Minimal Per-Flow Backlog Bounds at an Aggregate FIFO Server Under Piecewise-Linear Arrival Curves

Lukas Wildberger(✉), Anja Hamscher, and Jens B. Schmitt

DISCO Lab, RPTU Kaiserslautern-Landau, 67663 Kaiserslautern, Germany
lukas.wildberger@cs.rptu.de, {hamscher,jschmitt}@cs.uni-kl.de

Abstract. Network Calculus (NC) is a versatile methodology based on min-plus algebra to derive worst-case *per-flow* performance bounds in networked systems with many concurrent flows. In particular, NC can analyze many scheduling disciplines; yet, somewhat surprisingly, an aggregate FIFO server is a notoriously hard case due to its min-plus *non-linearity*. A resort is to represent the FIFO residual service by a family of functions with a free parameter instead of just a single curve. For simple token-bucket arrival curves, literature provides optimal choices for that free parameter to minimize delay and backlog bounds. In this paper, we tackle the challenge of more general arrival curves than just token buckets. In particular, we derive residual service curves resulting in minimal backlog bounds for general piecewise-linear arrival curves. To that end, we first show that a backlog bound can always be calculated at a breakpoint of either the arrival curve of the flow of interest or its residual service curve. Further, we define a set of curves that characterize the backlog for a fixed breakpoint, depending on the free parameter of the residual service curve. We show that the backlog-minimizing residual service curve family parameter corresponds to the largest intersection of those curves with the arrival curve. In more complex scenarios finding this largest intersection can become inefficient as the search space grows in the number of flows. Therefore, we present an efficient heuristic that finds, in many cases, the optimal parameter or at least a close conservative approximation. This heuristic is evaluated in terms of accuracy and execution time. Finally, we utilize these backlog-minimizing residual service curves to enhance the DiscoDNC tool and observe considerable reductions in the corresponding backlog bounds.

Keywords: Network Calculus · FIFO Scheduling · Backlog Bound

1 Introduction

First-In First-Out (FIFO) is a popular scheduling policy for networked systems due to its simplicity and low cost of implementation. In various network analysis methods, it is an interesting policy to analyze. One such analysis method is

P. Prabhakar and A. Vandin (Eds.): QEST+FORMATS 2025, LNCS 16143, pp. 157–175, 2026.
https://doi.org/10.1007/978-3-032-05792-1_9

Network Calculus (NC) [5], which is a versatile methodology for deriving performance bounds in networked systems [1,6,7]. In particular, we are interested in the derivation of backlog bounds, which are important for sizing queues or buffers appropriately. It is straightforward to obtain a tight backlog bound at a FIFO node when considering all incoming traffic as a aggregate flow [7], such as in switches with a shared queue. A challenge arises when we are instead interested in a *per-flow* backlog bound. This means that we want to analyze how much data can queue up for an individual flow, rather than analyzing the aggregate traffic of the flows. This setting is considered for separate FIFO queues like in input-buffered switches with virtual output queues (VOQ) [10], or big data processing such as in Hadoop, where a job queue is fed data from a distributed file system through switches [13].

With NC, per-flow backlog bounds can be calculated using a so-called *residual service curve*, which represents the residual service that is exclusively available to a specific flow of interest (foi). However, due to the min-plus non-linearity of FIFO-scheduled systems [9], computing a residual service curve is hard. Defining a *family of functions* with a free parameter θ, rather than a single FIFO residual service curve, is a way of dealing with this. The family of FIFO residual service curves [6] is given by

$$\beta^1_\theta(t) = [\beta(t) - \alpha_2(t-\theta)]^+ \wedge \delta_\theta(t), \text{with } \theta \geq 0,$$

where β models the service available to the traffic aggregate and α_2 is an upper bound on the traffic of all the cross flows that are multiplexed into the aggregate together with the foi. Each value of θ results in a different residual service curve β^1_θ. Figure 1 shows such different β^1_θ, illustrating its dependence on the chosen θ. This has an effect on the resulting backlog bound that is obtained (drawn in Fig. 1 as vertical lines).

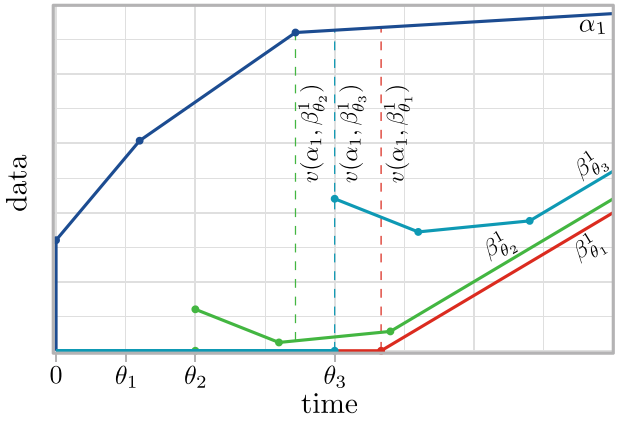


Fig. 1. FIFO residual service curves for different values of θ, and resulting backlog bounds.

Our goal is to obtain the smallest possible backlog bound. Due to β_θ^1 and the backlog bound changing depending on the chosen θ, there exists an optimal value of θ that provides the smallest backlog bound over all possible residual curves. In fact, a closed form for this θ exists when only considering token-bucket-constrained arrivals [1,7]. Yet, for more general functions such as piecewise linear (PWL) curves, to the best of our knowledge there exist no per-flow backlog bound results in literature regarding the calculation of θ values. Somewhat the only exception is [4], where an output bound for PWL arrival curves has been derived, yet without using NC. Nevertheless, it is interesting as the burst term of an output bound is also a bound on the backlog. The analysis is however restricted to the case of a constant rate server, whereas we deal with PWL service curves. Hence, in this paper we are considering PWL curves and are interested in closed form for θ that minimizes the backlog bound for such curves. To that end, we make the following contributions in this paper:

- In Sect. 4, we derive an exact method to find the backlog-minimizing θ value for PWL-constrained arrival and service curves.
- An efficient heuristic that determines the backlog-minimizing parameter θ in most cases is presented in Sect. 5. We evaluate its accuracy and execution time in comparison to the exact method.
- Finally, we show in Sect. 6 that the parameter θ, derived using our new methods, poses a significant accuracy improvement for backlog bounds computed by the DiscoDNC tool over its current default setting.

2 Network Calculus Background

Let $\mathbb{R}^+$ be the set of non-negative real numbers. $\mathcal{F} := \{f : \mathbb{R}^+ \to \mathbb{R} \cup \{+\infty\}\}$ is the set of (min, plus) functions. Based on $\mathcal{F}$, we let $\mathcal{F}^\uparrow$ be the set of non-decreasing functions $f \in \mathcal{F}$, and $\mathcal{F}_0^\uparrow$ be the set of functions in $\mathcal{F}^\uparrow$ with $f(0) = 0$.

Definition 1 (Basic Operators [3])**.** *Let $f, g \in \mathcal{F}$. The min-plus convolution of f and g is defined as $f \otimes g(t) := \inf_{0 \le s \le t}\{f(t-s) + g(s)\}$. The (max-plus) deconvolution is defined as $f \overline{\oslash} g(t) := \inf_{s \ge 0}\{f(t+s) - g(s)\}$. We define the $\wedge$ operator as $f \wedge g(t) := \min\{f(t), g(t)\}$.*

The *impulse function* $\delta_T(t)$ is defined as $\delta_T(t) = \infty$, if $t > T$ and 0 otherwise. The *indicator function* $\mathbb{1}_A$ is defined as $\mathbb{1}_A = 1$, if A is true and 0 otherwise. For a given $\beta \in \mathcal{F}$, the *lower non-decreasing closure* is defined as the largest non-decreasing function with $\beta_\downarrow \le \beta$, given by $\beta_\downarrow := \beta \overline{\oslash} 0$ [3, p. 107].

