# Tight performance bounds in the worst-case analysis of feed-forward networks *

Anne Bouillard, Eric Thierry

# Tight performance bounds in the worst-case analysis of feed-forward networks*

Anne Bouillard
ENS/INRIA (Paris, France)
anne.bouillard@ens.fr

Éric Thierry
ENS Lyon (France)
eric.thierry@ens-lyon.fr

**Abstract**

Network Calculus theory aims at evaluating worst-case performances in communication networks. It provides methods to analyze models where the traffic and the services are constrained by some minimum and/or maximum envelopes (arrival/service curves). While new applications come forward, a challenging and inescapable issue remains open: achieving *tight* analyzes of networks with aggregate multiplexing.

The theory offers efficient methods to bound maximum end-to-end delays or local backlogs. However as shown in a recent breakthrough paper [28], those bounds can be arbitrarily far from the exact worst-case values, even in seemingly simple feed-forward networks (two flows and two servers), under blind multiplexing (*i.e.* no information about the scheduling policies, except FIFO per flow). For now, only a network with three flows and three servers, as well as a tandem network called sink tree, have been analyzed tightly.

We describe the first algorithm which computes the maximum end-to-end delay for a given flow, as well as the maximum backlog at a server, for *any* feed-forward network under blind multiplexing, with piecewise affine concave arrival curves and piecewise affine convex service curves. Its computational complexity may look expensive (possibly super-exponential), but we show that the problem is intrinsically difficult (NP-hard). Fortunately we show that in some cases, like tandem networks with cross-traffic interfering along intervals of servers, the complexity becomes polynomial. We also compare ourselves to the previous approaches and discuss the problems left open.

**Keywords:** Network Calculus; feed-forward networks; blind multiplexing; linear programming

*This work has been partially funded by the French National Agency for Research (ANR PEGASE-SEGI-009-02)

# 1 Introduction

Network Calculus (NC) is a theory of deterministic queuing systems encountered in communications networks. With methods to compute deterministic bounds on delays, backlogs and other Quality-of-Service (QoS) parameters, it aims at analyzing critical behaviors and usually focuses on worst-case performances, either local performances (*i.e.* maximum buffer size at a node) or end-to-end performances (*i.e.* maximum end-to-end delay). The information about the system is stored in functions, such as *arrival curves* shaping the traffic or *service curves* quantifying the service guaranteed at the network nodes. Relevant applications range from Internet QoS [16] to the analysis of System-on-Chip [10], industrial Ethernets [29], critical embedded networks [9]. At the present time, the theory has developed and yield accomplished results which are mainly recorded in two reference books [12, 19]. It is an alternative to other approaches for worst-case performance analysis like holistic methods [33], trajectory methods [21] or model checking [17]. It is believed that Network Calculus and its extensions have advantages like modularity and scalability that will allow valuable analyzes of complex networks [22].

From the beginning [11, 13, 14], Network Calculus methods have always put an emphasis on the use of $(\min, +)$ or $(\max, +)$ tropical algebras, known for their applications to different discrete event systems [2]. The definitions of arrival curve as well as simple minimum service curve can be easily stated with the $(\min, +)$ convolution. A general scheme consists in combining constraint curves thanks to algebraic operations like $(\min, +)$ convolution or $(\max, +)$ deconvolution. Using a few lemmas, one can either propagate constraints through the network and then retrieve performance bounds from all those computations, or express the network behavior with a set of $(\min, +)$ functional equations which must be solved to get the bounds. In this framework, the analysis of a single flow crossing a sequence of servers is tight. The $(\min, +)$ convolution elegantly captures the *Pay Burst Only Once (PBOO)* phenomenon in tandems of servers (burstiness is amortized all along the servers). However as soon as the network presents some aggregate multiplexing of several flows, providing a tight analysis becomes much more difficult.

