# Unleashing the Power of Paying Multiplexing Only Once in Stochastic Network Calculus

Anne Bouillard
Huawei Technologies France
anne.bouillard@huawei.com

Paul Nikolaus, Jens Schmitt
Distributed Computer Systems (DISCO) Lab
TU Kaiserslautern
{nikolaus,jschmitt}@cs.uni-kl.de

## Abstract

The stochastic network calculus (SNC) holds promise as a framework to calculate probabilistic performance bounds in networks of queues. A great challenge to accurate bounds and efficient calculations are stochastic dependencies between flows due to resource sharing inside the network. However, by carefully utilizing the basic SNC concepts in the network analysis the necessity of taking these dependencies into account can be minimized. To that end, we fully unleash the power of the pay multiplexing only once principle (PMOO, known from the deterministic network calculus) in the SNC analysis. We choose an analytic combinatorics presentation of the results in order to ease complex calculations. In tree-reducible networks, a subclass of a general feedforward networks, we obtain a perfect analysis in terms of avoiding the need to take internal flow dependencies into account. In a comprehensive numerical evaluation, we demonstrate how this unleashed PMOO analysis can reduce the known gap between simulations and SNC calculations significantly, and how it favourably compares to state-of-the art SNC calculations in terms of accuracy and computational effort. Driven by these promising results, we also consider general feedforward networks, when some flow dependencies have to be taken into account. To that end, the unleashed PMOO analysis is extended to the partially dependent case and a case study of a canonical example topology, known as the diamond network, is provided, again displaying favourable results over the state of the art.

## 1 Introduction

Stochastic network calculus (SNC) is a promising uniform framework to calculate probabilistic end-to-end performance bounds for individual flows in networks of queues. The most prominent goal is to control tail probabilities for the end-to-end (e2e) delay, i.e., probabilities for rare events shall be bounded, e.g., $\mathbf{P}(\text{e2e delay} > 10\text{ms}) \leq 10^{-6}$. Many modern systems are eager after such guarantees, as exemplified in visions like, e.g. Tactile Internet [18] or Industrial IoT [9].

SNC originates in the deterministic analysis by Rene Cruz [14, 15] and was subsequently supplemented by the use of min-plus algebra [1]. In the following years, it was transferred to a stochastic setting [10, 11, 20, 26, 13]. Over the course of almost 30 years, two main branches of SNC have evolved: either by characterizing arrivals and service by envelope functions / tail bounds [16, 11, 26], or, by moment-generating function (MGF) bounds [10, 20]. While a larger class of processes can be modelled with tail bounds, [32] comes to the conclusion that using MGFs leads to tighter bounds under the assumption of independence.

It should be mentioned that the *uniform* approach of SNC, in particular to apply the union bound to evaluate sample-path events, comes at a price: already in the single-node case there is a known gap between simulations and SNC calculations (see Figure 2a on the next page for some typical numerical results). In fact, there is a tight analysis for the single-node case for some traffic classes based on martingale techniques [12, 31] (avoiding the use of the union bound); yet, an end-to-end martingale analysis remains an elusive goal. Therefore, in this paper, we keep following the

(a) Original topology

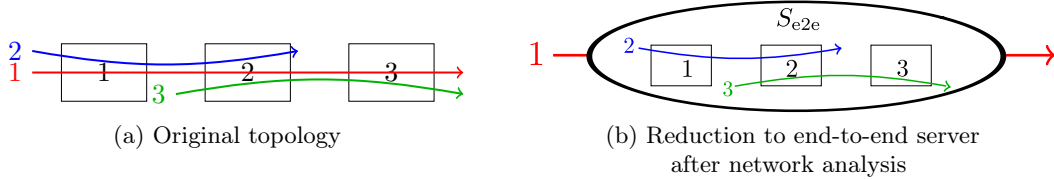(b) Reduction to end-to-end server after network analysis

Figure 1: Interleaved tandem network

uniform approach of SNC and concentrate our efforts on not widening the simulation-calculation gap further when analysing larger and more complex networks.

Analysing more general networks of queues, in particular feedforward networks, usually requires the consideration of stochastically dependent flows. Even if all external arrival and service processes are independent, the sharing of resources by individual flows at queues generally makes them stochastically dependent at subsequent queues. How much this kind of dependencies has to be taken into account is affected by the network analysis method because different methods require different levels of knowledge about the internal characterization of flows. Further on, we call these dependencies *method-pertinent*. To deal with (method-pertinent) stochastic dependencies in SNC, typically, the MGF of, e.g. the sum of dependent flows is upper bounded by Hölder's inequality (HI). While HI's generality is convenient and consistent with SNC's aspiration as a uniform framework, it often incurs the drawback of degrading bound accuracy considerably, further widening the simulation-calculation gap. In addition, it increases the computational effort by introducing an additional parameter to optimize for each application of HI, such that in larger scenarios runtimes quickly become prohibitive (see also Subsection 4.2). Consequently, SNC analysis methods with less method-pertinent dependencies are strongly favourable as they require less invocations of HI. In fact, previous work in relatively simple network scenarios has indicated that techniques which completely avoid HI achieve significantly better delay bounds [36, 28].

Different network analysis methods have been investigated intensively in the deterministic setting; some better known ones are, e.g. Separate Flow Analysis (SFA) or Pay multiplexing only once (PMOO) [33]. These typically try to reduce the analysis (e.g. for the delay) of a particular flow in the network to a simple analysis of this flow traversing a single server; the characteristics of that so-called end-to-end server depend on the cross-flows and the servers of the original network, see also Figure 1. The main goal for the network analysis in the deterministic case was to properly account for the bursts of the flows to ensure that they are only "paid once". In particular, the PMOO principle [33, 34, 8, 19] tries to ensure this for all flows in the network by, whenever possible, concatenating servers first before calculating residual service by "subtracting" cross-flows. As discussed above, in SNC the main concern is dependencies, yet it turns out that, to some extent, this is related to the proper accounting of bursts. In fact, in [28], it has been observed that the PMOO analysis known from deterministic network calculus [33, 34, 8, 19] leads to less method-pertinent dependencies compared to SFA and is thus also promising in an SNC analysis.



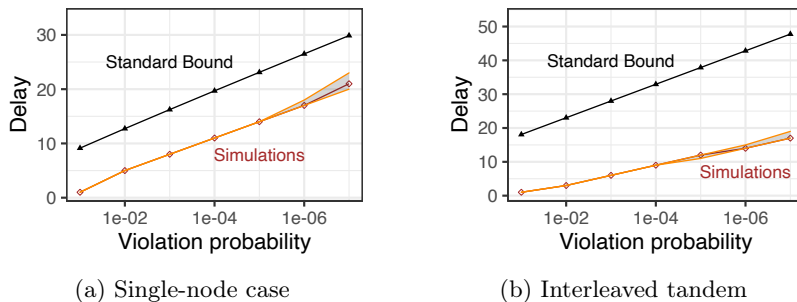(a) Single-node case

(b) Interleaved tandem

Figure 2: SNC delay bounds and simulation results (for traffic with exponentially distributed increments and constant rate servers). For more details, see Section 4.

2

However, the application of PMOO in the SNC has, so far, been limited to so-called nested interference structures – this is very restrictive. For instance, for a network with interleaved interference as in Figure 1a, state-of-the-art analysis still requires at least one application of HI, even if we assume all external arrival processes to be independent. This is also illustrated in Figure 2b (anticipating some of the results from Section 4): method-pertinent dependencies force the state-of-the-art SNC bounds to deteriorate significantly; in particular, it is observable that, in comparison to the single-node case, even the scaling of the delay bounds is not captured correctly any more and the simulation-calculation gap widens considerably even for such a small network.

The overall goal of our paper is therefore to unleash the power of the PMOO principle in the SNC framework in order to not widen the simulation-calculation gap further even in more complex and larger networks of queues. To that end, we make the following contributions:

- We present a PMOO-based SNC end-to-end analysis for a subclass of feedforward networks, so-called tree-reducible networks; the main result is given in Theorem 6. It achieves *zero* method-pertinent stochastic dependencies when external arrivals and service processes are independent. I.e., if all input flows are assumed to be independent, we can derive bounds without using Hölder's inequality. Also, Theorem 6 allows us to calculate the residual service in one big step avoiding the sequencing penalty in previous network analysis methods.

- We apply analytic combinatorics [22] to recover bounds from state-of-the-art analysis methods in simple networks and enable a generalization to more complex settings.

- We conduct an extensive numerical evaluation with respect to the accuracy of the new bounds for several traffic classes and different network topologies.

- We discuss first results to extend our new method from tree-reducible to general feedforward networks, still striving for the goal to minimize method-pertinent dependencies.

## 2   MGF-based Network Calculus and Analytic Combinatorics

In this section, we briefly introduce the two main tools we employ in our paper, namely the MGF-based stochastic network calculus in Subsection 2.1 and analytic combinatorics in Subsection 2.2. Furthermore, we show in Subsection 2.3 how analytic combinatorics can be used to express complex calculations in MGF-based SNC in a concise manner.

### 2.1   Stochastic network calculus framework

We present the main concepts of the moment-generating function (MGF)-based network calculus for the single-server case (the network case is postponed to Section 3 in the context of our new contribution). More details can be found in [10, 20]. We assume time is discrete and space is continuous. We deal with bivariate functions and always assume that their definition domain is $\{(s,t) \in \mathbb{N}^2 \mid s \leq t\}$ and that they are non-negative: $f(s,t) \geq 0$ for all $s \leq t$.

**Bivariate arrival processes**   A bivariate (arrival) process $A$ of a flow at some point in the network represents the amount of data of the flow traversing that point of the network during any interval of time: let $a_i$ be the amount of data during the $i$-th time slot; we define for all $s \leq t$, $A(s,t) = \sum_{i=s+1}^{t} a_i$, with the convention $A(t,t) = 0$. Note that bivariate arrival processes are additive: for all $s \leq u \leq t$, $A(s,u) + A(u,t) = A(s,t)$.

**(min, plus) operations**  To describe the transformation between bivariate processes when traversing a server, we rely on the following operations in the (min,plus)-algebra. Let $f$ and $g$ be two bivariate functions.

- *(min, plus)-convolution:* $f \otimes g(s,t) = \min_{s \leq u \leq t} f(s,u) + g(u,t)$;

- *(min, plus)-deconvolution:* $f \oslash g(s,t) = \max_{0 \leq u \leq s} f(u,t) - g(u,s)$.

**$S$-servers**  Let $S$ be a non-negative bivariate function. A server is a dynamic $S$-server if the relation between its bivariate arrival and departure processes $A$ and $D$ satisfies for all $t \geq 0$, $A(0,t) \geq D(0,t) \geq A \otimes S(0,t)$.

This notion of server can be too weak in some situations when performing a network analysis, and we introduce work-conserving $S$-servers: assume that $S(t,t) = 0$ for all $t$ and let $start(t) = \sup\{s \leq t \mid A(0,s) = D(0,s)\}$ be the last instant before $t$ when the server is empty. We also call it the *start of the backlogged period* of $t$. Note that from this definition, if the server is idle at time $t$, then $start(t) = t$. We say that the $S$-server is *work-conserving* if for all $t$ and $s \in \{start(t),\ldots,t\}$, $D(0,t) \geq D(0,s) + S(s,t)$. In other words, the service offered between times $s$ and $t$ is at least $S(s,t)$, provided that the server is always backlogged between times $s$ and $t$. We can always assume that $S$ is additive, whereas this does not have to hold for dynamic $S$-servers.

The notions of dynamic $S$-server and of work-conserving $S$-server, respectively, correspond to the notions of service curve and strict service curve in deterministic network calculus [6]. Note that a work-conserving $S$-server is also a dynamic $S$-server.

**Departure process characterization and performance bounds**  Consider a dynamic $S$-server and $A$ and $D$ its respective bivariate arrival and departure processes.

The backlog at time $t$ is $q(t) = A(0,t) - D(0,t)$ and the virtual delay at time $t$ is $d(t) = \inf\{T \in \mathbb{N} \mid A(0,t) \leq D(0,t+T)\}$.

**Theorem 1** (Sample-Path Bounds [10, 20]). *Consider a dynamic $S$-server traversed by a flow with bivariate arrival process $A$. Then,*

- $D(s,t) \leq A \oslash S(s,t)$;

- $q(t) \leq A \oslash S(t,t)$;

- $d(t) \geq T \Rightarrow \exists s \leq t,\ A(s,t) > S(s,t+T-1)$.

**$(\sigma,\rho)$-constraints**  MGF-based SNC relies on bounds on the MGF and the Laplace transform for the arrival and service process, respectively. Specifically, we assume that $A$ and $S$ are bivariate stochastic processes, and that $\mathbf{E}[e^{\theta A(s,t)}]$ and $\mathbf{E}[e^{-\theta S(s,t)}]$ both exist for some $\theta > 0$. We say that the bivariate arrival process $A$ is $(\sigma_A, \rho_A)$-constrained if there exist $\theta > 0$ and $\sigma_A(\theta), \rho_A(\theta) \in \mathbb{R}_+$ such that $\mathbf{E}[e^{\theta A(s,t)}] \leq e^{\theta(\sigma_A(\theta) + \rho_A(\theta)(t-s))}$. Similarly, we say that the bivariate service process $S$ is $(\sigma_S, \rho_S)$-constrained if there exist $\theta > 0$ and $\sigma_S(\theta), \rho_S(\theta) \in \mathbb{R}_+$ such that $\mathbf{E}[e^{-\theta S(s,t)}] \leq e^{\theta(\sigma_S(\theta) - \rho_S(\theta)(t-s))}$.

Such linear bounds in the transform space provide the advantage that they are closed with respect to network calculus operations, e.g. multiplexing of flows. However, more general bounds are not fundamentally precluded and in some of the following results we also provide this generality when it does not harm the clarity of the presentation.

## 2.2 Analytic combinatorics framework

Let $f = (f_n)_{n\in\mathbb{N}} \in \mathbb{R}_+^{\mathbb{N}}$ be a sequence of non-negative numbers. The (ordinary) generating function (related to the $z$-transform) associated to $f$ is the function $F(z) = \sum_{n=0}^{\infty} f_n z^n$. An important example in the following is the geometric sequence $f_n = a^n$, for which

$$F(z) = \sum_{n=0}^{\infty} a^n z^n = \frac{1}{1-az}. \tag{1}$$

Given a generating function $F(z)$, we denote the $n$-th term of its associated sequence by $[z^n]F(z)$. In other words, with the previous notation, $[z^n]F(z) = f_n$. Analytic combinatorics [22] is a branch of combinatorics whose aim is the study of such sequences, and, in particular, the relation between the singularities of the generating function and the asymptotic behaviour of $f$. We denote by $r_F$ the dominant singularity of $F$. In the following, we only use rational fractions, and the dominant singularity corresponds to the smallest root of the denominator. For example, the dominant singularity of $F(z)$ in Equation (1) is $r_F = a^{-1}$.

