0, 1)$ and $\Delta_2 = [1, 6]$. For these subintervals Δ_j, $j = 1, 2$ and flow set $\mathcal{S}$, Algorithm 1 is invoked, and optimization problem (8) is solved for each subinterval Δ_j. The output of Algorithm 1 at time $t = 0$ is the following rate assignment: on the interval Δ_1, flow f_3 is processed, and then processing of flows f_2, f_3 is performed on the interval Δ_2. This rate assignment is implemented on $\Delta_1 = [0, 1)$, i.e., the deadline of flow f_3 is satisfied by $t = 1$, but then a new flow f_4 arrives at this time. Therefore, at time $t = 1$ flow f_3 is removed from the set $\mathcal{S}$, because $d_3 = t = 1 \le 1$ (lines 7 - 9 of Algorithm 2), and set $\mathcal{S}$ consists of flows f_1, f_2 and f_4 with $n = 3$. Then, interval $[1, 6]$ is divided into two intervals $\Delta_1 = [1, 2)$ and $\Delta_2 = [2, 6]$, and Algorithm 1 is applied to the flow set $\mathcal{S}$ and these intervals. Rate assignment obtained by Algorithm 1 at time $t = 1$ will satisfy the deadline of flow f_4 on the interval $[1, 2)$. Further, at time $t = 2$, when flow f_5 arrives, flows f_1 and f_2 are removed from $\mathcal{S}$, since for each of these two flows condition (10) is satisfied: $g_1(t = 2) = g_2(t = 2) = 5$, $c_1 = c_2 = 1$, and $(d_1 - t) = (d_2 - t) = 4$ for $t = 2$. In addition, flow f_4 is also removed from the set $\mathcal{S}$ since $d_4 = t = 2$. Thus, at time $t = 2$ the flow set $\mathcal{S}$ contains only one flow f_5, and its deadline will be satisfied by time $t = 6$.

V. Simulations

To evaluate and compare the performance of our algorithms we implemented two types of numerical simulations: flow-level and packet-level. Flow-level simulations do not take into account some features of the real network flows, but allow us to estimate theoretical performance of different rate control mechanisms. On the other hand, packet-level simulations

demonstrate how various rate control approaches behave in a more realistic environment.

A. Flow-level Simulations

Results of the flow-level simulations were obtained for a tree network topology shown in Fig. 7, where source-destination pairs were formed by the green (bottom level) switches only. Performance of the algorithms was compared with the best-effort objective value of optimization (4) obtained using COIN-OR [17] branch and cut (CBC) finding the best feasible solution to the mixed integer linear program. The timeout of the CBC solver was set to 10 minutes due to the computational power limitation.

In the experiments the time line was divided into unit length intervals, and it was assumed for simplicity that flows can arrive and depart only at the beginning of each interval. For each source-destination pair, the flows arrived according to a Poisson distribution with a specific for each pair rate. Further, the flow sizes were distributed according to the exponential distribution with a specific for each source-destination pair parameter, and the deadline of a flow was proportional to its size such that the coefficient of proportionality was the same for all flows. Capacity of each link was equal to 2. All flow arrival rate and size parameters were generated randomly according to a uniform distribution. In particular, arrival rate was sampled either from interval $(0, 2)$ or from $(0, 4)$. Flow sizes were sampled from the interval $[0, x]$, where x was varying from 4 to 16 with a step of 3 in each experiment. Deadline of each flow was defined as follows: let τ be the minimum possible time in which a flow can be completely transmitted, i.e., under the assumption that all link bandwidth is available. Then, deadline d of this flow satisfies $(d-r) = q\tau$. Parameter q, therefore, regulates the tightness of the deadlines: $q = 1$, for example, implies that a flow's deadline can be satisfied only if maximum available link bandwidth (determined by the capacity of the bottleneck link) is provided over the whole lifespan of this flow.

Fig. 6 illustrates the results of the flow-level simulations, in which 100 random cases were simulated for each x and the medians of the fractions of satisfied deadlines are plotted. As can be observed, Iterative LPA (ILPA) generally slightly outperforms LPA and OLPA. Additionally, ILPA is generally within 10% of the 10-minute timeout best-effort performance, which is better than LPA (15%) and OLPA (20%).