Definition 2 (Pseudo-Inverse [3])**.** *Let $f \in \mathcal{F}$ be a non-negative and non-decreasing function. The pseudo-inverse f^{-1} is for all $x \in \mathbb{R}^+$ given by*

$$f^{-1}(x) = \inf\{t \mid f(t) \ge x\}.$$

Next, we define various notions that are used to model a network and derive its performance bounds. Let $A, D \in \mathcal{F}_0^{\uparrow}$ be the *cumulative arrival* and *departure process* of a flow in the network, assuming causality $A \geq D$. Furthermore, we assume the system to be lossless. We define the most important performance measures for such a system:

Definition 3 (Virtual Delay at Time t)**.** *The* virtual delay *of data arriving at system $\mathcal{S}$ at time t is the time until this data would be served, assuming FIFO-per-flow order of service,*

$$d_{A,D}(t) = \inf\{d \geq 0 : A(t) \leq D(t+d)\} = D^{-1}(A(t)) - t \tag{1}$$

Definition 4 (Backlog at Time t)**.** *The* backlog *of system $\mathcal{S}$ at time t is the vertical distance between arrival process A and departure process D at time t,*

$$q_{A,D}(t) \coloneqq A(t) - D(t). \tag{2}$$

Arrival and service curves are central for the performance analysis using NC.

Definition 5 (Arrival Curve)**.** *Let $\alpha \in \mathcal{F}_0^{\uparrow}$. We say that α is an* arrival curve *for arrival process A if it holds for all $0 \leq s \leq t$ that*

$$A(t) - A(s) \leq \alpha(t-s) \Longleftrightarrow A = A \otimes \alpha.$$

An example is a *token-bucket* arrival curve $\gamma_{r,b}(t) = b + rt$ if $t > 0$, $\gamma_{r,b}(0) = 0$.

Definition 6 (Service Curve)**.** *Let a flow with arrival process A and departure process D traverse a system $\mathcal{S}$. The system offers a* min-plus service curve *β to the flow if $\beta \in \mathcal{F}$ and it holds for all $t \geq 0$ that*

$$D(t) \geq A \otimes \beta\ (t) = \inf_{0 \leq s \leq t} \{A(t-s) + \beta(s)\}.$$

An example is a *rate-latency* curve $\beta_{R,T}(t) \coloneqq R \cdot [t-T]^+$, $[x]^+ \coloneqq \max\{x, 0\}$. We define two characteristic distances between functions.

Definition 7. *Let $f, g \in \mathcal{F}$. The horizontal deviation between f and g is*

$$h(f,g) \coloneqq \sup_{t \geq 0}\{\inf\{d \geq 0 \mid f(t) \leq g(t+d)\}\}$$

and the vertical deviation *between f and g is*

$$v(f,g) \coloneqq \sup_{t \geq 0} \{f(t) - g(t)\}.$$

Using these concepts, one can derive a backlog bound [3, p. 115], [7, p. 118].

Theorem 8 (Backlog Bound)**.** *Assume a single flow with arrival process A, with arrival curve $\alpha \in \mathcal{F}_0^{\uparrow}$, and departure process D traverses a system $\mathcal{S}$. Let the system $\mathcal{S}$ offer a service curve $\beta \in \mathcal{F}_0^{\uparrow}$. The backlog $q(t)$ satisfies for all t*

$$q_{A,D}(t) \leq v(\alpha, \beta).$$

A FIFO residual service curve can be calculated as follows.

Theorem 9 (Residual Service Curve for FIFO [6]). *Let $t \geq 0$. Consider a system $\mathcal{S}$ that multiplexes two flows f_1 and f_2 using FIFO scheduling. The arrivals of f_2, A_2, are constrained by α_2. Further, assume that $\mathcal{S}$ guarantees a service curve β to the aggregate of the flows. Then, for any $\theta \geq 0$, the residual service of f_1 is*

$$\beta_\theta^1(t) = [\beta(t) - \alpha_2(t-\theta)]^+ \wedge \delta_\theta(t) \tag{3}$$

Definition 10 (PWL Concave Normal Form [3]). *Let r_i, $b_i \in \mathbb{R}^+$ and set $\gamma_i = \gamma_{r_i,b_i}$. The piecewise linear concave function $f = \min\{\gamma_i\}$ is said to be in normal form, if γ_i are sorted by a decreasing rate and no γ_i can be removed without modifying the minimum:*

$$i < j \Rightarrow r_i > r_j, \tag{4}$$

$$\forall i, \exists t > 0, \forall j \neq i, \gamma_i(t) < \gamma_j(t). \tag{5}$$

If $f = \min\{\gamma_i\}, i \in \{1, \ldots, n\}$ is in normal form, then there is a sequence of a_i of respective intersections of the linear functions γ_i and γ_{i+1}. These intersections, denoted by a_i, are also called breakpoints of f.

Definition 11 (PWL Convex Normal Form [3]). *Let R_i, $T_i \in \mathbb{R}^+$ and set $\beta_i = \beta_{R_i,T_i}$. The PWL convex function $f = \max\{\beta_i\}$ is said to be in normal form, if β_i are sorted by an increasing rate and no β_i can be removed without modifying the maximum:*

$$i < j \Rightarrow R_i < R_j \ \wedge \ \forall i, \exists t > 0, \forall j \neq i, \beta_i(t) > \beta_j(t).$$

If $f = \max\{\beta_i\}, i \in \{1, \ldots, n\}$ is in normal form, then there is a sequence of s_i of respective intersections of the linear functions β_i and β_{i+1}. These intersections, denoted by s_i, are also called breakpoints of f.

Definition 12. *Let the linear segment of a given PWL (concave or convex) function f at time t be called f^t, with $f^t = \beta_{R,T}$ or $f^t = \gamma_{r,b}$. The rate, r or R, and the y-axis intercept b or x-axis intercept T of this linear segment is then also referred to as r^t, R^t, b^t, T^t.*

Definition 13. *Let α be a PWL concave curve in normal form and β a PWL convex curve in normal form. Let A and B be the sets of breakpoints of α and β, respectively. Let I_A be the set of all breakpoints mapped to α, defined as $I_A \coloneqq A \ \cup \ \{\alpha^{-1}(\beta(s)) \mid s \in B\}$. Then the first point in time for which the corresponding rate of α is less than or equal to the corresponding rate of β, mapped to α, is called $a^*_{\alpha,\beta}$. We define $a^*_{\alpha,\beta}$ as follows:*

$$a^*_{\alpha,\beta} \coloneqq \min\{i \in I_A : r^i \leq R^{\beta^{-1}(\alpha(i))}\}.$$

3 System Model

In this section, we introduce the system setting considered in this paper. We also discuss the challenge associated with this setting and how to deal with it.

Figure 2 illustrates the system model under investigation. Here, each (distributed) flow (or job) $f_1, \ldots, f_n$ is associated with its own flow queue, each with its own size. These flows send their requests into a single (task) queue, which ensures the FIFO order between the individual flows when processed by system $\mathcal{S}$, for which we know a service curve β. The central question of our work now is: how to appropriately size the individual flow queues, or, in other words, how to calculate per-flow backlog bounds.

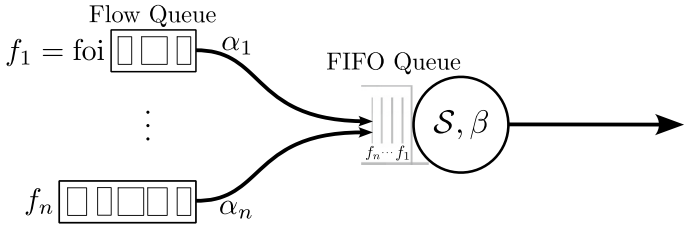


Fig. 2. Distributed per-flow queues served by single FIFO system.