Next, we apply the framework of analytic combinatorics to simplify some SNC calculations, in particular, when computing error bounds for backlogs and delays. Let us first present some operations on sequences and their associated generating functions.

**Cauchy product:** Let $f = (f_n)_{n\in\mathbb{N}}$ and $g = (g_n)_{n\in\mathbb{N}}$ be two sequences. The Cauchy product of these sequences is $h_{\mathrm{Cau}} = (h_{\mathrm{Cau},n})_{n\in\mathbb{N}}$ with $h_{\mathrm{Cau},n} = \sum_{p=0}^{n} f_p g_{n-p}$. The generating function of $h_{\mathrm{Cau}}$ is

$$H_{\mathrm{Cau}}(z) = \sum_{n=0}^{\infty} \sum_{p+q=n} f_p g_q z^n = \left(\sum_{p=0}^{\infty} f_p z^p\right)\left(\sum_{q=0}^{\infty} g_q z^q\right) = F(z)G(z),$$

and is also called the Cauchy product of $F$ and $G$. The dominant singularity of $H_{\mathrm{Cau}}$ is $r_{H_{\mathrm{Cau}}} = \min(r_F, r_G)$. In the SNC, the (min,plus)-convolution corresponds to a Cauchy product.

**Hadamard product:** Let $f = (f_n)_{n\in\mathbb{N}}$ and $g = (g_n)_{n\in\mathbb{N}}$ be two sequences. The Hadamard product of these sequences is $h_{\mathrm{Had}} = (h_{\mathrm{Had},n})_{n\in\mathbb{N}}$ with $h_{\mathrm{Had},n} = f_n g_n$. There is no simple expression for the generating function of the Hadamard product in the general case. Yet, in the case where $f = (a^n)_{n\in\mathbb{N}}$ is a geometric sequence, the Hadamard product of $F$ and $G$ is

$$H_{\mathrm{Had}}(z) = \sum_{n=0}^{\infty} a^n g_n z^n = G(az). \tag{2}$$

The dominant singularity of $H_{\mathrm{Had}}$ is $r_{H_{\mathrm{Had}}} = r_F \cdot r_G$. In the SNC, the Hadamard product is used when computing the (min,plus)-deconvolution.

**Sum of last terms:** Let $f = (f_n)_{n\in\mathbb{N}}$ be a summable sequence. Let $g_n = \sum_{m\geq n} f_m$. The generating function of $g = (g_n)_{n\in\mathbb{N}}$, proved in Appendix A.1, is

$$G(z) = \frac{F(1) - zF(z)}{1-z}. \tag{3}$$

Note that 1 is not a singularity of $G(z)$ if it is not a singularity of $F$. Consequently, $F$ and $G$ have the same dominant singularity. This operation is used when computing delay bounds.

## 2.3 A combinatorial view of MGF-based network calculus

The purpose of this subsection is to present the $(\sigma, \rho)$-constraints on arrival and service processes as generating functions. This alleviates the calculation of residual service constraints in complex scenarios, thus enabling accurate performance bounds. The use of analytic combinatorics in not new in the field of queueing theory. For example, Flajolet and Guillemin in [21] analyze Markovian queues, Guillemin and Pinchin in [24] study the processor sharing policy, and more recently, [7] presents an attempt to use analytical combinatorics in Network Calculus. The approach of all these references is to model the distributions of the queues by a generating functions and use the machinery to compute the asymptotic behavior of the queue. Here, we use generating functions only for the purpose of simplifying the computations of the SNC framework. To our knowledge, this is the first time this tool is used for this purpose.

**Definition 2.** Let $A$ be a bivariate arrival process. An *arrival bounding generating function* of $A$ at $\theta$ is a function $F_A(\theta, z)$ such that for all $s, n \geq 0$, $\mathbf{E}[e^{\theta A(s,s+n)}] \leq [z^n] F_A(\theta, z)$.

For example, if $A$ is $(\sigma_A, \rho_A)$-constrained and $\sigma_A(\theta), \rho_A(\theta) \in \mathbb{R}_+$,

$$F_A(\theta, z) = \sum_{n=0}^{\infty} e^{\theta(\sigma_A(\theta) + \rho_A(\theta)n)} z^n = \frac{e^{\theta \sigma_A(\theta)}}{1 - e^{\theta \rho_A(\theta)} z}$$

is an arrival bounding generating function for $A$ at $\theta$.

Similarly, if $S$ is a service process, a *service bounding generating function* of $S$ at $\theta$ is a function $F_S(\theta, z)$, such that for all $s, n \geq 0$, $\mathbf{E}[e^{-\theta S(s,s+n)}] \leq [z^n] F_S(\theta, z)$.

In the following, it shall be clear from the context whether a bounding generating function is an arrival or a service one. Therefore, we omit this precision for the sake of readability.

Using bounding generating functions of $A$ and $S$ at $\theta$, we can derive bounding generating functions for the departure process and the violation probability of the probabilistic delay bound. The backlog bound can be directly deduced from the departure process. The following results provide bounding generating functions for the departure process and the performance bounds.

**Lemma 1** (Departure process bounding generating function). *Consider a dynamic $S$-server traversed by a flow with bivariate process $A$ that is $(\sigma_A, \rho_A)$-constrained. Assume that $S$ has the service bounding generating function $F_S$ and that the arrival and service processes $A$ and $S$ are independent. Then, for all $\theta$ such that $e^{-\theta \rho_A(\theta)} r_S(\theta) < 1$, an arrival bounding generating function of the departure process $D$ is*

$$F_D(\theta, z) = \frac{e^{\theta \sigma_A(\theta)} F_S(\theta, e^{\theta \rho_A(\theta)})}{1 - e^{\theta \rho_A(\theta)} z}.$$

*Proof.* As $F_A(\theta, z)$ is geometric, we have $[z^{n+m}] F_A(\theta, z) = e^{\theta \rho_A(\theta) m} [z^n] F_A(\theta, z)$. Since $D(s,t) \leq A \oslash S(s,t) = \sup_{u \leq s} A(u,t) - S(u,s)$, then for $\theta > 0$,

$$\mathbf{E}[e^{\theta A \oslash S(s,t)}] \leq \sum_{u \leq s} \mathbf{E}[e^{\theta(A(u,t) - S(u,s))}]$$

$$= \sum_{u \leq s} \mathbf{E}[e^{\theta A(u,t)}] \cdot \mathbf{E}[e^{-\theta S(u,s)}]$$

$$\leq \sum_{u \leq s} [z^{t-u}] F_A(\theta, z) \cdot [z^{s-u}] F_S(\theta, z)$$

$$= e^{\theta \rho_A(\theta)(t-s)} \sum_{u \leq s} [z^{s-u}] F_A(\theta, z) \cdot [z^{s-u}] F_S(\theta, z)$$

$$\leq e^{\theta \rho_A(\theta)(t-s)} \sum_{n \geq 0} [z^n] F_A(\theta, z) \cdot [z^n] F_S(\theta, z)$$

6

$$= e^{\theta \rho_A(\theta)(t-s)} e^{\theta \sigma_A(\theta)} F_S(\theta, e^{\theta \rho_A(\theta)})$$
$$= [z^{t-s}] F_A(\theta, z) F_S(\theta, e^{\theta \rho_A(\theta)}).$$

To obtain the second to last equality, we applied the Hadamard product to a geometric series as in Equation (2) with $z = 1$. We now recognize the $(t-s)$-th term of a geometric series with ratio $e^{\theta \rho_A(\theta)}$, so

$$F_D(\theta, z) = \frac{e^{\theta \sigma_A(\theta)} F_S(\theta, e^{\theta \rho_A(\theta)})}{1 - e^{\theta \rho_A(\theta)} z}$$

is a bounding generating function for the departure process. $\qquad \square$

**Corollary 1** (Probabilistic backlog bound). *Let us denote $q(t)$ as the backlog at time $t$. Then, under the same assumptions as in Lemma 1,*

$$\mathbf{P}(q(t) \geq b) \leq e^{-\theta b} e^{\theta \sigma_A(\theta)} F_S(\theta, e^{\theta \rho_A(\theta)}).$$

*Proof.* As from Theorem 1 we have $q(t) \leq A \oslash S(t, t)$, we can apply the Chernoff bound using the first term of the generating function $F_D(\theta, z)$ of Lemma 1:

$$\mathbf{P}(q(t) \geq b) \leq \mathbf{E}[e^{\theta A \oslash S(t,t)}] e^{-\theta b}$$
$$\leq e^{\theta \sigma_A(\theta)} F_S(\theta, e^{\theta \rho_A(\theta)}) e^{-\theta b}.$$

$\qquad \square$

In the particular case of $F_S(\theta, z) = \frac{e^{\theta \sigma_S(\theta)}}{1 - e^{-\theta \rho_S(\theta)} z}$, for all $\theta \geq 0$ such that $\rho_A(\theta) < \rho_S(\theta)$,

$$\mathbf{P}(q(t) \geq b) \leq \frac{e^{\theta(\sigma_A(\theta) + \sigma_S(\theta))}}{1 - e^{\theta(\rho_A(\theta) - \rho_S(\theta))}} e^{-\theta b}.$$

We have made the assumption of $(\sigma_A, \rho_A)$-constraints for the arrival processes, and our proofs rely on this assumption. Yet, we remark that the calculations can be done in a more general setting, at the price of more complex formulas. For example, if the arrival bounding generating function of the arrival process $A$ is a sum of geometric series, as for example for Markov-modulated arrivals, the calculation directly follows, and the arrival bounding generating function of $D$ would be a sum of geometric series as well. Another case where computations can be adapted is when the service is bounded by a geometric series: symmetric computations can be done.

**Lemma 2** (Delay bound generating function). *Let us denote $d(t)$ as the virtual delay at time $t$. Under the same assumptions as in Lemma 1, a bounding generating function for the violation probability of the probabilistic delay bound is*

$$F_d(\theta, z) = e^{\theta \sigma_A(\theta)} \frac{e^{\theta \rho_A(\theta)} F_S(\theta, e^{\theta \rho_A(\theta)}) - z F_S(\theta, z)}{1 - e^{-\theta \rho_A(\theta)} z}. \tag{4}$$

A proof can be found in Appendix A.2. In the particular case of $F_S(\theta, z) = \frac{e^{\theta \sigma_S(\theta)}}{1 - e^{-\theta \rho_S(\theta)} z}$, for all $\theta \geq 0$ such that $\rho_A(\theta) < \rho_S(\theta)$, $F_d(\theta, z) = \frac{e^{\theta(\sigma_A(\theta) + \sigma_S(\theta) + \rho_A(\theta))}}{1 - e^{-\theta(\rho_S(\theta) - \rho_A(\theta))}} \cdot \frac{1}{1 - e^{-\theta \rho_S(\theta)} z}$, so

$$\mathbf{P}(d(t) \geq T) \leq \frac{e^{\theta(\sigma_A(\theta) + \sigma_S(\theta) + \rho_A(\theta))}}{1 - e^{-\theta(\rho_S(\theta) - \rho_A(\theta))}} e^{-\theta \rho_S(\theta) T}.$$

So far, we have focused on the case of independent stochastic processes, let us now release this assumption. Since our analysis is based on moment-generating functions, the common approach in the literature is to upper bound the product of dependent processes by Hölder's inequality: let

$X_1, \ldots, X_n \geq 0$ be integrable random variables; then, for all positive numbers $q_1, \ldots, q_n$ such that $\sum_{i=1}^{n} \frac{1}{q_i} = 1$,

$$\mathbf{E}[X_1 \cdots X_n] \leq \prod_{i=1}^{n} \mathbf{E}[X_i^{q_i}]^{\frac{1}{q_i}}. \tag{5}$$

Note that $\mathbf{E}[e^{p\theta A(s,t)}]^{1/p} \leq e^{p\theta(\sigma_A(p\theta) + \rho_A(p\theta)(t-s))(1/p)} = e^{\theta(\sigma_A(p\theta) + \rho_A(p\theta)(t-s))}$ and, similarly, $\mathbf{E}[e^{-q\theta S(s,t)}]^{1/q} \leq e^{\theta(\sigma_S(q\theta) - \rho_S(q\theta)(t-s))}$.

The following lemma contains the translation of an existing output bound from [3] to the combinatorial framework as well as backlog and delay bounds for the dependent case.

**Lemma 3.** *Consider a dynamic S-server traversed by a flow with bivariate process $A$ that is $(\sigma_A, \rho_A)$-constrained, and assume that the process $S$ is also $(\sigma_S, \rho_S)$-constrained. Then for all $p, q$ such that $\frac{1}{p} + \frac{1}{q} = 1$ and $\rho_A(p\theta) < \rho_S(q\theta)$,*

- $F_D(\theta, z) = \frac{e^{\theta(\sigma_A(p\theta) + \sigma_S(q\theta))}}{1 - e^{-\theta(\rho_S(q\theta) - \rho_A(p\theta))}} \cdot \frac{1}{1 - e^{\theta\rho_A(p\theta)}z}$;

- $\mathbf{P}(q(t) \geq b) \leq \frac{e^{\theta(\sigma_A(p\theta) + \sigma_S(q\theta))}}{1 - e^{-\theta(\rho_S(q\theta) - \rho_A(p\theta))}} e^{-\theta b}$;

- $\mathbf{P}(d(t) \geq T) \leq \frac{e^{\theta(\sigma_A(p\theta) + \sigma_S(q\theta) + \rho_A(p\theta))}}{1 - e^{-\theta(\rho_S(q\theta) - \rho_A(p\theta))}} e^{-\theta\rho_S(q\theta)T}$.

# 3 Tree Network Analysis under Complete Independence

In this section, we derive a result enabling us to unleash the power of the PMOO principle for the SNC when all arrival and service processes are originally independent. In particular, this result is directly applicable to tree networks. However, given an overall network with servers and flows, it is only from the perspective of the flow of interest (foi) that the network has to constitute a tree. In other words, in a preliminary step, we (attempt to) reduce the network to a tree, with all associated arrival and service processes still being independent, by performing the following steps:

1. Terminate flows *after* their last direct or indirect interaction with the foi and reduce the network by deleting all flows with which the foi has neither a direct nor indirect dependency (also delete all servers that do not carry any flows any more).