B. Packet-level Simulations

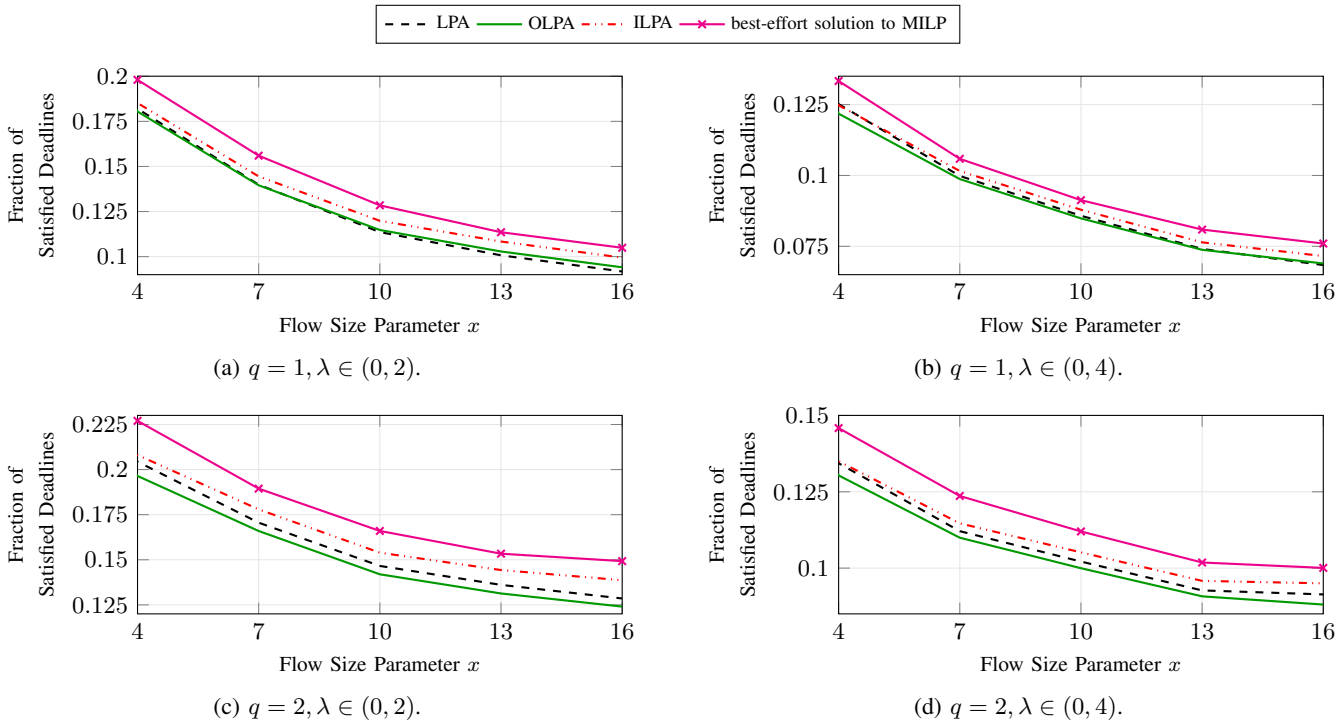
Results of flow-level simulations demonstrate that the proposed algorithms achieve a high level of performance in an idealistic environment, i.e., when the network controller has freedom to adjust the flow rates arbitrarily without deployment delays and the packet-level granularity is not taken into account. To evaluate how well our rate control algorithms behave under more realistic scenarios, we implemented LPA, ILPA, OLPA, pFabric [1], and D^3 [3] on ns-3.25 with COIN-OR [17] linear programming (CLP) as the LP solver. For pFabric, we used the remaining time to the deadline quantized in microseconds to prioritize the packets. We allowed D^3 to quench the expired flows.

1) Simulation Setup: Extensive packet-level simulations were implemented under two different scenarios: the request burst scenario and the stochastic demand scenario. We conducted simulations for a multi-tier tree data center network with the topology shown in Fig. 8. The network consisted of 12 hosts, 4 top-of-rack (ToR) switches and a root switch as in [3]. Each packet in our simulations had a fixed size of 2500 bytes. Taking into account the protocol overheads, the capacities of the links were set to 10.2 Gbps bidirectionally with propagation delay 1 microsecond (μs) to provide 10 Gbps capacity for the packet payloads. We also set all transmit and receive buffer sizes to 125 KB. In each simulation, flows utilized the shortest paths to send data to their destinations. As assumed in the formulation, each flow chose only one path to send its traffic if multiple choices were available. A flow was deemed satisfied if it was completely transmitted by its deadline plus the connection establishment time (1.5 RTT for TCP connection). We kept track of the fraction of the flows that met their deadlines, which is also named *application throughput* in [3] and [1].