We point out that per-flow queues in total generally require more space than an aggregate shared queue, i.e., we incur a *segregation penalty*. The extent of that penalty needs to weighed against advantages from the distributed setting as given in Fig. 2. We come back to that issue in Sect. 6.

Throughout the following sections, we assume that the flows $f_1 = \text{foi}, \ldots, f_n$ are constrained by PWL concave arrival curves $\alpha_1, \ldots, \alpha_n \in \mathcal{F}$. We aggregate all cross flows arrival curves $\alpha_2, \ldots, \alpha_n$, and simply call it α_2. The system $\mathcal{S}$ that multiplexes the flows according to FIFO, offers a PWL convex service curve $\beta \in \mathcal{F}$ to the flow aggregate.

When applying NC to compute per-flow backlog bounds under PWL arrival and service curves, an issue arises: the residual service curve β_θ^1 may contain a finite number of segments with negative slopes, which violates the non-decreasing property. While some literature formally requires that $\beta_\theta^1 \in \mathcal{F}_0^\uparrow$ [7], this constraint is not always reflected in the conditions of the corresponding theorems. Anyway, the following lemma states that this violation does not impact the calculation of the vertical deviation (which we need to compute backlog bounds), since we can make use of the lower non-decreasing closure.

Lemma 14. *Let $\alpha \in \mathcal{F}$ and $\beta \in \mathcal{F}$ be given. Then it holds that*

$$v(\alpha, \beta) = v(\alpha, \beta_\downarrow). \tag{6}$$

The proof for this and all subsequent lemmas can be found in the arXiv version of this paper [14].

4 Derivation of the θ Parameter for Minimal Per-Flow Backlog Bounds

In this section, we derive the value θ_{opt} that minimizes per-flow backlog bounds. In order to derive such a value of θ, we begin by formulating the problem mathematically as

$$\theta_{\text{opt}} = \arg\min_{\theta \geq 0} \{v\big(\alpha_1, \beta_\theta^1\big)\}. \tag{7}$$

In the following, we will transform this problem step by step until we are able to determine a solution. At first, in order to constrain the choice of θ, we make use of the following lemma:

Lemma 15. *Let α_1 and α_2 be PWL concave arrival curves and let β be a PWL convex service curve under FIFO multiplexing. Let $h(\alpha_2, \beta)$ be the horizontal deviation of α_2 and β. Then, it holds for $0 \leq \theta \leq h(\alpha_2, \beta)$ that*

$$v(\alpha_1, \beta_\theta^1) \geq v(\alpha_1, \beta_{h(\alpha_2,\beta)}^1).$$

Note that for token-bucket arrival curves and rate-latency service curves, the result of Lemma 15 has already been stated in [1,3,7,8].

Based on Lemma 15, we conclude that the search space for θ can be restricted to $\theta \geq h(\alpha_2, \beta)$. This allows us to transform the problem in Eq. (7) to

$$\theta_{\text{opt}} = \arg\min_{\theta \geq h(\alpha_2,\beta)} \{v\big(\alpha_1, \beta_\theta^1\big)\}.$$

To proceed, we define the family of functions $v_t(\theta)$ as the vertical distances between α_1 and β_θ^1 at an arbitrary but fixed time t:

$$\begin{aligned} v_t(\theta) &:= \begin{cases} \alpha_1(t) - \beta_\theta^1(t), & \text{if } h(\alpha_2, \beta) \leq \theta < t, \\ \alpha_1(\theta), & \text{if } \theta \geq t > h(\alpha_2, \beta), \end{cases} \\ &= \begin{cases} \alpha_1(t) - \beta(t) + \alpha_2(t - \theta), & \text{if } h(\alpha_2, \beta) \leq \theta < t, \\ \alpha_1(\theta), & \text{if } \theta \geq t > h(\alpha_2, \beta). \end{cases} \end{aligned}$$

Here, we have used that $\beta_\theta^1(t) = \beta(t) + \alpha_2(t - \theta)$, for $h(\alpha_2, \beta) \leq \theta < t$. Particularly, by definition we have $\beta_\theta^1 = [\beta(t) - \alpha_2(t - \theta)]^+ \wedge \delta_\theta(t)$. Since for $\theta < t$ it holds that $\delta_\theta(t) = 0$, and because $h(\alpha_2, \beta) \leq \theta$ ensures that $\beta(t) \geq \alpha_2(t - \theta)$ for all t (according to Definition 7), both the positive part and $\delta_\theta(t)$ can be omitted.

With this, the original problem in Eq. (7) can now be rewritten as:

$$\begin{aligned} \theta_{\text{opt}} &= \arg\min_{\theta \geq h(\alpha_2,\beta)} \{v\big(\alpha_1, \beta_\theta^1\big)\} \\ &= \arg\min_{\theta \geq h(\alpha_2,\beta)} \{\sup_{t \geq 0}\{\alpha_1(t) - \beta_\theta^1(t)\}\} \\ &= \arg\min_{\theta \geq h(\alpha_2,\beta)} \{\sup_{t \geq 0}\{v_t(\theta)\}\} \end{aligned} \tag{8}$$

Here, the supremum is over all $t \geq 0$. The following lemma will be instrumental to restrict the relevant values of t to a finite set.

Lemma 16. *Let α be a PWL concave function and β be a PWL convex function. Let A and B be the set of breakpoints of α and β, respectively. Then, the vertical deviation of α and β can always be calculated at some time $t \in A \cup B$.*

Remark 17. *The result of Lemma 16 also applies to the horizontal deviation.*

In the following, we denote the set of breakpoints of α_1, α_2, and β as A_1, A_2, and B, respectively. Lemma 16 ensures that only A_1 and the set of breakpoints of β^1_θ, induced by $A_2 \cup B$, need to be considered. A closer examination of these breakpoints reveals structural differences. In particular, the breakpoints of β^1_θ need further attention, as they depend on θ. In contrast, the breakpoints of α_1, A_1, remain invariant with respect to θ. We call the set of breakpoints A_1 *absolute time set* of α_1. The absolute time set of the breakpoints of β^1_θ, including the shift by θ, is denoted by $\mathcal{T}^{\text{abs}}_{\beta^1_\theta}$. The set of breakpoints of β^1_θ, without being shifted by θ, is called *relative time set* and is given by

$$\mathcal{T}^{\text{rel}}_{\beta^1_\theta} = A_2 \cup B.$$

The relative time set $\mathcal{T}^{\text{rel}}_{\beta^1_\theta}$ allows us the usage of the breakpoints without having a dependency on θ. In the following, we derive from this relative time set an absolute time set of β^1_θ, which is also no longer dependent on θ. This is achieved by determining a single value of θ for each $t \in \mathcal{T}^{\text{rel}}_{\beta^1_\theta}$. Formally, for each $t \in \mathcal{T}^{\text{rel}}_{\beta^1_\theta}$, the corresponding value θ_t is defined as the solution to the following equation:

$$\alpha_1(\theta_t) = \alpha_1(t) - \beta^1_{\theta_t}(t). \tag{9}$$

As established in Lemma 16, only the breakpoints of the curves are relevant for computing the backlog bound. So we consider each relative point in time $t \in \mathcal{T}^{\text{rel}}_{\beta^1_\theta}$, *assuming* that it would be the one for which the vertical deviation is taken on. Then we obtain θ_t as the solution of Eq. (9), that minimizes this backlog bound under this assumption. In order to justify Eq. (9), let us consider an arbitrary, but fixed time $t \in \mathcal{T}^{\text{rel}}_{\beta^1_\theta}$. At this time t, the vertical distance between the two functions is given by $\alpha_1(t) - \beta^1_\theta(t)$, for $\theta < t$. This distance decreases as θ increases, so higher values of θ appear to be better for minimizing the backlog bound. However, this perspective neglects an important factor—namely, the behavior of the vertical distance at time θ. Specifically, as θ increases, the vertical distance at time θ, given by $\alpha_1(\theta)$, also increases. Consequently, selecting a larger value of θ to reduce the distance at time t simultaneously results in a larger distance at time θ. The optimal balance of this trade-off is given by θ_t as the solution of Eq. (9). For $\theta_t < t$, the respective θ_t can be derived as follows:

$$\begin{aligned}
& \alpha_1(\theta_t) = \alpha_1(t+\theta_t) - \beta^1_{\theta_t}(t+\theta_t) \\
\Leftrightarrow \quad & \alpha_1(\theta_t) = \alpha_1(t+\theta_t) - \beta(t+\theta_t) + \alpha_2(t) \\
\Leftrightarrow \quad & \beta(t+\theta_t) - \alpha_1(t+\theta_t) + \alpha_1(\theta_t) = \alpha_2(t) \\
\Leftrightarrow \quad & \beta(t+\theta_t) - \alpha_1(t+\theta_t) + \alpha_1 \otimes \delta_t(t+\theta_t) = \alpha_2(t) \\
\Leftrightarrow \quad & (\beta - \alpha_1 + (\alpha_1 \otimes \delta_t))(t+\theta_t) = \alpha_2(t),
\end{aligned} \tag{10}$$

$$\Rightarrow \theta_t = d_{\beta-\alpha_1+(\alpha_1\otimes\delta_t),\alpha_2}(t) \tag{11}$$

$$= ([\beta - \alpha_1 + (\alpha_1 \otimes \delta_t)]^+)^{-1}(\alpha_2(t)) - t \tag{12}$$

where we used for Eq. (11) that the horizontal distance is given by the shift θ_t in Eq. (10) (see also Eq. (1)). We apply the positive part in Eq. (12), since the pseudo-inverse requires wide-sense increasing functions and the horizontal distance for a positive function α_2 only requires non-negative functions.

Utilizing θ_t, the absolute time set is now explicitly given as

$$\mathcal{T}^{\text{abs}}_{\beta^1_\theta} = \mathcal{T}^{\text{rel}}_{\beta^1_\theta} + \theta_t = \{a_i^2 + \theta_{a_i^2} \mid a_i^2 \in A_2\} \cup \{s_i + \theta_{s_i} \mid s_i \in B\}.$$

Combining this with A_1, we obtain the complete set of absolute breakpoint times:

$$\mathcal{T} = A_1 \cup \mathcal{T}^{\text{abs}}_{\beta^1_\theta}.$$

This reduces the number of $v_t(\theta)$ curves to be evaluated from an uncountable set (for $t \geq 0$) to a finite set (for $t \in \mathcal{T}$), which can be computed given the parameters of the arrival and service curves. Since t is now an element of the finite set $\mathcal{T}$, we can replace the supremum by a maximum in Eq. (8), leading to the following transformed problem:

$$\theta_{\text{opt}} = \underset{\theta \geq h(\alpha_2,\beta)}{\arg\min} \{\max_{t\in\mathcal{T}}\{v_t(\theta)\}\}.$$

We can further restrict the domain of θ. According to Lemma 16, the backlog bound can always be calculated at a breakpoint of either α_1 or β^1_θ. Moreover, $\forall t \in \mathcal{T}$ and $\theta > t_{\max}$, with $t_{\max} = \max \mathcal{T}$, it holds that $v_t(\theta) = \alpha_1(\theta)$, which increases as θ increases. Hence, the backlog-minimizing value of θ cannot lie beyond $t_{\max}$. Additionally, since $\max_{t\in\mathcal{T}}\{v_t(\theta)\} = \max\{\max_{t\in\mathcal{T}}\{v_t(\theta)\}, \alpha_1(\theta)\}$ trivially holds, as $\alpha_1(\theta) = v_0(\theta) \in \{v_t(\theta) \mid t \in \mathcal{T}\}$, the problem can be transformed into

$$\theta_{\text{opt}} = \underset{\theta\in[h(\alpha_2,\beta),t_{\max}]}{\arg\min} \{\max\{\max_{t\in\mathcal{T}}\{v_t(\theta)\}, \alpha_1(\theta)\}\}. \tag{13}$$

Figure 3 gives a graphical representation of Eq. (13). Here, two v_t curves, v_{t_1} and v_{t_2} are plotted against $\alpha_1(\theta)$. Each v_t represents the vertical distance between α_1 and β^1_θ at a specific breakpoint $t \in \mathcal{T}$. According to Lemma 16, only these breakpoints need to be considered to compute backlog bounds. As θ varies, the vertical distances $v_t(\theta)$ decrease at some of these breakpoints and increase at others. In order to minimize the overall backlog bound, we aim to find the value of θ that balances this trade-off such that the largest vertical distance, at all breakpoints, is minimized. Formally, this means that this (optimal) value of θ (θ_{opt}), marked by a red dot, is obtained at the intersection of $\max_{t\in\mathcal{T}}\{v_t(\theta)\}$ and $\alpha_1(\theta)$.

It is useful to gain further insight into the problem, and examine the behavior of the two functions, $\max_{t\in\mathcal{T}}\{v_t(\theta)\}$ and $\alpha_1(\theta)$, at the boundary values of the interval for θ. The following lemma characterizes the relation between these functions at the interval's endpoints.

Lemma 18. *Let* $\max_{t\in\mathcal{T}}\{v_t(\theta)\}$ *and* $\alpha_1(\theta)$ *be defined as above. For the endpoints of the given interval* $[h(\alpha_2,\beta), t_{\max}]$ *it holds that*

$$\alpha_1(h(\alpha_2,\beta)) \leq \max_{t\in\mathcal{T}}\{v_t(h(\alpha_2,\beta))\} \text{ and } \alpha_1(t_{\max}) = \max_{t\in\mathcal{T}}\{v_t(t_{\max})\}.$$

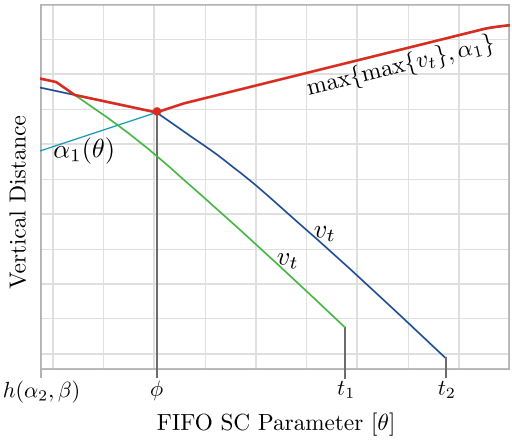


Fig. 3. Vertical distances $v_t(\theta)$ and $\alpha_1(\theta)$, for $\theta \geq h(\alpha_2,\beta)$.

To address the problem given by Eq. (13), we introduce the concept of a *first intersection* between $\max_{t\in\mathcal{T}}\{v_t(\theta)\}$ and $\alpha_1(\theta)$. This is essential for identifying the exact point in time where the function $\max_{t\in\mathcal{T}}\{v_t(\theta)\}$ drops onto the function $\alpha_1(\theta)$. We say that $\max_{t\in\mathcal{T}}\{v_t(\theta)\}$ and $\alpha_1(\theta)$ *first intersect* at

$$\phi = \min\{x \mid \alpha_1(x) = \max_{t\in\mathcal{T}}\{v_t(x)\}\}.$$

It is important to note that a first intersection between $\max_{t\in\mathcal{T}}\{v_t(\theta)\}$ and $\alpha_1(\theta)$ always exists. According to Lemma 18, these two functions are guaranteed to intersect at least at $\theta = t_{\max}$. Furthermore, by definition, every $v_t(\theta)$ curve, with $t \in \mathcal{T}$, is strictly decreasing for $\theta < t$ and strictly increasing for $\theta \geq t$. If $\max_{t\in\mathcal{T}}\{v_t(\theta)\}$ is taken at the respective strictly decreasing parts of the $v_t(\theta)$ curves, then the maximum function $\max_{t\in\mathcal{T}}\{v_t(\theta)\}$ itself is also strictly decreasing. The $\max_{t\in\mathcal{T}}\{v_t(\theta)\}$ is taken on the strictly decreasing parts until the maximum function $\max_{t\in\mathcal{T}}\{v_t(\theta)\}$ eventually equals $\alpha_1(\theta)$, which is strictly increasing. This is exactly the first intersection ϕ of $\max_{t\in\mathcal{T}}\{v_t(\theta)\}$ and $\alpha_1(\theta)$, after which $\max_{t\in\mathcal{T}}\{v_t(\theta)\} = \alpha_1(\theta)$ holds. So we conclude that $\max_{t\in\mathcal{T}}\{v_t(\theta)\}$ is strictly decreasing for $\theta < \phi$ and strictly increasing for $\theta \geq \phi$. Note that if $\max_{t\in\mathcal{T}}\{v_t(\theta)\}$ is directly assumed to be on the strictly increasing part of the $v_t(\theta)$ curves, then the first intersection is exactly at $h(\alpha_2,\beta)$.