2. Check for all remaining flows whether they directly or indirectly rejoin (a) the foi, or, (b) each other.

3. If there is any rejoining flow in step (2) then the network is not tree-reducible, otherwise, it is and the remaining flows and servers form an in-tree whose root is the last server traversed by the foi.

While not being perfectly formal, it should be clear from this procedure that tree-reducibility imposes clearly a restriction to general feedforward networks, simply speaking by prohibiting rejoining flows. Nevertheless, it covers quite a number of cases and generalizes significantly on previous work in SNC network analysis. In Section 5, we come back to the issue of general feedforward networks *with* rejoining flows and how we can leverage from the PMOO result for tree networks in that case.

Two elementary operations to analyse networks in network calculus are computing residual service offered to the flow of interest for a server crossed by multiple flows and the concatenation of servers when a flow traverses multiple servers. Let us recall the main results from the literature concerning MGF-based network calculus.
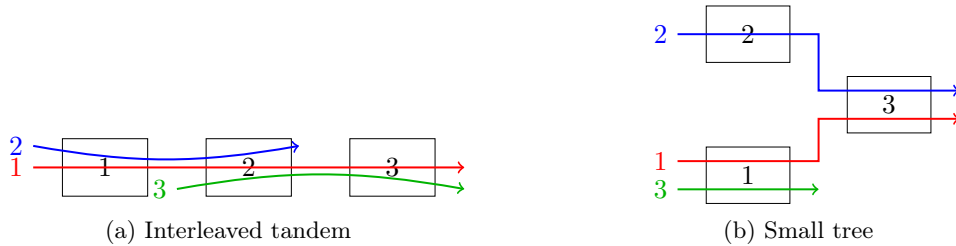
(a) Interleaved tandem　　　　　　　　(b) Small tree

Figure 3: Two example networks.

**Theorem 3** (Residual server [20]). *Consider a work-conserving $S$-server traversed by two flows with respective bivariate arrival processes $A_1$ and $A_2$. The residual server of flow 1 can be characterized by a dynamic $S_1$-server with*

$$S_1(s,t) = [S(s,t) - A_2(s,t)]_+, \ \forall s \le t,$$

*where $[x]_+ = \max(0, x)$.*

Here, we make no assumption about the service policy regarding the sharing between flows. This is also known as arbitrary or blind multiplexing.

**Theorem 4** (End-to-end server [20]). *Consider a flow traversing a tandem of dynamic $S_i$-servers, $i \in \{1, \ldots, n\}$. The overall service offered is a dynamic $S_{\mathrm{e2e}}$-server with $S_{\mathrm{e2e}} = S_1 \otimes \cdots \otimes S_n$.*

In this section, we generalize and combine these two central theorems for tree networks. We first bound the service process for the end-to-end dynamic server offered to a flow traversing a network in Subsection 3.1. Next, in Subsection 3.2 we derive the corresponding bounding generating function and apply the framework from Subsection 2.3 to derive performance bounds.

## 3.1 A Pay-Multiplexing-Only-Once formula for bivariate processes

In this section, we consider a tree network. More precisely, the network can be described as follows.

1. The network is composed of $n$ servers, numbered from 1 to $n$ according to the topological order: if server $h$ is the successor of server $j$, then $h > j$. The topology is a tree directed to server $n$: each server $j \ne n$ has a unique successor that we denote by $j^\bullet$. Each server $j$ is work-conserving and offers the service $S_j(s,t)$ during the time slots $s+1, \ldots, t$. By convention, we write $n^\bullet = n+1$.

2. There are $m$ flows in the network, numbered from 1 to $m$. We denote $\pi_i = \langle \pi_i(1), \ldots, \pi_i(\ell_i) \rangle$ the path, i.e. the sequence of servers, of length $\ell_i \in \mathbb{N} \setminus \{0\}$ followed by flow $i$ and $A_i(s,t)$ is the amount traffic of flow $i$ arriving in the network during the time slots $s+1, \ldots, t$. We write $i \in j$ if flow $i$ crosses server $j$ (or equivalently $j \in \pi_i$).

3. We denote $A_i^{(j)}(s,t)$ the bivariate process of flow $i$ at the input of server $j$, and consequently, $A_i^{(j^\bullet)}(s,t)$ is the bivariate process at the output of server $j$. Note that we have $A_i = A_i^{(\pi_i(1))}$.

**Example 5.** The networks of Figures 3a and 3b are two examples of tree networks. They both have three flows and three servers. In the network of Figure 3a, we have $1^\bullet = 2$, $2^\bullet = 3$ and $3^\bullet = 4$, whereas in the network of Figure 3b, we have $1^\bullet = 2^\bullet = 3$ and $3^\bullet = 4$. Graphically, the relation $j^\bullet = h$ means that $(j, h)$ is an edge of the network.

We can now state the main result of this section. Theorem 6 also generalizes that of [8, 33, 34] in deterministic network calculus from tandems to trees. More precisely, all the results in those references assume that the flow for which we compute the residual service curve traverses the entire

tandem network. Here we relax this assumption. However, the principle of the proof remains the same, and our result is directly applicable to deterministic network calculus. For the sake of self-containedness, we give a proof in Appendix B.1.

**Theorem 6.** *With the notations above, assume that the last server traversed by flow 1 is $n$. The residual service offered to flow 1 is a dynamic $S_{\mathrm{e2e}}$-server with $\forall 0 \leq t_{\pi_1(1)} \leq t_{n+1}$,*

$$S_{\mathrm{e2e}}(t_{\pi_1(1)}, t_{n+1}) = \left[ \inf_{\forall j,\ t_j \leq t_{j\bullet}} \sum_{j=1}^{n} [S_j(t_j, t_{j\bullet})] - \sum_{i=2}^{m} A_i(t_{\pi_1(1)}, t_{\pi_i(\ell_i)\bullet}) \right]_+ .$$

Note that there is a parameter $t_j$ for each server $j$ following the topological order of the servers.

**Example 7.** The end-to-end server for flow 1 for the network in Figure 3a is

$$S_{\mathrm{e2e}}(t_1, t_4) = \left[ \inf_{t_1 \leq t_2 \leq t_3 \leq t_4} (S_1(t_1, t_2) + S_2(t_2, t_3) + S_3(t_3, t_4) - A_2(t_1, t_3) - A_3(t_2, t_4)) \right]_+ .$$

In this formula, the infimum is computed at all times $t_2$ and $t_3$ satisfying the conditions $t_1 \leq t_2 \leq t_3 \leq t_4$ ($t_1$ and $t_4$ are fixed), according to the network's topology.

The end-to-end server for flow 1 for the network in Figure 3b is

$$S_{\mathrm{e2e}}(t_1, t_4) = \left[ \inf_{\substack{t_1 \leq t_3 \leq t_4, \\ t_2 \leq t_3}} (S_1(t_1, t_3) + S_2(t_2, t_3) + S_3(t_3, t_4) - A_2(t_2, t_4) - A_3(t_1, t_3)) \right]_+ .$$

Here again, the infimum is computed for all $t_2$ and $t_3$, but the conditions change with the network topology: no lower bound is enforced for $t_2$ as server 2 has no predecessor, that is still upper bounded by $t_3$ and the new bounds for $t_3$ become $t_1 \leq t_3 \leq t_4$, as $(1, 3)$ is an edge of the network.

*Proof of Theorem 6.* Consider server $j$. For all $t_j \leq t_{j\bullet}$ in the same backlogged period, $\sum_{i \in j} [A_i^{(j\bullet)}(0, t_{j\bullet}) - A_i^{(j\bullet)}(0, t_j)] \geq S_j(t_j, t_{j\bullet})$. In particular, this formula is true when $t_j = start_j(t_{j\bullet})$, the start of the backlogged period of $t_{j\bullet}$ at server $j$. In that case $A_i^{(j\bullet)}(0, t_j) = A_i^{(j)}(0, t_j)$, and

$$\sum_{i \in j} [A_i^{(j\bullet)}(0, t_{j\bullet}) - A_i^{(j)}(0, t_j)] \geq S_j(t_j, t_{j\bullet}).$$

Summing over all servers $j$, we obtain

$$\sum_{j=1}^{n} \Big( \sum_{i \in j} [A_i^{(j\bullet)}(0, t_{j\bullet}) - A_i^{(j)}(0, t_j)] \Big) \geq \sum_{j=1}^{n} S_j(t_j, t_{j\bullet}).$$

Now, by exchanging the two sums on the left-hand side, most of the terms cancel, and (remind that $A_i^{(\pi_i(1))} = A_i$)

$$\sum_{i=1}^{m} [A_i^{(\pi_i(\ell_i)\bullet)}(0, t_{\pi_i(\ell_i)\bullet}) - A_i(0, t_{\pi_i(1)})] \geq \sum_{j=1}^{n} S_j(t_j, t_{j\bullet}).$$

Keeping only $A_1^{(\pi_i(\ell_i)\bullet)}(0, t_{\pi_i(\ell_i)\bullet})$ on the left-hand side of the inequality and using $A_i^{(\pi_i(\ell_i)\bullet)}(0, \cdot) \leq A_i(0, \cdot)$, we obtain

$$A_1^{(\pi_1(\ell_1)\bullet)}(0, t_{\pi_1(\ell_1)\bullet}) \geq A_1(0, t_{\pi_1(1)}) + \sum_{j=1}^{n} S_j(t_j, t_{j\bullet}) - \sum_{i=2}^{m} [A_i^{(\pi_i(\ell_i)\bullet)}(0, t_{\pi_i(\ell_i)\bullet}) - A_i(0, t_{\pi_i(1)})]$$

$$\geq A_1(0, t_{\pi_1(1)}) + \sum_{j=1}^{n} S_j(t_j, t_{j\bullet}) - \sum_{i=2}^{m} A_i(t_{\pi_i(1)}, t_{\pi_i(\ell_i)\bullet}),$$

We also have $\forall j \in \pi_1$, $A_1^{(j)}(0, t_j) = A_1^{(j^\bullet)}(0, t_j) \le A_1^{(j^\bullet)}(0, t_{j^\bullet})$. So $A_1^{(\pi_1(\ell)^\bullet)}(0, t_{\pi_1(\ell_1)^\bullet}) \ge A_1(0, t_{\pi_1(1)})$.

The final result follows by taking the minimum on all possible values of $t_j$, $j \notin \{\pi_1(1), \pi_1(\ell_1)^\bullet\}$, and noticing that $\pi_1(\ell_1)^\bullet = n + 1$. $\qquad\square$

This result is valid for any bivariate arrival and service processes. The next subsection demonstrates its use in the analytic combinatorics framework. Before that, let us briefly discuss the power of Theorem 6: Looking at the end-to-end server calculation, it can be noted that all arrival and service processes appear in the formula as they originally enter the network. We point out that the tree generalization is crucial here, because it avoids the need for characterizing internal flows that may have become dependent after sharing a server. Hence, this calculation introduces *no method-pertinent dependencies* at all. Moreover, the end-to-end server calculation is performed in one big step, in contrast to a sequential application of network calculus operations. This is already known to *avoid a sequencing penalty* [2], yet completely avoiding it by performing all operations *simultaneously* is only possible now with Theorem 6.

## 3.2 Bounding generating function of the end-to-end server

In this subsection, we assume that

$(H_1)$ all processes and servers are mutually independent;

$(H_2)$ flow $A_i$ is $(\sigma_{A_i}, \rho_{A_i})$-constrained for all flow $i \in \{1, \dots, m\}$ and $S_j$ is $(\sigma_{S_j}, \rho_{S_j})$-constrained for all server $j \in \{1, \dots, n\}$.

We can now derive a service bounding generating function for the end-to-end dynamic server.

**Theorem 8.** *Under hypothesis $(H_1)$ and $(H_2)$, the end-to-end service for flow 1 is bounded by the service bounding generating function*

$$F_{S_{e2e}}(\theta, z) = \frac{e^{\theta(\sum_{i=2}^m \sigma_{A_i}(\theta) + \sum_{j=1}^n \sigma_{S_j}(\theta))}}{\prod_{j \notin \pi_1}(1 - e^{-\theta(\rho_{S_j}(\theta) - \sum_{i \in j} \rho_{A_i}(\theta))})} \prod_{j \in \pi_1} \frac{1}{1 - e^{-\theta(\rho_{S_j}(\theta) - \sum_{1 \ne i \in j} \rho_{A_i}(\theta))} z}.$$

*Proof.* From Theorem 6, we can derive a bound of the Laplace transform of the service for flow 1. For the sake of concision, in the last line, we omit the function argument $\theta$.

$$\mathbf{E}[e^{-\theta S_{e2e}(t_{\pi_1(1)}, t_{n+1})}] \le \sum_{\forall j,\ t_j \le t_{j^\bullet}} \mathbf{E}[e^{-\theta(\sum_{j=1}^n S_j(t_j, t_{j^\bullet}) - \sum_{i=2}^m A_i(t_{\pi_i(1)}, t_{\pi_i(\ell_i)^\bullet}))}]$$

$$= \sum_{\forall j,\ t_j \le t_{j^\bullet}} \prod_{j=1}^n \mathbf{E}[e^{-\theta S_j(t_j, t_{j^\bullet})}] \prod_{i=2}^m \mathbf{E}[e^{\theta A_i(t_{\pi_i(1)}, t_{\pi_i(\ell_i)^\bullet})}] \qquad (6)$$

$$\le \sum_{\forall j,\ t_j \le t_{j^\bullet}} \prod_{j=1}^n e^{\theta(\sigma_{S_j}(\theta) - \rho_{S_j}(\theta)(t_j^\bullet - t_j))} \prod_{i=2}^m e^{\theta(\sigma_{A_i}(\theta) + \rho_{A_i}(\theta)(t_{\pi_i(\ell_i)^\bullet} - t_{\pi_i(1)}))}$$

$$\le \sum_{\substack{\forall j,\ u_j \ge 0, \\ \sum_{j \in \pi_1} u_j = t_{n+1} - t_{\pi_1(1)}}} \prod_{j=1}^n e^{\theta(\sigma_{S_j}(\theta) - \rho_{S_j}(\theta) u_j)} \prod_{i=2}^m e^{\theta(\sigma_{A_i}(\theta) + \rho_{A_i}(\theta)(\sum_{j \in \pi_i} u_j))}$$

$$= e^{\theta(\sum_{i=2}^m \sigma_{A_i}(\theta) + \sum_{j=1}^n \sigma_{S_j}(\theta))} \sum_{\substack{\forall j,\ u_j \ge 0, \\ \sum_{j \in \pi_1} u_j = t_{n+1} - t_{\pi_1(1)}}} \prod_{j=1}^n e^{-\theta(\rho_{S_j}(\theta) - \sum_{1 \ne i \in j} \rho_{A_i}(\theta)) u_j}$$

$$= \frac{e^{\theta(\sum_{i=2}^m \sigma_{A_i} + \sum_{j=1}^n \sigma_{S_j})}}{\prod_{j \notin \pi_1}(1 - e^{-\theta(\rho_{S_j} - \sum_{i \in j} \rho_{A_i})})} \sum_{\sum_{j \in \pi_1} u_j = t_{n+1} - t_{\pi_1(1)}} \prod_{j \in \pi_1} e^{-\theta(\rho_{S_j} - \sum_{1 \ne i \in j} \rho_{A_i}) u_j}.$$

11

In Equation (6), we use the independence of the arrivals and service; in the third line, the constraints on the processes; in the fourth line, we changed the variables: $u_j = t_j^\bullet - t_j$ (hence $t_{\pi_i(\ell_i)^\bullet} - t_{\pi_i(1)} = \sum_{j \in \pi_i} u_j$), and in the sixth line summed the terms not constrained by $\sum_{j \in \pi_1} u_j = t_{n+1} - t_{\pi_1(1)}$.