2) Request Burst Scenario: We first focus on the request burst scenario that takes place when a number of flows arrive at the network at the same time. The concurrent flows, which have the same arrival time, were generated to be sent between randomly picked source-destination pairs of the hosts. We varied the number of concurrent flows to create different network workloads. The size of each flow was uniformly distributed over $[5, 100]$ KB, and the deadline of the flow was set according to exponential random variables with mean 2, 3, and 4 milliseconds (ms). For each number of concurrent flows, we conducted 10 experiments, and the fraction of satisfied deadlines was defined based on the aggregated number of the flows generated in all 10 experiments. Simulation results shown in Fig. 9 demonstrate that LPA, ILPA, and OLPA all outperform pFabric and D^3 as the number of concurrent flows increases.

Our methods allocate bandwidth before packets arrive at the network. As such, we can utilize available bandwidth more efficiently than pFabric and D^3. pFabric does not limit the sending rate of the flows, and congestion can happen when bandwidth is not enough to support all the flows. D^3 needs the packets to probe the routing path, which is generally not as efficient as our approach. pFabric performs better than D^3 in this scenario since the D^3 allocates the bandwidth in a first-come-first-serve manner. The flows that are closer to their destination can benefit from this bandwidth allocation strategy no matter when their deadlines are. However, under an online scenario, flows no longer have the same arrival time and D^3 can perform better than pFabric because of the bandwidth reservation for the early coming flows. We can observe the phenomenon in the next simulation.

3) Stochastic Demand Scenario: Unlike request bursts, stochastic demands arrive at the network intermittently. We let each source-destination pair of hosts send flows to the network according to a Poisson process with the average inter-arrival time uniformly distributed over $(0, 50]$ milliseconds. The flow sizes were uniformly distributed over $[0, x]$ KB and the deadlines were set according to the tightness parameter q



(a) $q = 1, \lambda \in (0, 2)$.

(b) $q = 1, \lambda \in (0, 4)$.

(c) $q = 2, \lambda \in (0, 2)$.

(d) $q = 2, \lambda \in (0, 4)$.

Fig. 6: Flow-level simulation results. The value of the flow size parameter x determines interval $[0, x]$ from which the flow sizes are sampled.

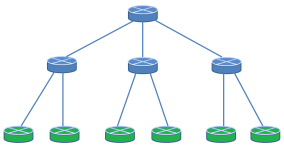


Fig. 7: Network topology used for flow-level simulations.

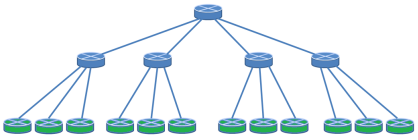


Fig. 8: The multi-tier tree topology used for packet-level simulations.

similarly to the flow-level case. In particular, given q, we set the deadline of a flow as the arrival time plus q times of the minimum time needed to complete the flow when the flow could acquire full capacity of all the links on its path.

We varied the flow size parameter x to adjust the load of the system. Under different loads, the fraction of the satisfied deadlines converged to a limit as the system approached the steady state. 100 traffic instances were generated randomly per x and we plotted the median of the corresponding limits along with the flow size parameters in Fig. 10, in which the online solutions OLPA, pFabric, and D^3 were simulated and compared. We also fixed $x = 50$ (KB) and generated 300 random traffic instances to obtain the statistical results shown in Fig. 11.