The following theorem provides the solution to the problem stated in Eq. (13), given by the first intersection ϕ.

Theorem 19. *Let* $\max_{t\in\mathcal{T}}\{v_t(\theta)\}$ *and* $\alpha_1(\theta)$ *be defined as above. Further, let the first intersection of* $\max_{t\in\mathcal{T}}\{v_t(\theta)\}$ *and* $\alpha_1(\theta)$ *be at* ϕ*. Then, it holds*

$$\theta_{\text{opt}} = \underset{\theta\in[h(\alpha_2,\beta),t_{\max}]}{\arg\min}\{\max\{\max_{t\in\mathcal{T}}\{v_t(\theta)\},\alpha_1(\theta)\}\} = \phi.$$

Proof. According to Lemma 18 it holds that $\alpha_1(h(\alpha_2,\beta)) \leq \max_{t\in\mathcal{T}}\{v_t(h(\alpha_2,\beta))\}$ and $\alpha_1(t_{\max}) = \max_{t\in\mathcal{T}}\{v_t(t_{\max})\}$ so the following also holds:

$$\forall\ \theta < \phi : \alpha_1(\theta) \leq \max_{t\in\mathcal{T}}\{v_t(\theta)\}\ \wedge\ \forall\ \theta \geq \phi : \alpha_1(\theta) = \max_{t\in\mathcal{T}}\{v_t(\theta)\}.$$

So the maximum function can be described as follows:

$$\max\{\max_{t\in\mathcal{T}}\{v_t(\theta)\},\alpha_1(\theta)\} = \begin{cases}\max_{t\in\mathcal{T}}\{v_t(\theta)\}, & \text{if } \theta < \phi,\\ \alpha_1(\theta), & \text{if } \theta \geq \phi.\end{cases}$$

Suppose $\arg\min_{\theta\in[h(\alpha_2,\beta),t_{\max}]}\{\max\{\max_{t\in\mathcal{T}}\{v_t(\theta)\},\alpha_1(\theta)\}\}$ is assumed at ϕ', with $\phi' \in [h(\alpha_2,\beta),t_{\max}]$ and $\phi \neq \phi'$. So

$$\underset{\theta\in[h(\alpha_2,\beta),t_{\max}]}{\arg\min}\{\max\{\max_{t\in\mathcal{T}}\{v_t(\theta)\},\alpha_1(\theta)\}\} = \phi'$$

should hold. Consider two cases: $\phi' < \phi$ and $\phi' > \phi$.
Case I ($\phi' < \phi$): Since $\phi' < \phi$ and $\max\{\max_{t\in\mathcal{T}}\{v_t(\theta)\},\alpha_1(\theta)\} = \max_{t\in\mathcal{T}}\{v_t(\theta)\}$ is strictly decreasing for $\theta < \phi$, it holds that:

$$\max\{\max_{t\in\mathcal{T}}\{v_t(\phi')\},\alpha_1(\phi')\} > \max\{\max_{t\in\mathcal{T}}\{v_t(\phi)\},\alpha_1(\phi)\}. \quad \lightning$$

Case II ($\phi' > \phi$): Since $\phi' > \phi$ and $\max\{\max_{t\in\mathcal{T}}\{v_t(\theta)\},\alpha_1(\theta)\} = \alpha_1(\theta)$ is strictly increasing for $\theta \geq \phi$, it holds that:

$$\max\{\max_{t\in\mathcal{T}}\{v_t(\phi')\},\alpha_1(\phi')\} > \max\{\max_{t\in\mathcal{T}}\{v_t(\phi)\},\alpha_1(\phi)\}. \quad \lightning$$

So $\theta_{\text{opt}} = \arg\min_{\theta\in[h(\alpha_2,\beta),t_{\max}]}\{\max\{\max_{t\in\mathcal{T}}\{v_t(\theta)\},\alpha_1(\theta)\}\} = \phi$ holds. □

Remark 20. *Note that the first intersection of* $\max_{t\in\mathcal{T}}\{v_t(\theta)\}$ *and* $\alpha_1(\theta)$ *is also the last of all intersections of* $v_t(\theta)$ *curves,* $t \in \mathcal{T}$*, with* $\alpha_1(\theta)$*. See again Fig. 3.*

5 Efficient Calculation of Near-Optimal Backlog Bounds

In the following, we expand on the calculation of the FIFO SC parameter θ_{opt} and its corresponding minimal backlog bound $q_{\min}$. We strive for an efficient calculation of backlog bounds that are still near to the minimal ones from the previous section using Theorem 19. In particular, according to Remark 20, we calculate the v_t curves of all $t \in \mathcal{T}$, calculate their respective intersection points with α_1, then determine ϕ as the maximum of all intersections to obtain θ_{opt} and $q_{\min} = v(\alpha_1, \beta^1_{\theta_{\text{opt}}})$, where $q_{\min}$ is measured at θ_{opt}. This constitutes an exact

method, but requires the calculation of the intersection of *all* possible breakpoints $t \in \mathcal{T}$ of α_1 and α_2. For an efficient calculation of more complex scenarios (in particular with more flows and more segments per flow), the question arises whether we can exclude certain $t \in \mathcal{T}$ in a first step to reduce the number of v_t curves and intersections we have to calculate. We first describe the central idea: Instead of considering α_1 as a whole, we take a *decomposition* approach. From Definition 10, we know that α_1 is the minimum of n token-bucket arrival curves γ_i. Hence, we can split α_1 into its token-bucket segments $S_i, 1 \leq i \leq n$. For a token-bucket foi, we can identify a particular breakpoint $\tau \in \mathcal{T}^{\text{rel}}_{\beta^1_\theta}$ where the backlog is taken on and find a closed form for θ_{opt} that minimizes its backlog $q_{\min}$.

Theorem 21. *Let the foi α_1 be a token-bucket arrival curve γ_{r_1,b_1} and let the cross-flow α_2 be a PWL concave arrival curve in normal form. Further, let β be a PWL convex service curve. The FIFO SC parameter that minimizes the backlog bound is given by:*

$$\begin{aligned}\theta_{\text{opt}} &= \underset{\theta \in [h(\alpha_2,\beta), t_{\max}]}{\arg\min} \{\max_{t \in \mathcal{T}} \{v_t(\theta)\}\} \\ &= h(\alpha_2 + \gamma_{r_1,0}, \beta) = \frac{b_2^\tau + (r_2^\tau + r_1) \cdot \tau}{R^{\beta^{-1}(\alpha_2(\tau))}} + T^{\beta^{-1}(\alpha_2(\tau))} - \tau,\end{aligned}$$

with $\tau = a^*_{\alpha_2+\gamma_{r_1,0},\beta}$.