We can recognize the terms of the Cauchy product of geometric series, hence the service is bounded by the generating function

$$F_{S_{e2e}}(\theta, z) = \frac{e^{\theta(\sum_{i=2}^m \sigma_{A_i}(\theta) + \sum_{j=1}^n \sigma_{S_j}(\theta))}}{\prod_{j \notin \pi_1} \left(1 - e^{-\theta(\rho_{S_j}(\theta) - \sum_{i \in j} \rho_{A_i}(\theta))}\right)} \prod_{j \in \pi_1} \frac{1}{1 - e^{-\theta(\rho_{S_j}(\theta) - \sum_{1 \neq i \in j} \rho_{A_i}(\theta))} z}.$$

$\square$

**Example 9.** A service bounding generating function for the end-to-end server for flow 1 in the network of Figure 3a is

$$F_{e2e}(\theta, z) = \frac{e^{\theta(\sigma_{A_2} + \sigma_{A_3} + \sigma_{S_1} + \sigma_{S_2} + \sigma_{S_3})}}{(1 - e^{-\theta(\rho_{S_1} - \rho_{A_2})} z)(1 - e^{-\theta(\rho_{S_2} - \rho_{A_2} - \rho_{A_3})} z)(1 - e^{-\theta(\rho_{S_3} - \rho_{A_3})} z)}.$$

A service bounding generating function for the end-to-end server for flow 1 in the network of Figure 3b is

$$F_{e2e}(\theta, z) = \frac{e^{\theta(\sigma_{A_2} + \sigma_{A_3} + \sigma_{S_1} + \sigma_{S_2} + \sigma_{S_3})}}{1 - e^{-\theta(\rho_{S_2} - \rho_{A_2})}} \frac{1}{1 - e^{-\theta(\rho_{S_1} - \rho_{A_3})} z} \frac{1}{1 - e^{-\theta(\rho_{S_3} - \rho_{A_2})} z}.$$

Note the slight difference between the two formulas: informally, there is a '$z$' variable only for factors corresponding to the servers crossed by flow 1.

The end-to-end service process is not necessarily $(\sigma, \rho)$-constrained. Corollary 2 below gives a simpler bounding generating function, a proof is given in Appendix B.1. In short, $G(\theta, z)$, defined in the statement of Corollary 2, is a bounding generating function of $F_{e2e}(\theta, z)$ obtained by bounding its factors not related to the dominant singularity.

Let us first alleviate the notations and denote

$$F_{S_{e2e}}(\theta, z) = e^{\theta \sigma_{S_{e2e}}(\theta)} \prod_{j \in \pi_1} \frac{1}{1 - e^{-\theta \rho_j'(\theta)} z},$$

with

$$e^{\theta \sigma_{S_{e2e}}(\theta)} = \frac{e^{\theta(\sum_{i \neq 1} \sigma_{A_i}(\theta) + \sum_j \sigma_{S_j}(\theta))}}{\prod_{j \notin \pi_1} \left(1 - e^{-\theta(\rho_{S_j}(\theta) - \sum_{i \in j} \rho_{A_i}(\theta))}\right)} \quad \text{and} \quad \rho_j'(\theta) = \rho_{S_j}(\theta) - \sum_{1 \neq i \in j} \rho_{A_i}(\theta),$$

and assume (without loss of generality, by renumbering the servers) that $\rho_j'(\theta) = \rho_1'(\theta)$ for all $j \leq k$ and $\rho_j'(\theta) > \rho_1'(\theta)$ for all $j > k$.

**Corollary 2.** $G(\theta, z) = \frac{e^{\theta \sigma_{S_{e2e}}(\theta)}}{\prod_{j=k+1}^m \left(1 - e^{-\theta(\rho_j'(\theta) - \rho_1'(\theta))}\right)} \left(\frac{1}{1 - e^{-\theta \rho_1'(\theta)} z}\right)^k$ *is a bounding generating function of* $F_{S_{e2e}}(\theta, z)$.

## 3.3 Performance bounds

We can now apply the results of Subsection 2.3 to the end-to-end dynamic server. In the following, we assume stability, that is we consider only values of $\theta$ such that for all $j \in \{1, \ldots, n\}$, $\rho_j'(\theta) > \rho_{A_1}(\theta)$ (the residual rate at server $j$ is larger than the arrival rate).

**Output departure process and backlog bound**   With Lemma 1, we can compute an arrival bounding generating function for the departure process:

$$F_D(\theta, z) = \frac{e^{\theta(\sigma_{A_1}(\theta)+\sigma_{S_{\text{e2e}}}(\theta))}}{\prod_{j\in\pi_1}\left(1 - e^{\theta(\rho_{A_1}(\theta)-\rho'_j(\theta))}\right)} \frac{1}{1 - e^{\theta\rho_{A_1}(\theta)}z}.$$

We recognize a $(\sigma, \rho)$-constraint for the departure process, and observe that $\rho_D(\theta) = \rho_A(\theta)$.

From Corollary 1, the backlog bound for flow 1 is then

$$\mathbf{P}(q(t) \ge b) \le \frac{e^{\theta(\sigma_{A_1}(\theta)+\sigma_{S_{\text{e2e}}}(\theta))}}{\prod_{j\in\pi_1}(1 - e^{\theta(\rho_{A_1}(\theta)-\rho'_j(\theta))})}e^{-\theta b}.$$

**Delay bound**   It is not as straightforward to compute the delay bound: the bounding generating function of the service does not correspond to a $(\sigma, \rho)$-constraint. The bounding generating function for the delay is

$$F_d(\theta, z) = e^{\theta\sigma_{A_1}(\theta)}\frac{e^{\theta\rho_{A_1}(\theta)}F_{S_{\text{e2e}}}(\theta, e^{\theta\rho_{A_1}(\theta)}) - zF_{S_{\text{e2e}}}(\theta, z)}{1 - e^{-\theta\rho_{A_1}(\theta)}z}. \tag{7}$$

As $F_{S_{\text{e2e}}}(\theta, z)$ is a rational function, $F_d(\theta, z)$ is. Moreover, all the singularities of these functions are known. Note that $e^{\theta\rho_{A_1}(\theta)}$ is not a singularity of $F_d(\theta, z)$. Indeed, when $z = e^{-\theta\rho_{A_1}(\theta)}$, the numerator of Equation (7) is zero. The singularities of $F_d(\theta, z)$ are then those of $F_{S_{\text{e2e}}}(\theta, z)$, that is $e^{\theta\rho'_j(\theta)}$, for $j \in \{1, \dots, n\}$. All the terms of the generating function of Equation (7) can be computed automatically with symbolic computation tools such as SageMath [35]. However, there are several cases where the exact expression is compact enough to be given here.

- Residual rates are all distinct (see derivation in Appendix B.2): $\forall T \ge 0$,

$$\mathbf{P}(d(t) \ge T) \le [z^T]F_d(\theta, z) = \sum_{j=1}^{n}\frac{e^{\theta(\sigma_{A_1}(\theta)+\sigma_{S_{\text{e2e}}}(\theta)+\rho_{A_1}(\theta))}}{1 - e^{\theta(\rho_{A_1}(\theta)-\rho'_j(\theta))}} \cdot \left[\prod_{k\ne j}\frac{1}{1 - e^{\theta(\rho'_j(\theta)-\rho'_k(\theta))}}\right]e^{-\theta\rho'_j(\theta)T}. \tag{8}$$

- Residual rates are all equal (see derivation in Appendix B.3): $\forall T > 0$,

$$\mathbf{P}(d(t) \ge T) \le [z^T]F_d(\theta, z) = e^{\theta(\sigma_{A_1}(\theta)+\sigma_{S_{\text{e2e}}}(\theta)+\rho_{A_1}(\theta))} \cdot \left[\sum_{i=1}^{n}\binom{T+i-2}{T-1}\frac{e^{-\theta\rho'_1(\theta)T}}{\left(1 - e^{-\theta(\rho'_1(\theta)-\rho_{A_1}(\theta))}\right)^{n-i+1}}\right]. \tag{9}$$

In the general case, one can first use the service bounding generating function of Corollary 2 to be in the case of equal residual rates, and then use Equation (9) to obtain $\forall T > 0$,

$$\mathbf{P}(d(t) \ge T) \le \frac{e^{\theta(\sigma_{A_1}(\theta)+\sigma_{S_{\text{e2e}}}(\theta)+\rho_{A_1}(\theta))}}{\prod_{j=k+1}^{n}1 - e^{-\theta(\rho'_j(\theta)-\rho'_1(\theta))}} \cdot \left[\sum_{i=1}^{k}\binom{T+i-2}{T-1}\frac{e^{-\theta\rho'_1(\theta)T}}{\left(1 - e^{-\theta(\rho'_1(\theta)-\rho_{A_1}(\theta))}\right)^{k-i+1}}\right]. \tag{10}$$

## 3.4   Recovering existing results for the canonical tandem

In this subsection, we recover the state of the art in SNC for the "canonical tandem" (Figure 4), as it is frequently analysed in the literature, e.g. [11, 20, 32]. Since this simple flow interference structure provides no opportunity for the PMOO principle to pay off, our new bounds based on analytic combinatorics cannot further improve it (in contrast to the scenarios we present in Section 4). To the best of our knowledge, the best technique for this topology is to calculate the optimum of
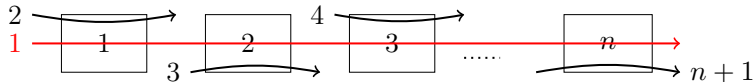
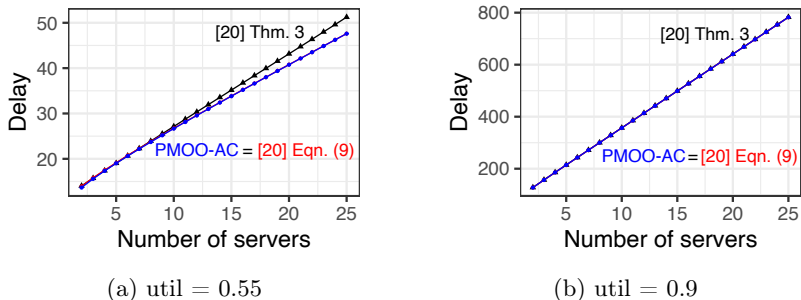Figure 4: Canonical tandem.



(a) util = 0.55

(b) util = 0.9

Figure 5: Stochastic delay bounds for the canonical tandem (for traffic with independent exponentially distributed increments with $\lambda = 1$ and constant rate servers).

two bounds originally derived in [20, Theorem 3], denoted by "[20] Thm. 3" below. However, this bound targets a closed-form solution for the stochastic delay and, hence, compromises on tightness. Attempting a fair comparison, we also compare the novel result from Equation (10) to a tighter version of "[20] Thm. 3" that does not make this compromise and is given in [20] Eqn. (9) as an intermediate result. One can show that our Equation (10) recovers the result in [20] Eqn. (9), since both start with the same inequalities and, afterwards, only equivalent term manipulations are used.

We also verified this numerically for exponential i.i.d. arrivals in Figure 5. The closed-form solution in "[20] Thm. 3" tends to lead to less accurate bounds, depending on the utilization as well as the number of servers. We note that it is the only approach that exploits the positive part in Theorem 3, but this effect is negligible for the chosen traffic class. Anyway, the positive part can also be incorporated in the analytic combinatorics; yet, we omit this here for the sake of brevity.

## 3.5 Relation to state-of-the-art analyses

A typical network calculus analysis to derive end-to-end performance bounds consists of three separate sequential steps:

1. Reducing the network to a tandem traversed by the flow of interest, by output bounding.

2. Reducing the tandem to an end-to-end server that represents the complete system.

3. Computation of performance bounds.

Various SNC analysis techniques have been derived in the literature. In [4], a *sequential* separated flow analysis (SFA) is used, where each network calculus operation is applied one after the other in contrast to a simultaneous technique as in Theorem 6. On the other hand, it has been shown in [20] that combining step (2) and (3) is able to avoid a sequencing penalty to some degree (see also [2]). Moreover, one could also convolve servers as much as possible before subtracting cross-flows. In [29], this is called "PMOO", however, it is not fully able to use the power of the PMOO principle, as it involves a sequential order of network calculus operations which is why we call it "seqPMOO" in the following.

However, none of the above mentioned techniques is able to avoid a sequencing penalty entirely. On the other hand, the newly introduced PMOO analysis avoids this pitfall by combining all three steps of the analysis and also reduces the overall number of calculations. In the next section (Section 4), we show that this approach leads to significantly more accurate delay bounds.

14

# 4   Numerical Evaluation of Tree Networks

In this section, we compute bounds on the delay's violation probability and stochastic delay bounds applying state-of-the-art techniques as well as our unleashed PMOO. We perform several experiments for different network topologies. For the arrivals, we assume three discrete-time processes all adhering to the class of $(\sigma_A, \rho_A)$-constrained arrivals: Exponentially distributed arrivals increments [2] with parameter $\lambda$, Weibull distributed arrival increments with fixed shape parameter $k = 2$ and scale parameter $\lambda$, and discrete-time Markov-modulated On-Off (MMOO) arrivals [10, 2]. The latter can be described by three parameters: the probability to stay in the "On"-state in the next time step, $p_{\text{on}}$, the probability to stay in the "Off"-state, $p_{\text{off}}$, and a constant peak rate $P$, at which data is sent during the "On"-state. For the service, we always assume work-conserving $S$-servers with a constant rate.