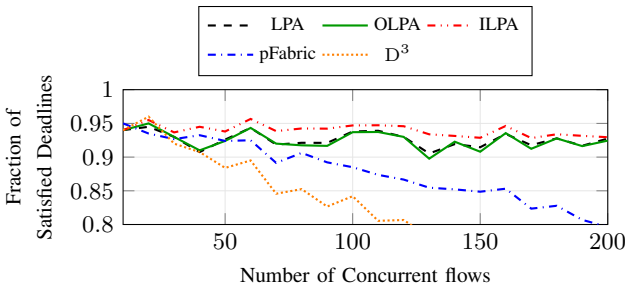
Not surprisingly, larger tightness parameter q enables the algorithms to achieve more deadlines. In Fig. 10, OLPA satisfies a large fraction of deadlines and it outperforms pFabric and D^3. OLPA can maintain a good performance while the load is getting heavier (with larger x). Statistically, OLPA is also more promising in achieving high deadline satisfaction fraction as shown in the box plots in Fig. 11. As discussed in the previous part, D^3 can perform better than pFabric statistically when the deadlines are tight (Fig. 11a).

VI. Conclusion

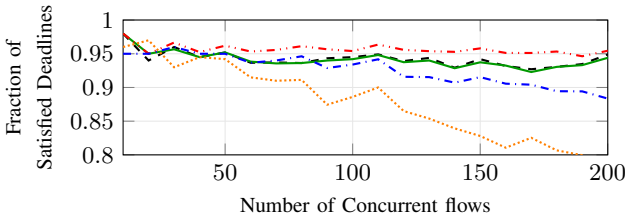
In this article we investigate the problem of scheduling network flows with an objective to maximize the number of satisfied flow deadlines. Unlike previous heuristic-based attempts, we approach the problem directly from an optimization perspective and propose relaxation-based benchmark methods that can be used to evaluate the performance of other solutions to this problem. We show that the problem is NP-hard, and moreover, it cannot be approximated within a constant factor in polynomial time (unless P=NP). Nevertheless, with proper optimization formulation, we develop offline and online algorithms that can achieve excellent performance as demonstrated by both flow-level and packet-level simulations.

Acknowledgements

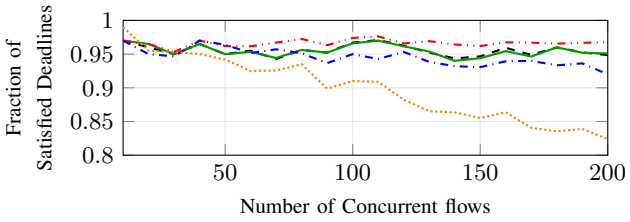
This work was supported by the National Science Foundation (NSF) under Grant No. CPS-1544761, and by the Huawei HIRP program.



(a) The mean of the generated deadline is 2 ms.

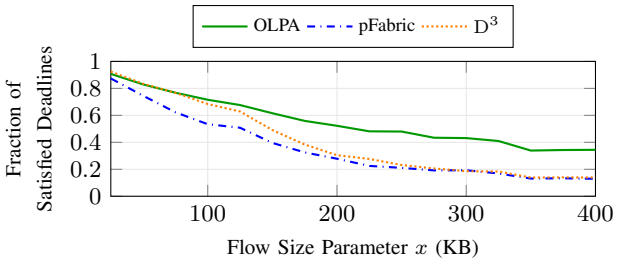


(b) The mean of the generated deadline is 3 ms.

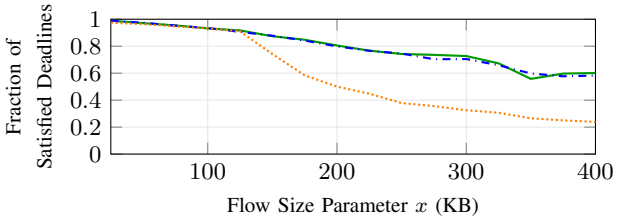


(c) The mean of the generated deadline is 4 ms.

Fig. 9: The fraction of satisfied deadlines under request burst scenario.

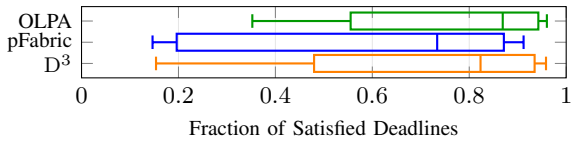


(a) $q = 1$.

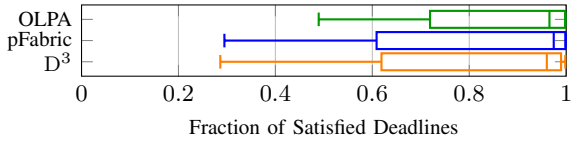


(b) $q = 2$.