Proof. According to Remark 20, the result of Theorem 19 also states that the optimal θ is given by the last of all intersections of $v_t(\theta)$ curves and $\alpha_1(\theta)$. In order to obtain the last intersection, let us consider the intersections of $\alpha_1(\theta)$ and $v_t(\theta)$, with $t \in \mathcal{T}^{\text{rel}}_{\beta^1_\theta}$. The optimal value θ_t for each $t \in \mathcal{T}^{\text{rel}}_{\beta^1_\theta}$ can be calculated as follows:

$$\begin{aligned}
& \alpha_1(\theta_t) = \alpha_1(t + \theta_t) - \beta(t + \theta_t) + \alpha_2(t) \\
\Leftrightarrow \quad & r_1\theta_t + b_1 = r_1 t + r_1\theta_t + b_1 - \beta(t + \theta_t) + \alpha_2(t) \\
\Leftrightarrow \quad & \beta(t + \theta_t) = r_1 t + \alpha_2(t) \\
\Leftrightarrow \quad & \theta_t = \beta^{-1}(r_1 t + \alpha_2(t)) - t \\
\Leftrightarrow \quad & \theta_t = d_{\alpha_2+\gamma_{r_1,0}}(t).
\end{aligned}$$

Now we need to find the time at which the backlog is maximal. As the backlog bound coincides with $\alpha_1(\theta_{\text{opt}})$ and α_1 is increasing, we look for the maximal θ_t:

$$\theta_{\text{opt}} = \sup_{t \in \mathcal{T}^{\text{rel}}_{\beta^1_\theta}} \{d_{\alpha_2+\gamma_{r_1,0},\beta}(t)\} \overset{(Remark\ 17)}{=} \sup_{t \geq 0} \{d_{\alpha_2+\gamma_{r_1,0},\beta}(t)\} = h(\alpha_2 + \gamma_{r_1,0}, \beta).$$

Using the piecewise linear nature of the curves, it is clear that this horizontal deviation is taken on at $\tau = a^*_{\alpha_2+\gamma_{r_1,0},\beta}$, and thus

$$\theta_{\text{opt}} = h(\alpha_2 + \gamma_{r_1,0}, \beta) = \frac{b_2^\tau + (r_2^\tau + r_1) \cdot \tau}{R^{\beta^{-1}(\alpha_2(\tau))}} + T^{\beta^{-1}(\alpha_2(\tau))} - \tau.$$

□

Using Theorem 21, we can find the optimal value θ^i_{opt} that minimizes q^i_{min} for each γ_i corresponding to segment S_i of α_1. Considering Eq. (5), we recognize that q^i_{min} may not be assumed at a valid point in time when considering α_1 as a whole. Indeed, each segment S_i is only defined over its respective interval $I_i := [a_i, a_{i+1})$. If now $\theta^i_{\text{opt}} \notin I_i$, we adjust its value as follows (with an arbitrarily small $\epsilon > 0$):

$$\tilde{\theta}^i_{\text{opt}} := \begin{cases} a_i, & \theta^i_{\text{opt}} < a_i, \\ a_{i+1} - \epsilon, & \theta^i_{\text{opt}} > a_{i+1}, \\ \theta^i_{\text{opt}}, & \text{otherwise.} \end{cases} \tag{14}$$

This adjustment follows from the properties of the involved curves. We know that γ_i and α_2 are concave curves, their sum is hence also concave. In addition, the service curve β is convex. If we now assume θ^i_{opt} outside of the interval I_i, we can use these properties to find the largest backlog bound that is still assumed over I_i as follows: In case 1 of Eq. (14), if $\theta^i_{\text{opt}} < a_i$, we set $\tilde{\theta}^i_{\text{opt}} = a_i$. Since α_1 and α_2 are concave, their aggregate rate decreases in time, while due to the convexity of β, its rate increases in time. As such, measuring the backlog at any point $x \in I_i$, $x > a_i$, results in a smaller backlog than if measured at a_i. For case 2 it is similar, but we need to take the half-open nature of I_i into account. Using an arbitrarily small $\epsilon > 0$, and assuming the backlog at $a_{i+1} - \epsilon$ ensures that we can approximate the largest possible backlog that is still in I_i as close as we desire. This leaves us with two vectors of n values for θ^i_{opt} and $\tilde{\theta}^i_{\text{opt}}$. For purposes of the heuristic, we let $\Theta := [\theta^1_{\text{opt}}, \ldots, \theta^n_{\text{opt}}]$ and $\tilde{\Theta} := [\tilde{\theta}^1_{\text{opt}}, \ldots, \tilde{\theta}^n_{\text{opt}}]$. We proceed with two observations about θ^i_{opt} and $\tilde{\theta}^i_{\text{opt}}$:

Lemma 22. *It holds that* $\theta^1_{\text{opt}} \geq \ldots \geq \theta^n_{\text{opt}}$.

Lemma 23. *There is at most one* θ^i_{opt} *with* $\theta^i_{\text{opt}} = \tilde{\theta}^i_{\text{opt}}$.

Using these insights, we proceed with our decomposition heuristic:

1. Calculate $\Theta_{\text{diff}} := \Theta - \tilde{\Theta} = [\theta^1_{\text{opt}} - \tilde{\theta}^1_{\text{opt}}, \ldots, \theta^n_{\text{opt}} - \tilde{\theta}^n_{\text{opt}}]$.
2. Check whether there exists an entry with value $i = 0$ in Θ_{diff}. If yes, *return* $v(\alpha_1, \beta^1_{\theta^i_{\text{opt}}})$ *and break.*
3. If no such entry exists in Θ_{diff}, iterate over $t \in A_1$ and find the first intersection ϕ. *Return* $v(\alpha_1, \beta^1_\phi)$ *and break.*

Let us discuss the rationale and potential shortcomings of the heuristic: If a θ^i_{opt} is unchanged after applying Eq. (14), it may seem that we should have obtained the same value of θ_{opt} when using Theorem 19 on α_1 without decomposing the curve into its n segments. Consequently, setting $\theta_{\text{opt}} = \theta^i_{\text{opt}}$ seems to be the correct choice in this case. From Lem. 23, we know that the heuristic can find at most one such θ^i_{opt}, hence the result is unique. However, in fact, this does not yield the same result as using Theorem 19 in all cases. Whenever $\alpha_1(\tau)$ is assumed at a different segment than $\alpha_1(\tau + \theta_{\text{opt}})$, the heuristic calculates a larger θ_{opt}, and consequently larger backlog bound, as it only considers one segment of α_1. Furthermore, the heuristic can also falsely assume that there exists

no entry $i = 0$ in Θ_{diff}. In step 3, we know that all $\tilde{\theta}^i_{\text{opt}}$ in Θ_{diff} are $t \in A_1$, as we have adjusted them to their respective interval limits using Eq. (14)[1]. For each $\tilde{\theta}^i_{\text{opt}}$, we calculate its $v_{\tilde{\theta}^i_{\text{opt}}}$ curve and obtain the intersection with α_1. From Remark 20, it follows that the largest of these intersections is equal to ϕ, hence the θ_{opt} that minimizes the backlog bound. We remark that the heuristic always returns a backlog bound, as it always finds a value for θ – it may, however, be conservative.