## 4.1   Interleaved tandem

In our first numerical evaluation, we calculate stochastic delay bounds for a standard example in network calculus when PMOO effects shall be illustrated: the interleaved tandem network in Figures 3a. These are compared to simulation results. The "standard bound" consists of the minimum of the state-of-the-art techniques described in Section 3.5. They are then compared to our new analytic combinatorics based PMOO using two variants: The first, again based on Equation (10), is called "PMOO-AC: General Bound" in this example. The second, "PMOO-AC: Explicit Bound" exploits that we choose all parameters such that the residual rates are distinct. Therefore, we can apply Equation (8). In addition, we provide simulation results with respective pointwise 95%-confidence bands that are based on order statistics of a binomially distributed sample [25] to see how the bounds relate to the empirical delay. The results are provided in Figure 6.

We observe that, while the PMOO-AC bounds lead to quite similar results, they both outperform the standard bound significantly. For example, for a delay violation probability of $10^{-3}$, the delay bound is improved from 28 to 18, and for a violation probability of $10^{-7}$, from 45 to 31. These examples indicate an improvement of more than 35%.

These positive results are mainly caused by the fact that the PMOO analysis is able to provide bounds without introducing any method-pertinent dependencies. The standard bound, in contrast, suffers from such a dependency in the calculation and, consequently, needs to apply Hölder's inequality. Furthermore, it also potentially loses accuracy due to the sequencing penalty.

In addition, we see that the gap between simulation results and bounds is considerably reduced and the scaling of the delay in the simulations is captured well (getting closer to single-node results again). Recalling that Figure 6a without the new bounds has been used in Section 1 to illustrate that the known SNC gap will widen too much, we now provide a new prospect for this again.
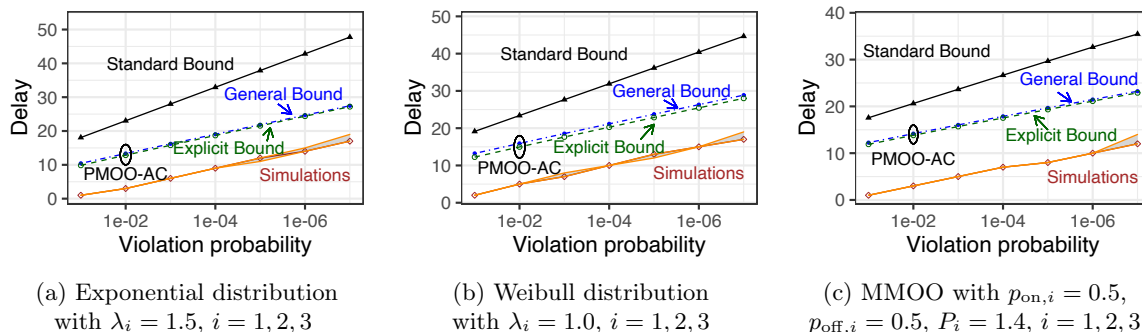


(a) Exponential distribution with $\lambda_i = 1.5$, $i = 1, 2, 3$

(b) Weibull distribution with $\lambda_i = 1.0$, $i = 1, 2, 3$

(c) MMOO with $p_{\text{on},i} = 0.5$, $p_{\text{off},i} = 0.5$, $P_i = 1.4$, $i = 1, 2, 3$

Figure 6: Delay bounds and simulations for the interleaved tandem with server rates $C_1 = 2.5$, $C_2 = 3.0$, $C_3 = 2.0$.

Figure 7: Extended interleaved tandem.
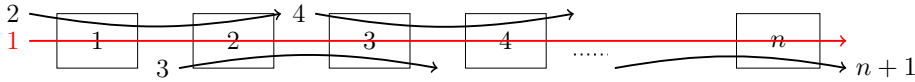


(a) Exponential distribution with $\lambda_i = 2.0$

(b) Weibull distribution with $\lambda_i = 0.7$

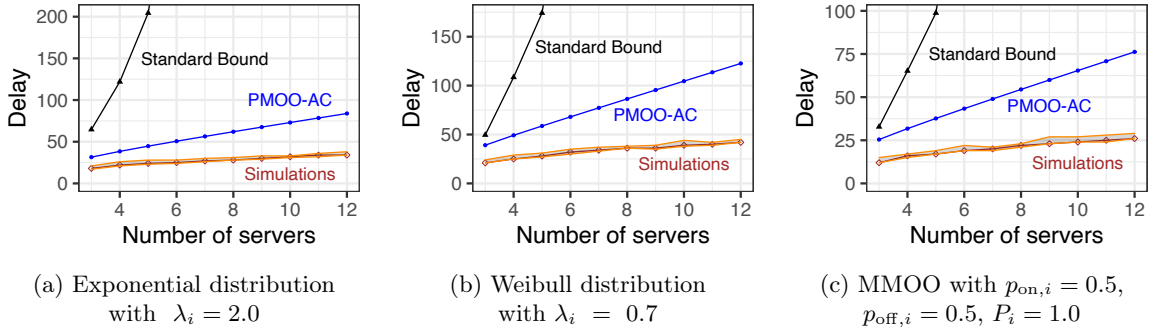(c) MMOO with $p_{\mathrm{on},i} = 0.5$, $p_{\mathrm{off},i} = 0.5$, $P_i = 1.0$

Figure 8: Delay bounds for the extended interleaved tandem with server rates $C_i = 2.0$ for $i = 1, \ldots, 12$.

## 4.2 Extended interleaved tandem

In the next experiment, we generalize the case of an interleaved tandem by varying its lengths while keeping the interference structure (Figure 7). Here, the standard bound only includes the sequential and simultaneous SFA approaches, as the number of possible (combinations of) network calculus operations in the seqPMOO grows exponentially with the number of servers and, therefore, is computationally prohibitive.

In Figure 8, we show stochastic delay bounds for a fixed violation probability of $10^{-6}$. While the standard bound explodes in the number of servers (we have only included the results from 3 to 5 servers), the new technique scales significantly better. This is mainly due to the fact that the SFA leads to the application of $n-1$ Hölder inequalities for an interleaved tandem of size $n$, whereas the new PMOO-AC does not incur any method-pertinent dependencies in the analysis of this topology.

Not only does this lead to tighter performance bounds, it also impacts runtimes significantly as it reduces an $n$-dimensional non-linear optimization problem (one $\theta$ and $n-1$ Hölder parameters) to a 1-dimensional optimization problem.[1] Specifically, for the standard bound (in the case of MMOO arrivals), runtimes increase quickly from 5.2 seconds (3 servers) over 2 minutes (4 servers) to roughly 1.5 hours (5 servers). On the other hand, PMOO-AC took a maximum runtime of 0.46 seconds in the case of 12 servers.

## 4.3 Case study: tree network

In Section 3.5, we explained that the PMOO-AC calculates performance bounds by combining all the steps of the analysis into a single large one. The following case study shall mainly investigate the benefit of this.

---

[1]In our numerical experiments, we apply a grid search followed by a downhill simplex algorithm to optimise the parameters.
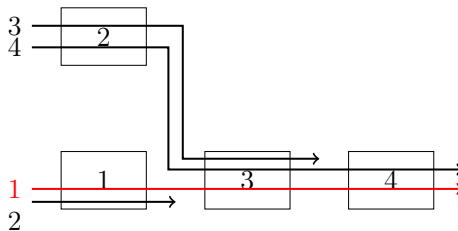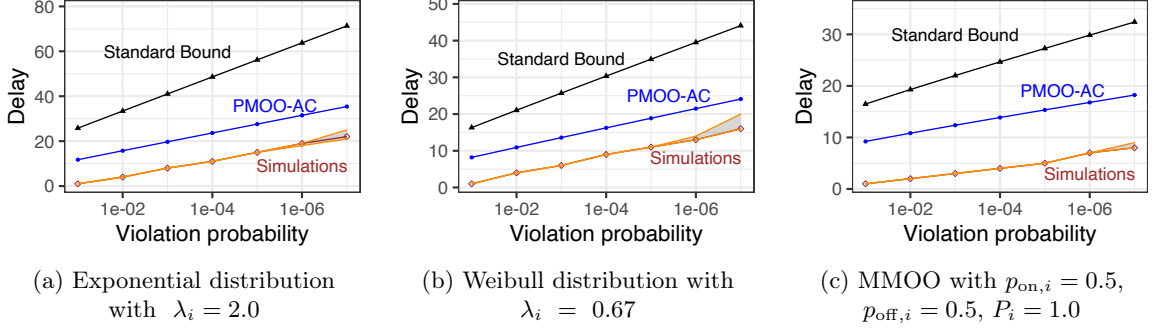


Figure 9: Case study tree network.

(a) Exponential distribution
with $\lambda_i = 2.0$

(b) Weibull distribution with
$\lambda_i = 0.67$

(c) MMOO with $p_{\mathrm{on},i} = 0.5$,
$p_{\mathrm{off},i} = 0.5$, $P_i = 1.0$

Figure 10: Delay bounds and simulations for the tree network with server rates $C_i = 2.0$, $i = 1, \ldots, 4$.

To that end, we consider the tree network in Figure 9. Standard SNC techniques first compute the output bounds of flows 3 and 4 at server 2 in order to compute the residual service at servers 3 and 4; this incurs again method-pertinent dependencies. The application of Theorem 6 allows us to circumvent these dependencies. In contrast, state-of-the-art analysis using SFA needs three applications of Hölder's inequality and seqPMOO requires one, respectively.

The results are shown in Figure 10. Similar to the results for the interleaved tandem, we observe that the PMOO considerably improves the bound on the delay's violation probability. Again, we are able to achieve a similar scaling compared to the simulation results, in contrast to the standard bound. Further, while the state-of-the-art analysis requires an optimisation of up to 4 parameters (three Hölder and $\theta$), the PMOO only has $\theta$ to optimise, substantially improving the runtime.

# 5 Towards Feedforward Network Analysis under Partial Dependence

In previous sections, we focused on mutually independent flows and servers in tree-reducible networks. However, if we want to analyse general feedforward networks, rejoining flows eventually force us to deal with dependencies. As discussed in Section 3, in the presence of rejoining flows, the network will not be tree-reducible any longer. Thus, we cannot completely avoid method-pertinent dependencies using PMOO, but need to extend it to the partially dependent case.

As mentioned in Section 2, the use of Hölder's inequality allows us to calculate performance bounds also in the dependent case. Yet, as we discuss in this section, its number of applications as well as their induced inaccuracy can be influenced by the transformation to a tree-reducible network – our goal is to minimize the impact of Hölder's inequality on the performance bounds.

First, in Subsection 5.1, we extend PMOO under partial dependence; next, in Subsection 5.2, we apply this to the canonical example of rejoining flows: the "diamond network". Finally, in Subsection 5.3, we discuss how to extend these promising results to larger feedforward networks.

## 5.1 End-to-end server and performance bounds

To deal with the partial dependency of random variables, we adapt the notion of *dependency graph* defined to prove the Lovász Local Lemma (see [27], for instance). Let $(X_h)_{h \in H}$ be a finite family of processes. In our case a process can be a service process $S_j$ or an arrival process $A_i$. The dependency graph associated to $(X_h)_{h \in H}$ is the graph whose vertices are the processes $(X_h)_{h \in H}$ and process $X_h$ is not adjacent with processes with which it is mutually independent. More precisely, if $\Gamma(X_h)$ is the set of neighbours of $X_h$, $X_h$ is mutually independent of $\{X_{h'} \mid X_{h'} \notin (\Gamma(X_h) \cup \{X_h\})\}$.

Assume that the dependency graph contains $K$ strongly connected components $G_1, \ldots, G_K$, and that for all $h \in H$, and for some $\theta > 0$, $\mathbf{E}[e^{\theta X_h(s_h, t_h)}] \leq e^{\theta \lambda_h(\theta, s_h, t_h)}$. The following lemma shows

how to use as few Hölder inequalities as possible.

**Lemma 4.** *With the hypotheses and notations above, for positive $(p_h)_{h \in H}$ such that $\forall k \in \{1, \ldots, K\}$, $\sum_{h \in G_k} \frac{1}{p_h} = 1$,*

$$\mathbf{E}[e^{\theta(\sum_{h \in H} X_h(s_h, t_h))}] \leq \prod_{h \in H} e^{\theta \lambda_h(p_h \theta, s_h, t_h)}.$$

*Proof.*

$$\mathbf{E}[\prod_{h \in H} e^{\theta X_h(s_h, t_h)}] = \prod_{k=1}^{K} \mathbf{E}[\prod_{h \in G_k} e^{\theta X_h(s_h, t_h)}] \leq \prod_{k=1}^{K} \prod_{h \in G_k} (e^{p_h \theta \lambda_h(p_h \theta, s_h, t_h)})^{1/p_h} = \prod_{h \in H} e^{\theta \lambda_h(p_h \theta, s_h, t_h)}.$$

The first equality is deduced from the dependency graph: we have mutual independence between families of processes of distinct connected components. The inequality is the application of the Hölder inequality inside the connected components, as well as the use of the MGF bounds. □

We can now derive an end-to-end service for partially dependent servers and arrival processes.

**Theorem 10.** *The end-to-end service for flow 1 is bounded by the service bounding generating function*

$$F_{S_{e2e}}(\theta, (p_i), (q_j), z) = \frac{e^{\theta(\sum_{i \neq 1} \sigma_{A_i}(p_i \theta) + \sum_j \sigma_{S_j}(q_j \theta))}}{\prod_{j \notin \pi_1}(1 - e^{-\theta(\rho_{S_j}(q_j \theta) - \sum_{i \in j} \rho_{A_i}(p_i \theta))})} \prod_{j \in \pi_1} \frac{1}{1 - e^{-\theta(\rho_{S_j}(q_j \theta) - \sum_{1 \neq i \in j} \rho_{A_i}(p_i \theta))} z},$$

*for all $(p_i), (q_j)$ such that for all $k \leq K$, $\sum_{A_i \in G_k \setminus \{1\}} \frac{1}{p_i} + \sum_{S_j \in G_k} \frac{1}{q_j} = 1$.*

*Proof.* The proof follows almost along the same lines as in Theorem 8. Lemma 4 is used at Equation (6). If $X_h = A_i$, we have $s_h = t_{\pi_i(1)}$, $t_h = t_{\pi_i(\ell_i)}$ and $\lambda_h(\theta, s_h, t_h) = \sigma_{A_i}(\theta) + \rho_{A_i}(\theta)(t_h - s_h)$. If $X_h = S_j$, we have $s_h = t_j$, $t_h = t_{j\bullet}$ and $\lambda_h(\theta, s_h, t_h) = \sigma_{S_j}(\theta) - \rho_{S_j}(\theta)(t_h - s_h)$. □

This is straightforward if $A_1$'s connected component is a singleton: $A_1$ is then mutually independent of all other processes and Lemmas 1 and 2 apply. If $A_1$ is not mutually independent of all other processes, the end-to-end server and $A_1$ are not independent. While a first solution is to apply again Hölder inequality, we propose to compute the bound directly. Instead of computing the end-to-end server first, we directly start with the delay expression from Theorem 6. The proofs use the same arguments as before, so we defer them to Appendix C.