The NC models are usually classified according to the topology of the network, the scheduling policies and the type of service guaranteed at each server. For general topologies where the flows may interfere with cyclic dependencies, the complexity of computing worst-case performances is still open. Even the simpler question of deciding *stability*, *i.e.* whether global backlog or end-to-end delays remain bounded, is unset for many policies. Related results can be found in the *Adversarial Queuing* literature where the Permanent Session Model matches Network Calculus models [4]. A well-known necessary condition for stability is an utilization factor $< 1$ at each server. This condition is also sufficient for feed-forward networks [14], or unidirectional rings [30]. But, this condition is not sufficient for FIFO scheduling since there exist unstable networks at arbitrarily low utilization factors [1]. For general FIFO networks, the best sufficient conditions and associated bounds on delays are provided by [24]

but they are usually not tight. More thrilling, for simple feed-forward networks like FIFO tandems, a recent paper [3] improving delays bounds has shown that those bounds were not tight yet.

In this paper, we investigate the complexity of computing exact worst-case performances (end-to-end delays and local backlogs) for *feed-forward networks* under *blind multiplexing*, *i.e.* no information about the policy except FIFO per flow (also called FIFO per micro-flow [25]). This assumption is weaker than FIFO scheduling for the aggregated flows. A first study of tandem networks put forward a new phenomenon called *Pay Multiplexing Once (PMOO)* (competition between flows for the resources is amortized all along the servers) [26]. It presented a new method taking into account PMOO and experiments showed a significant improvement to the end-to-end delay bounds with regard to previous NC approaches. This method could be formulated as a new $(\min, +)$ multi-dimensional convolution [5], thus preserving the NC spirit while being a good candidate for tight analysis of blind multiplexing. However a recent breakthrough paper [28] showed that those bounds could be arbitrarily far from the exact worst-case values, even in seemingly simple feed-forward networks (two flows and two servers). This paper suggested a new approach using linear programming, but for now, only a network with three flows and three servers, as well as sink-tree tandems, could be analyzed tightly.

Our paper describes the first algorithm which computes the maximum end-to-end delay for a given flow, as well as the maximum backlog at a server, for any feed-forward network under blind multiplexing, with piecewise affine concave arrival curves and piecewise affine convex strict service curves. It also provides a critical trajectory of the system, achieving the worst-case value. Its computational complexity may look expensive (possibly super-exponential), but we show that the problem is intrinsically difficult (NP-hard). Fortunately we show that in some cases, like tandem networks *i.e.* the scenarios studied in [28, 26], the complexity becomes polynomial. Beyond the fact that our solution applies to any feed-forward networks, and although we also use linear programming, several features distinguish our approach from [28]: we tackle both worst-case delays and backlogs, we directly compute worst-case performances instead of looking first for an end-to-end service curve, we avoid a decomposition/recomposition scheme for convex/concave curves which may lead to looser bounds and a more expensive complexity.

This is a long version of the conference paper [6]. In this paper, we (i) give a simplified approach and complete proofs of the results (ii) prove our conjecture concerning the NP-hardness of computing the exact worst-case delay.

The paper is organized as follows: after a presentation of the network model and the main NC notions in Section 2, we describe and analyze our algorithm in Section 3 where we also set the NP-hardness of the problem. Section 4 shows how it applies to tandem networks and compares our solution to previous works namely [28, 27], while experiments are discussed in Section 5 to assess the gain w.r.t. to older NC methods. Further interesting extensions and open problems are presented in Section 6.

Our results are relevant to Network Calculus and extensions like Real-Time

Calculus [31] (which uses strict service curves). But we do not address stochastic extensions like [15, 18].

# 2 Model and assumptions

## 2.1 Network Calculus framework

### 2.1.1 NC functions and systems

In Network Calculus, one must distinguish two kinds of objects: the real movements of data and the constraints that these movements satisfy. The real movements of data are modeled by *cumulative functions*: a cumulative function $f(t)$ counts the total amount of data that has achieved some condition up to time $t$ (*e.g.* the total amount of data which has gone through a given place in the network). We consider a *fluid model* where time is continuous, data can be divided into arbitrarily small pieces (time and data measures belong to $\mathbb{R}_+$) [1] and cumulative functions will belong to $\mathcal{F} = \{f : \mathbb{R}_+ \to \mathbb{R}_+ \mid f \text{ non} - \text{decreasing}, \text{left} - \text{continuous}, f(0) = 0\}$ [2]. Constraint functions either shape the traffic (*arrival curves*) or guarantee some service locally or globally (*service curves*). Constraint functions usually allow the $+\infty$ value. In this paper we will usually assume that they belong to $\mathcal{F}$ for commodity, but a careful look will show that all our solutions can be adjusted with no extra cost to deal with some $+\infty$ values if necessary.