5.1 Evaluation of the Decomposition Heuristic

We continue with an evaluation of the decomposition heuristic with respect to its accuracy and efficiency. To this end, we calculate and compare the backlog bounds and execution times of the exact method using Theorem 19 and the decomposition heuristic. We consider the following scenario: Assume we have a variable number of 2 to 10 cross flows at the node, multiplexed in a FIFO aggregate with the foi. Let α_1 be a PWL concave arrival curve for the foi with either two or four segments (as typically found in literature, e.g. [15]). Each cross flow arrival curve α^i_2 is modeled as a T-Spec curve, defined as $\alpha^i_2(t) := \min\{b^1_2 + r^1_2 t, b^2_2 + r^2_2 t\}$. We pick the packet size b^1_i for each flow i randomly from 0.001 Mbit, the sustained rate r^2_i from 1 Mbit/s–10 Mbit/s, and and the first breakpoint a^i_1 from 50 ms – 500 ms. For α_1, we pick a random breakpoint spacing from 100 ms–500 ms that is added to a^1_1 to obtain further breakpoints. We set the peak rate for each arrival curve as $r^1_i = r^2_i \cdot 8$, similar to [7]. For α_1 with four segments, we set the rate of the fourth segment as sustained rate, and assign rate $6 \cdot r^1_i$ to segment two, and rate $3 \cdot r^1_i$ to segment three. The burst value of each subsequent token bucket is set as

$$b^i = b^{i-1} - (r^i - r^{i-1}) \cdot a_i. \tag{15}$$

The server offers a rate-latency service curve $\beta_{R,T}$, where the rate is set such that there is an 80% utilization of the server when considering the sum of sustained rates of all flows. The latency is set as $T = 1/R$ s.

The experiments were run on hardware using an Intel i7-1165G7 CPU with 4 cores at 2.8 GHz base frequency, 32 GB DDR4-3200 MHz RAM, a 512 GB NVMe SSD and the Ubuntu 22.04 LTS operating system. The experiments are implemented using the Nancy library [16] for C#, and are run in JetBrains Rider 2024.3 with the .NET SDK 9.0.102 using default compiler settings.

We perform 500 iterations, where in each one of them we iterate again over the number of cross flows from 2 to 10. For each run, we measure the execution time using *System.Diagnostics.Process.UserProcessorTime*. This method returns the CPU time of the process that is running the experiments, without including any other processing time. We only measure code fragments that are used to calculate one of the two methods, ignoring any other code parts. We take the execution time of both methods and calculate the percentage difference of both

[1] They could be off by ϵ, but we can let $\epsilon \to 0$.

measured times. Of course, we also compare the backlog bounds obtained by using the exact method and the heuristic.

Table 1. Comparison of exact method and heuristic for α_1 with two segments.

Cross [#]	2	3	4	5	6	7	8	9	10
μ_{ex}	15.1	15.7	16.1	16.4	16.6	16.7	16.9	17.0	17.1
CI(95%)	0.8	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9
μ_{heu}	15.1	15.7	16.1	16.4	16.6	16.7	16.9	17.0	17.1
CI(95%)	0.8	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9
Acc. [%]	98.4	99.6	99.6	100	100	100	100	100	100
Inc. [%]	1.6	6.7	3.3	0	0	0	0	0	0
CI(95%)	0.0	0.3	0.1	0	0	0	0	0	0
$t_{\text{ex}}^{\text{exec}}$ [ms]	976	1362	1742	2114	2520	2918	3335	3758	4202
$t_{\text{heu}}^{\text{exec}}$ [ms]	14.3	12.6	12.8	13.0	13.3	13.6	14.1	14.2	14.6
Speedup	68	108	136	163	189	215	237	265	288

The results of the experiments for two segments are shown in Table 1. The results for four segments an be found in the arXiv version of this paper [14]. The tables are organized as follows. We report the mean and confidence interval (CI) for the backlog bound (μ_{ex} and μ_{heu}) as well as execution times ($t_{\text{ex}}^{\text{exec}}$ and $t_{\text{heu}}^{\text{exec}}$) for each number of cross flows. The accuracy reports how often the resulting backlog bound using the heuristic is equal to the backlog bound of the exact method. We also calculate the percentage increase and its CI from the exact method to the heuristic in cases where the results are not identical. For the execution times, the speedup from the exact method to the heuristic is calculated as $t_{\text{ex}}^{\text{exec}}/t_{\text{heu}}^{\text{exec}}$. Furthermore, for 3 and 7 cross flows, we illustrate the relation of the obtained backlog bounds and execution times for both methods and segment counts in Figs 4 and 5. For both figures, the scenario numbers are ordered by the value of the respective heuristic results (execution times and backlog bounds), from smallest to largest. For the backlog bounds, whenever the exact method calculates a more accurate backlog bound, this is represented as a spike down in the plot.

We observe that, for two segments, the heuristic provides very accurate results compared to the exact method. For 2 to 4 cross flows, a small percentage of runs do not produce the same backlog bound. For these, we observe a percentage increase of up to 6.7% of the backlog bound for 3 cross flows. (Yet, the overall mean of the iterations is not affected by this.) The accuracy for four α_1 segments is worse, but the percentage increase stays low across all numbers of cross flows. For the execution times, we observe that it grows in the number of cross flows. Additionally, in Table 1 we observe that the ratio of the execution times (speedup) between the two methods grows significantly. For four segments, we observe the same behavior, with a slower speedup across cross flow numbers.

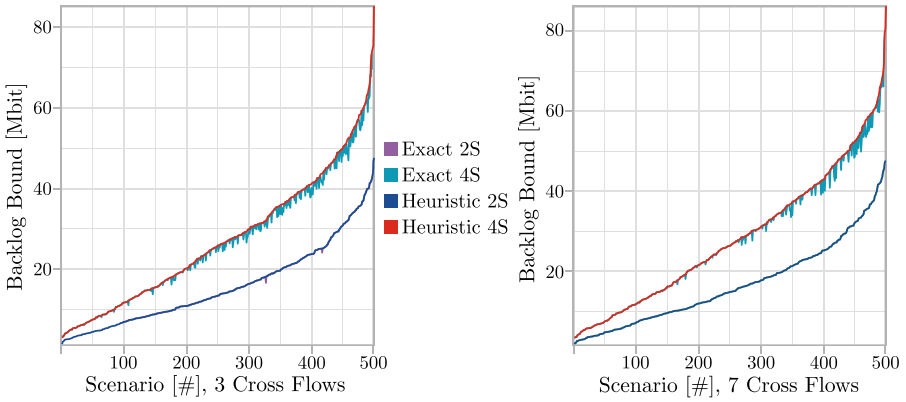


Fig. 4. Backlog bounds for varying numbers of cross flows and foi segments.

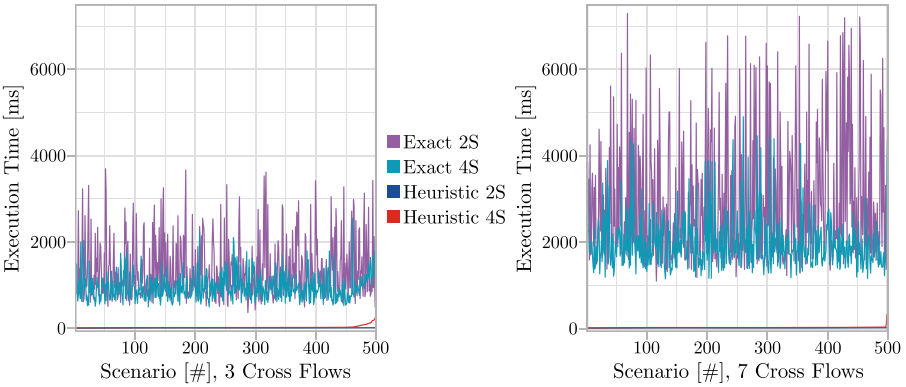


Fig. 5. Execution time for varying numbers of cross flows and foi segments.

In Fig. 5, we observe that the execution time of the exact method is faster for α_1 with four segments. We suspect that, due to the fact that α_1 is larger for four segments than it is for two, that we find ϕ earlier on average. This is supported by the fact that the calculation of the whole set of v_t curves results in similar execution times for both numbers of segments.