**Theorem 11.** *With the same notations as above, for all $(p_i), (q_j)$ such that for all $k \leq K$, $\sum_{A_i \in G_k} \frac{1}{p_i} + \sum_{S_j \in G_k} \frac{1}{q_j} = 1$ and $\theta \geq 0$ such that for all $j \in \{1, \ldots, n\}$, $\sum_{i \in j} \rho_{A_i}(p_i \theta) < \rho_{S_j}(q_j \theta)$,*

*1. the departure process of flow 1 is bounded by the arrival bounding generating function*

$$F_{D_{e2e}}(\theta, (p_i), (q_j), z) = \frac{e^{\theta \sigma_{A_1}(p_1 \theta)} F_{S_{e2e}}(\theta, (p_i)_{i \neq 1}, (q_j), e^{\theta \rho_{A_1}(p_1 \theta)})}{1 - e^{\theta \rho_{A_1}(p_1 \theta)} z};$$

*2. the delay of flow $A_1$ is bounded by the generating function*

$$F_d(\theta, (p_i), (q_j), z) = \frac{e^{\theta(\sigma_{A_1}(p_1 \theta) + \rho_{A_1}(p_1 \theta))} F_{S_{e2e}}(\theta, (p_i)_{i \neq 1}, (q_j), e^{\theta \rho_{A_1}(p_1 \theta)})}{1 - e^{-\theta \rho_{A_1}(p_1 \theta)} z}$$

$$- \frac{e^{\theta \sigma_{A_1}(p_1 \theta)} z F_{S_{e2e}}(\theta, (p_i)_{i \neq 1}, (q_j), z)}{1 - e^{-\theta \rho_{A_1}(p_1 \theta)} z}.$$
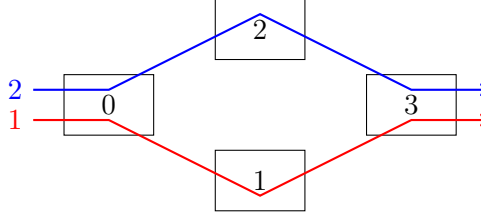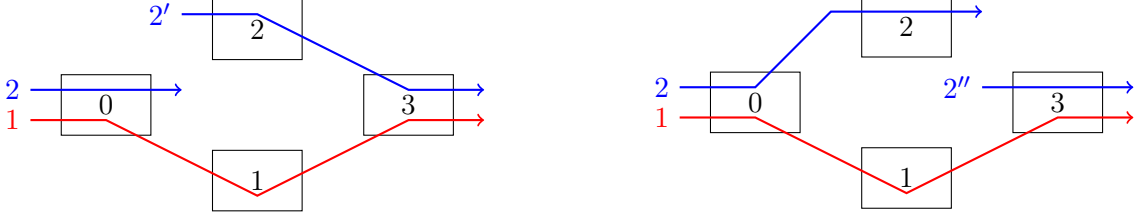
Figure 11: Diamond network.



Figure 12: Two possibilities for cutting flows to transform the diamond network into a tree-reducible network.

## 5.2 Case study: diamond network

Here, we consider the diamond network from Figure 11, where two originally independent flows separate and then rejoin. If we assume some kind of resource-sharing scheduling policy at server 0, then their outputs are dependent when interfering at server 3. This network is not tree-reducible.

### 5.2.1 Transformation into a tree-reducible network

A solution to transform this feedforward network into a tree-reducible network is the *cutting* of flows. If we are interested in the performance of flow 1, we can cut flow 2, either between servers 0 and 2 or between servers 2 and 3, as shown in Figure 12. Clearly, we obtain a tree-reducible network and can now apply Theorems 10 and 11 from the previous subsection.

The analysis of the two obtained networks follows similar lines, so let us focus on the transformation when flow 2 is cut between servers 2 and 3 (on the right of Figure 12). A flow $2''$ is then created, and it stochastically depends on flow 1, flow 2, server 0 and server 2. Using the independence of these flows and servers and results from Section 3.3, we can compute a bounding generating function for the arrival process of flow $2''$:

$$F_{A_{2''}}(\theta, z) = F_{D_2}(\theta, z) = \frac{e^{\theta(\sigma_{A_1}(\theta)+\sigma_{A_2}(\theta)+\sigma_{S_0}(\theta)+\sigma_{S_2}(\theta))}}{\left(1 - e^{-\theta\left(\rho_{S_0}(\theta)-\rho_{A_1}(\theta)-\rho_{A_2}(\theta)\right)}\right)\left(1 - e^{-\theta\left(\rho_{S_2}(\theta)-\rho_{A_2}(\theta)\right)}\right)} \frac{1}{1 - e^{\theta\rho_{A_2}(\theta)}z}.$$

Now, when analysing flow 1, we do not need to consider server 2 (a simple case of tree reduction). The dependency graph of the processes has one non-trivial connected component $\{A_1, A_2, A_{2''}, S_0\}$, and the other components $\{S_1\}$ and $\{S_3\}$ are singletons. In the last step, in order to compute the performance bounds for flow 1, we first obtain the bounding generating function of the end-to-end service with Theorem 10:

$$F_{S_{e2e}}(\theta, z) = \frac{e^{\theta(\sigma_{A_1}(p_{2''}\theta)+\sigma_{A_2}(p_2\theta)+\sigma_{A_2}(p_{2''}\theta)+\sigma_{S_0}(q_0\theta)+\sigma_{S_0}(p_{2''}\theta)+\sigma_{S_1}(\theta)+\sigma_{S_2}(p_{2''}\theta)+\sigma_{S_3}(\theta))}}{\left(1 - e^{-p_{2''}\theta(\rho_{S_0}(p_{2''}\theta)-\rho_{A_1}(p_{2''}\theta)-\rho_{A_2}(p_{2''}\theta))}\right)^{\frac{1}{p_{2''}}}\left(1 - e^{-p_{2''}\theta(\rho_{S_2}(p_{2''}\theta)-\rho_{A_2}(p_{2''}\theta))}\right)^{\frac{1}{p_{2''}}}}$$
$$\cdot \frac{1}{\left(1 - e^{-\theta(\rho_{S_0}(q_0\theta)-\rho_{A_2}(p_2\theta))}z\right)\left(1 - e^{-\theta\rho_{S_1}(\theta)}z\right)\left(1 - e^{-\theta(\rho_{S_3}(\theta)-\rho_{A_2}(p_{2''}\theta))}z\right)}.$$

Theorem 11 can now be applied to compute, for instance, the delay violation probability with the relation $1/p_1 + 1/p_2 + 1/p_{2''} + 1/q_0 = 1$.
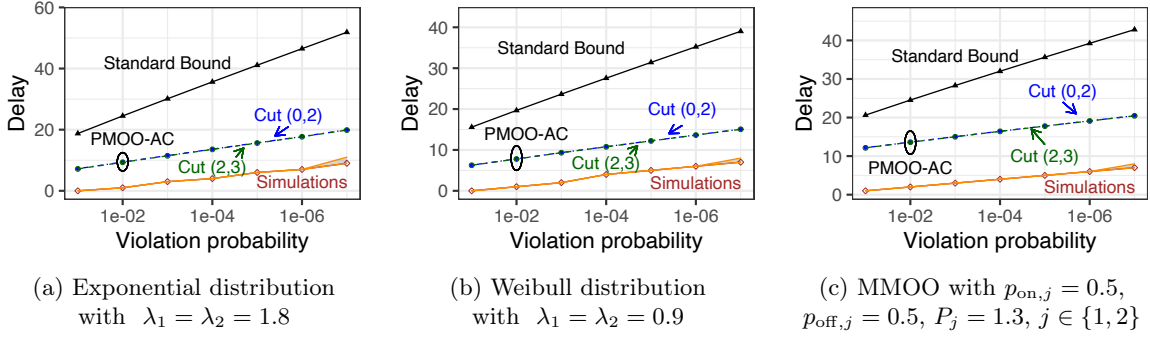
(a) Exponential distribution
with $\lambda_1 = \lambda_2 = 1.8$

(b) Weibull distribution
with $\lambda_1 = \lambda_2 = 0.9$

(c) MMOO with $p_{\text{on},j} = 0.5$,
$p_{\text{off},j} = 0.5$, $P_j = 1.3$, $j \in \{1,2\}$

Figure 13: Delay bounds and simulations for the diamond network with server rates $C_i = 2.0$ for $i = 1, \ldots, 4$.

### 5.2.2 Numerical evaluation

Here, we provide some numerical results for delay bounds for the diamond network from Figure 11. We compare a sequential analysis, representing the state of the art, with the aforementioned cutting technique to enable the PMOO analysis; simulation results are provided as well. The results are displayed in Figure 13.

We can observe that both cuts followed by PMOO-AC calculations lead to significantly tighter delay bounds than the standard approach. To be precise, for a violation probability of $10^{-6}$ the cut between servers 2 and 3 (called Cut (2,3) in Figure 13) as well as Cut (0,2) lead to a delay bound of 14, while the standard bound results in 27. Moreover, the cutting-based PMOO-AC is able to capture almost the same scaling as the simulations, which is in clear contrast to the standard bound. It is striking that the gap to the simulation does not increase significantly compared to the independent case, despite the application of Hölder's inequality. This indicates that the inaccuracy induced by Hölder's inequality cannot only be assessed by the number of its applications, but requires a detailed look at the dependency structure.

## 5.3 Discussion

In this section, we have compared state-of-the-art bounds with our new results for a canonical network with rejoining flows; here are only two options to cut the network, leading to almost the same bounds. While the cutting technique can generally be applied in larger networks, it may no longer be an option to simply try all (combinations of) cuts as these generally grow exponentially in the size of the network. A clever search for "good" cuts is conceivable – similar to advanced network analysis techniques in deterministic network calculus, for instance using machine learning techniques as in [23] – but left for future work.

In fact, other strategies to transform a feedforward into a tree-reducible network exist and may lead to more accurate or computationally efficient bounds. For example, a so-called *unfolding*, depicted in Figure 14 for the diamond example. Server 0 is duplicated in servers 0 and 0', flow 1
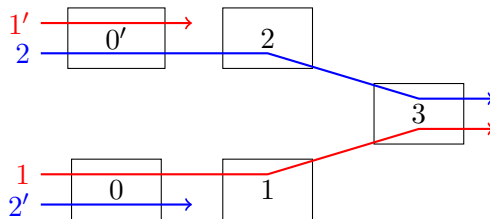


Figure 14: Unfolding of the diamond network of Figure 11.

(resp. flow 2) is duplicated in flows 1 and 1' (resp. flows 2 and 2'). The corresponding arrival and service are the same, and are therefore dependent. Under the assumption of independence of flows and servers, we obtain a dependence graph with non-trivial connected components $G_0 = \{S_0, S_{0'}\}$, $G_1 = \{A_1, A_{1'}\}$, $G_2 = \{A_2, A_{2'}\}$. The connected components are singletons.

To compute the performance bounds for flow 1, we can apply Theorem 11 using the following bounding generating function of the end-to-end server:

$$F_{S_{\text{e2e}}}(\theta, z) = \frac{e^{\theta(\sigma_{A_1}(p_{1'}\theta) + \sigma_{A_2}(p_2\theta) + \sigma_{A_2}(p_{2'}\theta) + \sigma_{S_0}(q_0\theta) + \sigma_{S_0}(q_{0'}\theta) + \sigma_{S_1}(\theta) + \sigma_{S_2}(\theta) + \sigma_{S_3}(\theta))}}{(1 - e^{-\theta(\rho_{S_0}(q_{0'}\theta) - \rho_{A_1}(p_{1'}\theta) - \rho_{A_2}(p_2\theta))})(1 - e^{-\theta(\rho_{S_2}(\theta) - \rho_{A_2}(p_2\theta))})}$$
$$\cdot \frac{1}{(1 - e^{-\theta(\rho_{S_0}(q_0\theta) - \rho_{A_2}(p_{2'}\theta))}z)(1 - e^{-\theta\rho_{S_1}(\theta)}z)(1 - e^{-\theta(\rho_{S_3}(\theta) - \rho_{A_2}(p_2\theta))}z)},$$

with the relations $1/q_0 + 1/q_{0'} = 1$, $1/p_1 + 1/p_{1'} = 1$ and $1/p_2 + 1/p_{2'} = 1$.

Even in the small diamond network, it is not simple to carry out an analytical comparison between cutting and unfolding. On the one hand, $(1 - e^{-pu})^{\frac{1}{p}} \geq 1 - e^{-u}$ for $u > 0$ and $p > 1$, so cutting has an advantage. On the other hand, unfolding provides more freedom in the Hölder parameters: if one sets $p_{2''}^C = q_{0'}^U = p_{1'}^U = p_2^U$, with the superscripts differentiating between parameters from the unfolding $(U)$ and cutting $(C)$ methods, one obtains $q_0^U < q_0^C$, $p_1^U < q_1^C$, and $p_{2'}^U < p_2^C$, and unfolding seems advantageous, at least from the point of view of the stability condition.

In our numerical case study for the diamond network, cutting actually outperformed unfolding. Yet, we speculate that the unfolding method could lead to better bounds in larger networks. To that end, we remark that in deterministic network calculus unfolding in some contexts leads to better performance bounds than cutting [5].

In conclusion, it is clear that, while we have "opened the door" for an accurate and efficient SNC analysis of general feedforward networks, there is now a "large room" to explore for future work.