Fig. 10: The median of the fraction of satisfied deadlines under stochastic demand scenario with the sizes of the flows uniformly distributed between 0 and x KB.



(a) $q = 1$.



(b) $q = 2$.

Fig. 11: The $1^{\text{st}} - 5^{\text{th}} - 50^{\text{th}} - 95^{\text{th}} - 99^{\text{th}}$ percentiles of the fraction of satisfied deadlines.

REFERENCES

[1] M. Alizadeh, S. Yang, M. Sharif, S. Katti, N. McKeown, B. Prabhakar, and S. Shenker. "pfabric: Minimal near-optimal datacenter transport," *ACM SIGCOMM Computer Communication Review*, 43(4), 435-446, 2013.

[2] N. Dukkipati, and N. McKeown. "Why flow-completion time is the right metric for congestion control," *ACM SIGCOMM Computer Communication Review*, 36(1), pp.59-62, 2006.

[3] C. Wilson, H. Ballani, T. Karagiannis, and A. Rowtron. "Better never than late: Meeting deadlines in datacenter networks," *in ACM SIGCOMM Computer Communication Review*, Vol. 41, No. 4, pp. 50-61, 2011.

[4] C.Y. Hong, M. Caesar, and P. Godfrey. "Finishing flows quickly with preemptive scheduling," *ACM SIGCOMM Computer Communication Review* 42, no. 4 (2012): 127-138, 2012.

[5] P. Brucker. "Scheduling algorithms." Vol. 3. Berlin: Springer, 2007.

[6] A. Munir, I. A. Qazi, Z. A. Uzmi, A. Mushtaq, S. N. Ismail, M. S. Iqbal, and B. Khan. "Minimizing flow completion times in data centers," *in INFOCOM, 2013 Proceedings IEEE* (pp. 2157-2165). IEEE, 2013.

[7] H. Xu, and B. Li. "RepFlow: Minimizing flow completion times with replicated flows in data centers," *in INFOCOM, 2014 Proceedings IEEE* (pp. 1581-1589). IEEE, 2014.

[8] N. Dukkipati, N. McKeown, and A. G. Fraser. "RCP-AC: Congestion control to make flows complete quickly in any environment," *in INFOCOM 2006. 25th IEEE International Conference on Computer Communications. Proceedings* (pp. 1-5). IEEE, 2006.

[9] N. Dukkipati. "Rate Control Protocol (RCP): Congestion control to make flows complete quickly." (Doctoral dissertation, Stanford University), 2007.

[10] M. Alizadeh, A. Greenberg, D. Maltz, J. Padhye, P. Patel, B. Prabhakar, S. Sengupta, and M. Sridharan. "Data center tcp (dctcp)," *ACM SIGCOMM computer communication review* 41, no. 4 (2011): 63-74, 2011.

[11] I. F. Akyildiz, A. Lee, P. Wang, M. Luo, and W. Chou. "A roadmap for traffic engineering in SDN-OpenFlow networks," *Computer Networks*, 71, 1-30, 2014.

[12] D. Zats, T. Das, P. Mohan, D. Borthakur, and R. Katz. "DeTail: reducing the flow completion time tail in datacenter networks," *ACM SIGCOMM Computer Communication Review*, 42(4), 139-150, 2012.

[13] V. Raghunathan, V. Borkar, M. Cao, and P. R. Kumar. "Index policies for real-time multicast scheduling for wireless broadcast systems," *in INFOCOM 2008. The 27th Conference on Computer Communications*. IEEE, 2008.

[14] S. Chen, L. Tong, and T. He. "Optimal deadline scheduling with commitment," *in Communication, Control, and Computing (Allerton), 2011 49th Annual Allerton Conference on* (pp. 111-118). IEEE, 2011.

[15] B. Lucier, I. Menache, J. S. Naor, and J. Yaniv. "Efficient online scheduling for deadline-sensitive jobs," *in Proceedings of the twenty-fifth annual ACM symposium on Parallelism in algorithms and architectures* (pp. 305-314). ACM, 2013.

[16] B. McCarl, and T. Spreen. "Applied mathematical programming using algebraic systems," 1997.

[17] COIN-OR projects. [Online]. Available: http://www.coin-or.org/projects/