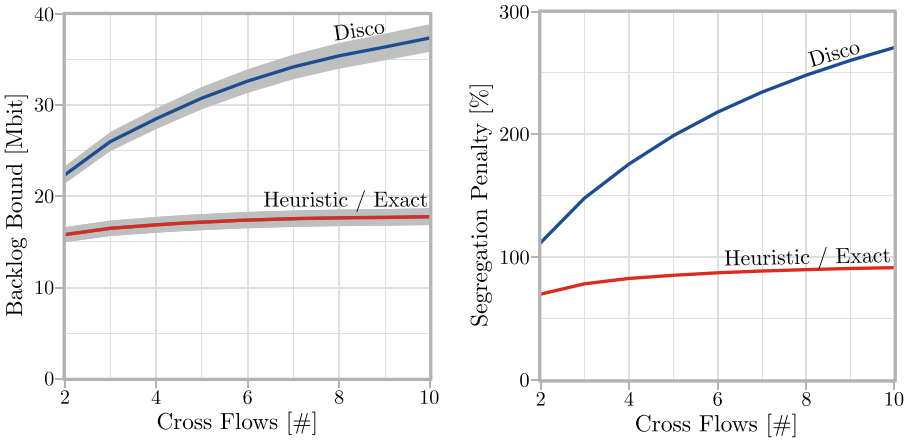
6 Usage of the Results in the DiscoDNC Tool

In the previous section, we have conducted our experiments using Nancy. Nancy is a general toolbox to support network calculus analyses but does not provide complete analysis methods by itself. Consequently, it does not mandate any specific value for θ, but instead we used it to compute such values. In contrast to Nancy, another open-source tool called DiscoDNC [2,11,12] provides complete network calculus analyses, in particular also for FIFO with a default setting of

θ as follows: $\theta_{\text{disco}} = \beta^{-1}\left(\sum_{i=2}^{n} b_1^i\right)$. In fact, this θ value coincides with the one that was used in Sect. 4 as a lower bound to the optimal theta value θ_{opt} *when* all arrival curves are token buckets and the service curve is a rate-latency service curve. For more complex curves, it holds that $\theta_{\text{disco}} < h(\alpha_2, \beta)$, however. Setting $\theta \leq h(\alpha_2, \beta)$ results in a β_θ^1 that is continuous in all points. It was consequently a reasonable first choice for an NC tool.

Yet, now we are able to adjust the θ value employed by DiscoDNC using our exact method and the decomposition heuristic. In the following, we are interested in the increase in quality of the calculated backlog bounds we can achieve when adjusting θ using the exact method as well as the heuristic.

We use the same experiment setup that we have used in Sect. 5.1. We compare the obtained backlog bound when using θ_{disco} against the backlog bound obtained using the exact method and heuristic. The results are illustrated in Fig. 6a. Here, we have calculated the mean backlog bounds and confidence intervals (CI) for the three methods for different numbers of cross flows. We observe that for all numbers of cross flows, the exact method and heuristic produce very similar backlog bounds (invisible difference in the figure). For $n = 10$, the backlog bound of both methods has the same mean and CI, with CI(95%) = 17.76±0.308. The backlog bound for the default value θ_{disco} has CI(95%) = 37.34 ± 0.506. For two cross flows, the exact method and heuristic have CI(95%) = 15.8 ± 0.272, and the default value has CI(95%) = 22.3 ± 0.320.



(a) Mean backlog bound.

(b) Segregation penalty.

Fig. 6. Comparisons of exact method and heuristic against the DiscoDNC tool.

Additionally, we study the segregation penalty (see Sect. 3) for this experimental setup. To this end, we calculate the segregation penalty as $(\sum q_{\min}^i - q_{\text{agg}})/q_{\text{agg}} \cdot 100$, where q_{agg} is the aggregate backlog bound. The results are given in Fig. 6b. We observe that the segregation penalty using the default θ value of

DiscoDNC grows significantly in the number of cross flows, in contrast to the new methods. This may potentially result in sub-optimal system design decisions.

To conclude, the backlog bound using θ_{disco} is significantly less accurate than the other two methods for any number of cross flows, with the difference becoming larger and larger the more cross flows traverse the server.

7 Conclusion

In this paper, we presented an exact method for deriving FIFO residual service curves that minimize backlog bounds for PWL concave arrival curves. While this method provides precise results, its computational inefficiency becomes apparent in more complex scenarios, such as settings with a large number of crossflows. To address this limitation, we introduced an efficient heuristic that provides an upper bound on the backlog-minimizing parameter of the FIFO residual service curve. Despite its approximate nature, we could show the heuristic yields backlog bounds that are close to those of the exact method, particularly in scenarios with a higher number of crossflows. Importantly, it achieves this with significantly reduced execution time. In a final experiment with the DiscoDNC tool, both the exact and heuristic approaches produced significantly more precise backlog bounds than the existing default setting of the tool.

References

1. Blanc, A.P.: Quality of service guarantees for FIFO queues with constrained inputs. University of California, San Diego (2006)
2. Bondorf, S., Schmitt, J.B.: The DiscoDNC v2 – a comprehensive tool for deterministic network calculus. In: Proceedings of the International Conference on Performance Evaluation Methodologies and Tools, ValueTools 2014, pp. 44–49 (2014). https://dl.acm.org/citation.cfm?id=2747659
3. Bouillard, A., Boyer, M., Le Corronc, E.: Deterministic Network Calculus: From Theory to Practical Implementation. Wiley (2018)
4. Cholvi, V., Echagüe, J., Boudec, J.Y.: Worst case burstiness increase due to FIFO multiplexing. Perform. Eval. **49**(1–4), 491–506 (2002)
5. Cruz, R.L.: A calculus for network delay. I. Network elements in isolation. IEEE Trans. Inf. Theory **37**(1), 114–131 (1991)
6. Cruz, R.L.: SCED+: efficient management of quality of service guarantees. In: Proceedings. IEEE INFOCOM 1998, the Conference on Computer Communications. Seventeenth Annual Joint Conference of the IEEE Computer and Communications Societies. Gateway to the 21st Century (Cat. No. 98. vol. 2, pp. 625–634. IEEE (1998)
7. Le Boudec, J.Y., Thiran, P.: Network Calculus: A Theory of Deterministic Queuing Systems for the Internet. Springer (2001). https://leboudec.github.io/netcal/
8. Lenzini, L., Mingozzi, E., Stea, G.: Delay bounds for FIFO aggregates: a case study. Comput. Commun. **28**(3), 287–299 (2005)
9. Liebeherr, J., Fidler, M., Valaee, S.: A system-theoretic approach to bandwidth estimation. IEEE/ACM Trans. Networking **18**(4), 1040–1053 (2009)

10. McKeown, N., Mekkittikul, A., Anantharam, V., Walrand, J.: Achieving 100% throughput in an input-queued switch. IEEE Trans. Commun. **47**(8), 1260–1267 (2002)
11. Scheffler, A., Bondorf, S.: Network calculus for bounding delays in feedforward networks of FIFO queueing systems. In: Abate, A., Marin, A. (eds.) QEST 2021. LNCS, vol. 12846, pp. 149–167. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-85172-9_8
12. Schmitt, J.B., Zdarsky, F.A.: The disco network calculator: a toolbox for worst case analysis. In: Proceedings of the 1st International Conference on Performance Evaluation Methodologies and Tools, pp. 8–es (2006)
13. Shvachko, K., Kuang, H., Radia, S., Chansler, R.: The hadoop distributed file system. In: 2010 IEEE 26th Symposium on Mass Storage Systems and Technologies (MSST), pp. 1–10. IEEE (2010)
14. Wildberger, L., Hamscher, A., Schmitt, J.B.: Minimal per-flow backlog bounds at an aggregate FIFO server under piecewise-linear arrival curves (2025). https://arxiv.org/abs/2506.16914
15. Wrege, D.E., Knightly, E.W., Zhang, H., Liebeherr, J.: Deterministic delay bounds for VBR video in packet-switching networks: fundamental limits and practical trade-offs. IEEE/ACM Trans. Networking **4**(3), 352–362 (1996)
16. Zippo, R., Stea, G.: Nancy: an efficient parallel network calculus library. SoftwareX **19**, 101178 (2022)