# 6 Conclusion

We have presented a new network analysis method that unleashes the power of the pay multiplexing only once (PMOO) principle in the stochastic network calculus. Based on this method, we applied techniques from analytic combinatorics to keep bounds accurate even in rather complex scenarios. Equipped with this, it is now possible to calculate rigorous probabilistic performance bounds for tree-reducible networks without incurring any method-pertinent stochastic dependencies. In numerical evaluations, we observed that we are largely successful in not widening the known simulation-calculation gap further, and, at least, closely capture the scaling of the performance bounds. We have also made a promising step towards a stochastic network calculus analysis of general feed-forward networks laying the foundation to reuse the PMOO as much as possible.

While our method can benefit from improvements based on a preliminary network transformation (*e.g.* flow prolongation [29]) at no cost, we believe it can also exploit other recent techniques such as the *h*-mitigators, which were successfully applied to sink-tree networks in [30]. As discussed at the end of Section 5, for future work it is very promising to invest more effort in good strategies for the transformation of large general feedforward networks into tree-reducible ones. More disruptively, the PMOO method could also be a first step to enable martingale techniques (as in [12, 31]) in the end-to-end analysis, in order to completely close the simulation-calculation gap.

# References

[1] François Baccelli, Guy Cohen, Geert Jan Olsder, and Jean-Pierre Quadrat. *Synchronization and linearity: an algebra for discrete event systems.* John Wiley & Sons Ltd, 1992.

[2] Michael A Beck. *Advances in Theory and Applicability of Stochastic Network Calculus.* PhD thesis, TU Kaiserslautern, 2016.

[3] Michael A. Beck and Jens Schmitt. On the Calculation of Sample-Path Backlog Bounds in Queueing Systems over Finite Time Horizons. Technical report, University of Kaiserslautern, Germany, may 2012.

[4] Michael A. Beck and Jens Schmitt. The disco stochastic network calculator version 1.0 - when waiting comes to an end. In *Proc. 7th International Conference on Performance Evaluation Methodologies and Tools (VALUETOOLS'13)*, 2013.

[5] Anne Bouillard. Trade-off between accuracy and tractability of network calculus in FIFO networks. *Performance Evaluation*, 153:102250, 2022.

[6] Anne Bouillard, Marc Boyer, and Euriell Le Corronc. *Deterministic Network Calculus: From Theory to Practical Implementation.* John Wiley & Sons, 2018.

[7] Anne Bouillard, Céline Comte, Élie de Panafieu, and Fabien Mathieu. Of kernels and queues: when network calculus meets analytic combinatorics. In *2018 30th International Teletraffic Congress (ITC 30)*, pages 49–54, 2018.

[8] Anne Bouillard, Bruno Gaujal, Sébastien Lagrange, and Eric Thierry. Optimal routing for end-to-end guarantees using network calculus. *Performance Evaluation*, 65(11-12):883–906, 2008.

[9] Hugh Boyes, Bil Hallaq, Joe Cunningham, and Tim Watson. The industrial internet of things (iiot): An analysis framework. *Computers in Industry*, 101:1–12, 2018.

[10] Cheng-Shang Chang. *Performance guarantees in communication networks.* Springer-Verlag, London, 2000.

[11] Florin Ciucu, Almut Burchard, and Jörg Liebeherr. Scaling properties of statistical end-to-end bounds in the network calculus. *IEEE/ACM Transactions on Networking (ToN)*, 14(6):2300–2312, 2006.

[12] Florin Ciucu, Felix Poloczek, and Jens Schmitt. Sharp per-flow delay bounds for bursty arrivals: The case of FIFO, SP, and EDF scheduling. In *Proc. IEEE International Conference on Computer Communications (INFOCOM'14)*, 2014.

[13] Florin Ciucu and Jens Schmitt. Perspectives on network calculus – no free lunch, but still good value. In *Proc. ACM Conf. on Applications, Technologies, Architectures, and Protocols for Computer Commun. (SIGCOMM'12)*, pages 311–322, 2012.

[14] Rene L Cruz. A calculus for network delay, part I: Network elements in isolation. *IEEE Transactions on information theory*, 37(1):114–131, 1991.

[15] Rene L Cruz. A calculus for network delay, part II: Network analysis. *IEEE Transactions on information theory*, 37(1):132–141, 1991.

[16] Rene L Cruz. Quality of service management in integrated services networks. In *Proc. Semi-Annual Research Review, CWC*, pages 4–5, 1996.

[17] Feng-Cheng Chang. Recursive formulas for the partial fraction expansion of a rational function with multiple poles. *Proceedings of the IEEE*, 61(8):1139–1140, 1973.

[18] Gerhard Paul Fettweis. The tactile internet: Applications and challenges. *IEEE Vehicular Technology Magazine*, 9(1):64–70, 2014.

[19] Markus Fidler. Extending the network calculus pay bursts only once principle to aggregate scheduling. In *International Workshop on Quality of Service in Multiservice IP Networks*, pages 19–34. Springer, 2003.

[20] Markus Fidler. An end-to-end probabilistic network calculus with moment generating functions. In *Proc. IEEE IWQoS'06*, pages 261–270, Jun. 2006.

[21] Philippe Flajolet and Fabrice Guillemin. The formal theory of birth-and-death processes, lattice path combinatorics and continued fractions. *Advances in Applied Probability*, 32(3):750–778, 2000.

[22] Philippe Flajolet and Robert Sedgewick. *Analytic combinatorics*. Cambridge University Press, 2009.

[23] Fabien Geyer and Steffen Bondorf. DeepTMA: Predicting effective contention models for network calculus using graph neural networks. In *Proc. IEEE International Conference on Computer Communications (INFOCOM'19)*, pages 1009–1017, 2019.

[24] Fabrice Guillemin and Didier Pinchon. Analysis of generalized processor-sharing systems with two classes of customers and exponential services. *Journal of Applied Probability*, 41, 09 2004.

[25] Gerald J Hahn and William Q Meeker. *Statistical intervals: a guide for practitioners*, volume 92. John Wiley & Sons, 2011.

[26] Yuming Jiang and Yong Liu. *Stochastic network calculus*, volume 1. Springer, 2008.

[27] Michael Mitzenmacher and Eli Upfal. *Probability and Computing: Randomized Algorithms and Probabilistic Analysis*. Cambridge University Press, 2005.

[28] Paul Nikolaus and Jens Schmitt. On Per-Flow Delay Bounds in Tandem Queues under (In)Dependent Arrivals. In *Proc. 16th IFIP Networking Conference (NETWORKING'17)*, pages 1–9, 2017.

[29] Paul Nikolaus and Jens Schmitt. Improving delay bounds in the stochastic network calculus by using less stochastic inequalities. In *Proc. 13th EAI International Conference on Performance Evaluation Methodologies and Tools (VALUETOOLS'20)*, page 96–103, 2020.

[30] Paul Nikolaus, Jens B. Schmitt, and Malte Schütze. h-mitigators: Improving your stochastic network calculus output bounds. *Computer Communications*, 144:188–197, 2019.

[31] Felix Poloczek and Florin Ciucu. Scheduling analysis with martingales. *Performance Evaluation*, 79:56 – 72, 2014. Special Issue: Performance 2014.

[32] Amr Rizk and Markus Fidler. Leveraging statistical multiplexing gains in single-and multi-hop networks. In *Proc. IEEE International Workshop on Quality of Service (IWQoS'17)*, pages 1–9, 2011.

[33] Jens Schmitt and Frank A. Zdarsky. The DISCO Network Calculator - A Toolbox for Worst Case Analysis. In *Proceedings of the First International Conference on Performance Evaluation Methodologies and Tools (VALUETOOLS'06)*, 2006.

[34] Jens Schmitt, Frank A Zdarsky, and Ivan Martinovic. Improving performance bounds in feed-forward networks by paying multiplexing only once. In *Proc. GI/ITG Conference on Measurement, Modeling, and Evaluation of Computer and Communication Systems (MMB'08)*, pages 1–15, 2008.

[35] The Sage Developers. Sagemath, the Sage Mathematics Software System (Version 9.2), 2020. https://www.sagemath.org.

[36] Timothy Zhu, Daniel S Berger, and Mor Harchol-Balter. SNC-meister: Admitting more tenants with tail latency SLOs. In *Proc. ACM Symposium on Cloud Computing (SoCC'16)*, pages 374–387, sep 2016.

# A  Additional Proofs of Section 2

## A.1  Proof of Equation (3)

$$G(z) = \sum_{n=0}^{\infty}(\sum_{m=n}^{\infty} f_m)z^n = \sum_{n=0}^{\infty}(\sum_{m=0}^{\infty} f_m)z^n - \sum_{n=0}^{\infty}(\sum_{m=0}^{n-1} f_m)z^n$$

$$= F(1)\sum_{n=0}^{\infty} z^n - \sum_{n=1}^{\infty}(\sum_{m=0}^{n-1} f_m)z^n = \frac{F(1)}{1-z} - z\sum_{n=0}^{\infty}(\sum_{m=0}^{n} f_m)z^n$$

$$= \frac{F(1)}{1-z} - z\sum_{n=0}^{\infty}(\sum_{m=0}^{n} f_m z^m z^{n-m}) = \frac{F(1)}{1-z} - z\sum_{m=0}^{\infty}(f_m z^m \sum_{n=m}^{\infty} z^{n-m})$$

$$= \frac{F(1)}{1-z} - z\frac{F(z)}{1-z} = \frac{F(1) - zF(z)}{1-z}.$$

## A.2  Proof of Lemma 2

As $F_A(\theta, z)$ is a geometric series, we have $[z^{n+m}]F_A(\theta, z) = e^{\theta \rho_A(\theta)m}[z^n]F_A(\theta, z)$. Let us denote $d(t)$ the delay at time $t$. We have for all $T > 0$,

$$\mathbf{P}(d(t) \geq T) = \mathbf{P}(A(0,t) > D(0, t + T - 1))$$

$$\leq \mathbf{P}(\exists s \leq t, \ A(s,t) > S(s, t + T - 1))$$

$$\leq \sum_{0 \leq s \leq t} \mathbf{P}(A(s,t) > S(s, t + T - 1))$$

$$\leq \sum_{0 \leq s \leq t-1} \mathbf{E}[e^{\theta(A(s,t) - S(s,t+T-1))}] \qquad (A(t,t) = 0 \text{ and } S(t, t+T-1) \geq 0)$$

$$= \sum_{0 \leq s \leq t-1} \mathbf{E}[e^{\theta A(s,t)}]\mathbf{E}[e^{-\theta S(s,t+T-1)}] \qquad (\text{independence of } A \text{ and } S)$$

$$\leq \sum_{0 \leq s \leq t-1} [z^{t-s}]F_A(\theta, z) \cdot [z^{t-s+T-1}]F_S(\theta, z)$$

$$\leq \sum_{u > 0}[z^u]F_A(\theta, z) \cdot [z^{u+T-1}]F_S(\theta, z) \qquad (u \leftarrow t - s)$$

$$= e^{-\theta \rho_A(\theta)(T-1)} \sum_{u > 0}[z^{u+T-1}]F_A(\theta, z) \cdot [z^{u+T-1}]F_S(\theta, z)$$

$$= e^{-\theta \rho_A(\theta)(T-1)} \sum_{u \geq T}[z^u]F_A(\theta, z) \cdot [z^u]F_S(\theta, z). \tag{11}$$

We recognize the sum of the last terms of a Hadamard product in the right-hand term. Let $h_u = [z^u]F_A(\theta, z) \cdot [z^u]F_S(\theta, z)$ and $h'_u = \sum_{u \geq T} h_u$. Let $H(z)$ and $H'(z)$ be their corresponding generating functions. In particular, we have $H(z) = e^{\theta \sigma_A(\theta)}F_S(\theta, e^{\theta \rho_A(\theta)}z)$. Now,

$$\mathbf{P}(d(t) \geq T) \leq e^{-\theta \rho_A(\theta)(T-1)}[z^T]H'(z),$$

and we recognize again a Hadamard product of a geometric series and $H'$, and use Equation (3) to obtain

$$F_d(z) = e^{\theta \rho_A(\theta)}H'(e^{-\theta \rho_A(\theta)}z) = e^{\theta \rho_A(\theta)}\frac{H(1) - e^{-\theta \rho_A(\theta)}zH(e^{-\theta \rho_A(\theta)}z)}{1 - e^{-\theta \rho_A(\theta)}z}$$

$$= e^{\theta(\sigma_A(\theta) + \rho_A(\theta))}\frac{F_S(\theta, e^{\theta \rho_A(\theta)}) - e^{-\theta \rho_A(\theta)}zF_S(\theta, z)}{1 - e^{-\theta \rho_A(\theta)}z}.$$

# B    Proofs and Computations of Section 3

## B.1    Proof of Corollary 2

It suffices to prove that for all $t \in \mathbb{N}$, $[z^t]F_{S_{e2e}}(\theta, z) \leq [z^t]G(\theta, z)$. For all $t \in \mathbb{N}$,

$$[z^t]F_{S_{e2e}}(\theta, z) = e^{\theta \sigma_{S_{e2e}}(\theta)} \sum_{u_1 + \ldots + u_n = t} \prod_{j=1}^{n} e^{-\theta \rho'_j(\theta)u_j}$$

$$= e^{\theta \sigma_{S_{e2e}}(\theta)}e^{-\theta \rho'_1(\theta)t} \sum_{u_1 + \ldots + u_n = t} \prod_{j=1}^{n} e^{-\theta(\rho'_j(\theta) - \rho'_1(\theta))u_j}$$

$$= e^{\theta \sigma_{S_{e2e}}(\theta)}e^{-\theta \rho'_1(\theta)t} \sum_{s_1 + s_2 = t} \left( \sum_{u_1 + \ldots + u_k = s_1} 1 \right)$$

$$\cdot \left( \sum_{u_{k+1} + \ldots + u_n = s_2} \prod_{j=k+1}^{n} e^{-\theta(\rho'_j(\theta) - \rho'_1(\theta))u_j} \right)$$

$$= e^{\theta \sigma_{S_{e2e}}(\theta)}e^{-\theta \rho'_1(\theta)t} \sum_{s_1 + s_2 = t} \binom{s_1 + k - 1}{k - 1}$$

$$\cdot \left( \sum_{u_{k+1} + \ldots + u_n = s_2} \prod_{j=k+1}^{n} e^{-\theta(\rho'_j(\theta) - \rho'_1(\theta))u_j} \right)$$

$$\leq e^{\theta \sigma_{S_{e2e}}(\theta)}e^{-\theta \rho'_1(\theta)t} \binom{t + k - 1}{k - 1} \left( \sum_{u_{k+1} + \ldots + u_n \leq t} \prod_{j=k+1}^{n} e^{-\theta(\rho'_j(\theta) - \rho'_1(\theta))u_j} \right)$$

$$\leq e^{\theta \sigma_{S_{e2e}}(\theta)}e^{-\theta \rho'_1(\theta)t} \binom{t + k - 1}{k - 1} \prod_{j=k+1}^{n} \frac{1}{1 - e^{-\theta(\rho'_j(\theta) - \rho'_1(\theta))}} = [z^t]G(\theta, z).$$

## B.2    Proof of Equation (8): $F_{S_{e2e}}(\theta, z)$ has singularities with multiplicities 1 only.

We first use partial fraction decomposition [17]:

$$\prod_{j=1}^{n} \frac{1}{1 - r_j z} = \sum_{j=1}^{n} \prod_{k \neq j} \left( \frac{1}{1 - r_j^{-1}r_k} \right) \frac{1}{1 - r_j z}.$$

and

$$\frac{r_0^{-1}}{1-r_0 z}\prod_{j=1}^{n}\frac{1}{1-r_0^{-1}r_j}-\frac{z}{1-r_0 z}\prod_{j=1}^{n}\frac{1}{1-r_j z}$$

$$=\frac{r_0^{-1}}{1-r_0 z}\prod_{j=1}^{n}\frac{1}{1-r_0^{-1}r_j}-z\Big[\sum_{j=0}^{n}\Big(\prod_{k\geq 0,k\neq j}\frac{1}{1-r_j^{-1}r_k}\Big)\frac{1}{1-r_j z}\Big]$$

$$=r_0^{-1}\prod_{j=1}^{n}\frac{1}{1-r_0^{-1}r_j}-\sum_{j=1}^{n}\frac{z}{1-r_0 r_j^{-1}}\Big(\prod_{k\geq 1,k\neq j}\frac{1}{1-r_j^{-1}r_k}\Big)\frac{1}{1-r_j z}.$$

In the bounding generating function of the delay, by identifying $r_0$ as $e^{-\theta\rho_{A_1}(\theta)}$ and for all $j>0$, $r_j$ as $e^{-\theta\rho'_j(\theta)}$, we obtain (for ease of presentation, we drop the dependence of the $\sigma$'s and $\rho$'s in $\theta$)

$$F_d(\theta,z)=e^{\theta\sigma_{A_1}}\frac{e^{\theta\rho_{A_1}}F_{S_{e2e}}(\theta,e^{\theta\rho_{A_1}})-zF_{S_{e2e}}(\theta,z)}{1-e^{-\theta\rho_{A_1}}z}$$

$$=\frac{e^{\theta(\sigma_{A_1}+\sigma_{S_{e2e}}+\rho_{A_1})}}{\prod_{j=1}^{n}\big(1-e^{\theta(\rho_{A_1}-\rho'_j)}\big)}\frac{1}{1-e^{-\theta\rho_{A_1}}z}-\frac{e^{\theta(\sigma_{A_1}+\sigma_{S_{e2e}})}z}{1-e^{-\theta\rho_{A_1}}z}\prod_{j=1}^{n}\frac{1}{1-e^{-\theta\rho'_j}z}$$

$$=\frac{e^{\theta(\sigma_{A_1}+\sigma_{S_{e2e}}+\rho_{A_1})}}{\prod_{j=1}^{n}\big(1-e^{\theta(\rho_{A_1}-\rho'_j)}\big)}+\sum_{j=1}^{n}\frac{e^{\theta(\sigma_{A_1}+\sigma_{S_{e2e}})}z}{e^{\theta(\rho'_j-\rho_{A_1})}-1}\Big(\prod_{k\neq j}\frac{1}{1-e^{\theta(\rho'_j-\rho'_k)}}\Big)\frac{1}{1-e^{-\theta\rho'_j}z}.$$

This is of the form $f(z)=a+\sum_{j=1}^{n}\frac{b_j z}{1-r_j z}$, and the coefficients of $f$ are $[z^0]f(z)=f(0)=a$ and for all $T>0$,

$$[z^T]f(z)=\sum_{j=1}^{n}[z^T]\frac{b_j z}{1-r_j z}=\sum_{j=1}^{n}[z^{T-1}]\frac{b_j}{1-r_j z}=\sum_{j=1}^{n}b_j r_j^{T-1}. \tag{12}$$

As a consequence, for all $T>0$,

$$[z^T]F_d(\theta,z)=\sum_{j=1}^{n}\frac{e^{\theta(\sigma_{A_1}+\sigma_{S_{e2e}})}}{e^{\theta(\rho'_j-\rho_{A_1})}-1}\Big(\prod_{k\neq j}\frac{1}{1-e^{\theta(\rho'_j-\rho'_k)}}\Big)e^{-\theta\rho'_j(T-1)}$$

$$=\sum_{j=1}^{n}\frac{e^{\theta(\sigma_{A_1}+\sigma_{S_{e2e}}+\rho_{A_1})}}{1-e^{\theta(\rho_{A_1}-\rho'_j)}}\Big(\prod_{k\neq j}\frac{1}{1-e^{\theta(\rho'_j-\rho'_k)}}\Big)e^{-\theta\rho'_j T}.$$

## B.3 Proof of Equation (9): $F_{S_{e2e}}(\theta,z)$ has exactly one singularity

In this subsection, we assume that all $\rho'_j(\theta)$, $j\in\{1,\ldots,n\}$ are equal.

We also use partial fractional decomposition:

$$\frac{1}{(1-r_0 z)(1-r_1 z)^n}=\frac{1}{(1-r_0 z)(1-r_0^{-1}r_1)^n}-\frac{r_0^{-1}r_1}{(1-r_0^{-1}r_1)^{n+1}}\Big[\sum_{i=1}^{n}\Big(\frac{1-r_0^{-1}r_1}{1-r_1 z}\Big)^i\Big],$$

and

$$\frac{r_0^{-1}}{(1-r_0 z)(1-r_0^{-1}r_1)^n}-\frac{z}{(1-r_0 z)(1-r_1 z)^n}=$$

$$\frac{r_0^{-1}}{(1-r_0^{-1}r_1)^n}+\frac{r_0^{-1}r_1 z}{(1-r_0^{-1}r_1)^{n+1}}\Big[\sum_{i=1}^{n}\Big(\frac{1-r_0^{-1}r_1}{1-r_1 z}\Big)^i\Big],$$

The bounding generating function of the delay can be obtained by replacing $r_0$ by $e^{-\theta\rho_{A_1}}$ and $r_1$ by $e^{-\theta\rho_1'}$:

$$F_d(\theta, z) = e^{\theta(\sigma_{A_1} + \sigma_{S_{e2e}})} \left( \frac{e^{\theta\rho_{A_1}}}{(1 - e^{-\theta(\rho_1' - \rho_{A_1})})^n} \right.$$

$$\left. + e^{\theta(\rho_{A_1} - \rho_1')} z \left[ \sum_{i=1}^{n} \frac{1}{(1 - e^{-\theta(\rho_1' - \rho_{A_1})})^{n-i+1}} \frac{1}{(1 - e^{-\theta\rho_1'} z)^i} \right] \right).$$

If $f(z) = \frac{1}{(1-rz)^m}$, then its coefficients are $[z^T]f(z) = \binom{T+m-1}{T} r^T$. Then, following the same computations as in (12), one can deduce that for all $T > 0$,

$$[z^T]F_d(\theta, z) = e^{\theta(\sigma_{A_1} + \sigma_{S_{e2e}} + \rho_{A_1} - \rho_1')} \left[ \sum_{i=1}^{n} \frac{1}{(1 - e^{-\theta(\rho_1' - \rho_{A_1})})^{n-i+1}} \binom{T+i-2}{T-1} e^{-\theta\rho_1'(T-1)} \right]$$

$$= e^{\theta(\sigma_{A_1} + \sigma_{S_{e2e}} + \rho_{A_1})} \left[ \sum_{i=1}^{n} \frac{1}{(1 - e^{-\theta(\rho_1' - \rho_{A_1})})^{n-i+1}} \binom{T+i-2}{T-1} e^{-\theta\rho_1' T} \right].$$

# C   Proof of Theorem 11

## C.1   Bounding generating function of the departure process

We have

$$A_1 \oslash S_{e2e}(s,t) = \sup_{u \leq s} A_1(u,t) - S_{e2e}(u,s)$$

$$= \sup_{u \leq s} \sup_{\substack{t_j \leq t_{j\bullet} \\ t_{\pi_1(1)} = u \\ t_{\pi_1(\ell_1)\bullet} = s}} A_1(u,t) - \sum_{j=1}^{n} S_j(t_j, t_{j\bullet}) + \sum_{i=2}^{m} A_i(t_{\pi_i(1)}, t_{\pi_i(\ell_i)\bullet}).$$

When computing the MGF $\mathbf{E}[e^{\theta(A_1 \oslash S_{e2e}(s,t))}]$, one can take advantage of the partial independence of $A_1$ with the other processes.

For all $(p_i), (q_j)$ such that for all $k \in \{1, \ldots, K\}$, $\sum_{A_i \in G_k} \frac{1}{p_i} + \sum_{S_j \in G_k} \frac{1}{q_j} = 1$, we have, using Lemma 4 in the second line,

$$\mathbf{E}[e^{\theta(A_1 \oslash S_{e2e}(s,t))}] \leq \sum_{u \leq s} \sum_{\substack{t_j \leq t_{j\bullet} \\ t_{\pi_1(1)} = u \\ t_{\pi_1(\ell_1)\bullet} = s}} \mathbf{E}[e^{\theta(A_1(u,t) - \sum_{j=1}^{n} S_j(t_j, t_{j\bullet}) + \sum_{i=2}^{m} A_i(t_{\pi_i(1)}, t_{\pi_i(\ell_i)\bullet}))}]$$

$$\leq \sum_{u \leq s} \sum_{\substack{t_j \leq t_{j\bullet} \\ t_{\pi_1(1)} = u \\ t_{\pi_1(\ell_1)\bullet} = s}} e^{\theta(\sigma_{A_1}(p_1\theta) + \rho_{A_1}(p_1\theta)(t-u))} \left( \prod_{j=1}^{n} e^{-\theta(\sigma_{S_j}(q_j\theta) - \rho_{S_j}(q_j\theta)(t_{j\bullet} - t_j))} \right.$$

$$\left. \prod_{i=2}^{m} e^{\theta(\sigma_{A_i}(p_i\theta) + \rho_{A_i}(p_i\theta)(t_{\pi_i(\ell_i)\bullet} - t_{\pi_i(1)}))} \right)$$

$$= \sum_{u \leq s} e^{\theta(\sigma_{A_1}(p_1\theta) + \rho_{A_1}(p_1\theta)(t-u))} \cdot \sum_{\substack{t_j \leq t_{j\bullet} \\ t_{\pi_1(1)} = u \\ t_{\pi_1(\ell_1)\bullet} = s}} \left( \prod_{j=1}^{n} e^{-\theta(\sigma_{S_j}(q_j\theta) - \rho_{S_j}(q_j\theta)(t_{j\bullet} - t_j))} \right.$$

$$\left. \prod_{i=2}^{m} e^{\theta(\sigma_{A_i}(p_i\theta) + \rho_{A_i}(p_i\theta)(t_{\pi_i(\ell_i)\bullet} - t_{\pi_i(1)}))} \right)$$

$$= \sum_{u \leq s} [z^{t-u}] F_{A_1}(\theta, p_1, z)[z^{s-u}] F_{S_{\mathrm{e2e}}}(\theta, (p_i), (q_j), z).$$

Similar calculations to those in Lemma 1 lead to

$$F_{D_{\mathrm{e2e}}}(\theta, (p_i), (q_j), z) = \frac{e^{\theta \sigma_{A_1}(p_1\theta)} F_{S_{\mathrm{e2e}}}(\theta, (p_i)_{i \neq 1}, (q_j), e^{\theta \rho_{A_1}(p_1\theta)})}{1 - e^{\theta \rho_{A_1}(p_1\theta)} z}.$$

## C.2 Bounding generating function of the delay

From the proof of Lemma 2 and then Lemma 4,

$$\mathbf{P}(d(t) \geq T) \leq \sum_{0 \leq s \leq t-1} \mathbf{E}[e^{\theta(A_1(s,t) - S_{\mathrm{e2e}}(s, t+T-1))}]$$

$$\leq \sum_{0 \leq s \leq t-1} \sum_{\substack{t_j \leq t_{j\bullet} \\ t_{\pi_1(1)} = s \\ t_{\pi_1(\ell_1)\bullet} = t+T-1}} \mathbf{E}[e^{\theta(A_1(s,t) - \sum_{j=1}^{n} S_j(t_j, t_{j\bullet}) + \sum_{i=2}^{m} A_i(t_{\pi_i(1)}, t_{\pi_i(\ell_i)\bullet}))}]$$

$$\leq \sum_{s \leq t-1} \sum_{\substack{t_j \leq t_{j\bullet} \\ t_{\pi_1(1)} = s \\ t_{\pi_1(\ell_1)\bullet} = t+T-1}} e^{\theta(\sigma_{A_1}(p_1\theta) + \rho_{A_1}(p_1\theta)(t-s))} \left( \prod_{j=1}^{n} e^{-\theta(\sigma_{S_j}(q_j\theta) - \rho_{S_j}(q_j\theta)(t_{j\bullet} - t_j))} \right.$$

$$\left. \prod_{i=2}^{m} e^{\theta(\sigma_{A_i}(p_i\theta) + \rho_{A_i}(p_i\theta)(t_{\pi_i(\ell_i)\bullet} - t_{\pi_i(1)}))} \right)$$

$$\leq \sum_{s \leq t-1} [z^{t-s}] F_{A_1}(\theta, p_1, z) \cdot [z^{t+T-s-1}] F_{S_{\mathrm{e2e}}}(\theta, (p_i), (q_j), z).$$

Following the same lines as in Lemma 2 leads to

$$F_d(\theta, (p_i), (q_j), z) = \frac{e^{\theta(\sigma_{A_1}(p_1\theta) + \rho_{A_1}(p_1\theta))}(F_{S_{\mathrm{e2e}}}(\theta, (p_i)_{i \neq 1}, (q_j), e^{\theta \rho_{A_1}(p_1\theta)})}{1 - e^{-\theta \rho_{A_1}(p_1\theta)} z} -$$

$$\frac{e^{\theta \sigma_{A_1}(p_1\theta)} z F_{S_{\mathrm{e2e}}}(\theta, (p_i)_{i \neq 1}, (q_j), z))}{1 - e^{-\theta \rho_{A_1}(p_1\theta)} z}.$$