Beyond usual operations like the minimum or the addition of functions, Network Calculus makes use of several classical $(\min, +)$ operations [2] such as: let $f, g \in \mathcal{F}$, $\forall t \in \mathbb{R}_+$,

- convolution: $(f * g)(t) = \inf_{0 \le s \le t}(f(s) + g(t-s))$;
- deconvolution: $(f \oslash g)(t) = \sup_{u \ge 0}(f(t+u) - g(u))$.

An *input/output system* is a subset $S$ of $\{(F^{in}, F^{out}) \in \mathcal{F} \times \mathcal{F} \mid F^{in} \ge F^{out}\}$. It models a flow crossing a system where $F^{in}$ (resp. $F^{out}$) is the cumulative function at the entry (resp. exit) of the system and $F^{in} \ge F^{out}$ indicates that the system only transmits data. The system may range from a single server to a large network with cross-traffic. A *trajectory* of the system $S$ is an element $(F^{in}, F^{out})$ of $S$.

Note that it is not required to suppose that the system is deterministic. The system may admit several trajectories $(F^{in}, F^{out})$, with the same input $F^{in}$ associated with different outputs $F^{out}$.

### 2.1.2 NC arrival curves

Given a data flow, let $F \in \mathcal{F}$ be its cumulative function at some point, *i.e.* $F(t)$ is the number of bits that have reached this point until time $t$, with $F(0) = 0$.

[1] we will call *bit* the data unit of measure.

[2] it is not exactly the same *fluid* model as [19] where functions are continuous rather than left-continuous

A function $\alpha \in \mathcal{F}$ is an *arrival curve* for $F$ if $\forall\ s,t \in \mathbb{R}_+,\ s \leq t$, we have $F(t) - F(s) \leq \alpha(t-s)$. We also say that $F$ is *upper-constrained* by $\alpha$. It means that the number of bits arriving between time $s$ and $t$ is at most $\alpha(t-s)$. A typical example of arrival curve is the affine function $\alpha_{\sigma,\rho}(t) = \sigma + \rho t$, $\sigma, \rho \in \mathbb{R}_+$ (sometimes called *leaky-bucket* arrival curve), where $\sigma$ represents the maximal number of packets that can arrive simultaneously (the maximal burst) and $\rho$ the maximal long-term rate of arrivals. An usual assumption is that $\alpha$ is sub-additive (otherwise it can be replaced by its sub-additive closure). Note that the sub-additive closure of our leaky-bucket curve $\alpha_{\sigma,\rho}$ is still $\alpha_{\sigma,\rho}$ except at $t = 0$ where the value must be 0.

### 2.1.3 NC service curves

Two types of minimum service curves are commonly considered: *simple service curves* and *strict service curves*. Given a trajectory $(F^{in}, F^{out})$ of an input/output system, we need to define the notion of *backlogged period* which is an interval $I \subseteq \mathbb{R}_+$ such that $\forall u \in I$, $F^{in}(u) - F^{out}(u) > 0$. Given $t \in \mathbb{R}_+$, the *start of the backlogged period* of $t$ is $start(t) = \sup\{u < t | F^{in}(u) = F^{out}(u)\}$. Since $F^{in}$ and $F^{out}$ are left-continuous, we also have $F^{in}(start(t)) = F^{out}(start(t))$. For instance, for any $t \in \mathbb{R}_+$, $]start(t), t[$ is a backlogged period, as well as $]start(t), t]$ if $F^{in}(t) - F^{out}(t) > 0$.

Let $\beta \in \mathcal{F}$, we define:

- $\mathcal{S}_{simple}(\beta) = \{(F^{in}, F^{out}) \in \mathcal{F} \times \mathcal{F} \mid F^{in} \geq F^{out} \geq F^{in} * \beta\}$;
- $\mathcal{S}_{strict}(\beta) = \{(F^{in}, F^{out}) \in \mathcal{F} \times \mathcal{F} \mid F^{in} \geq F^{out}$ and for any backlogged period $]s,t[,\ F^{out}(t) - F^{out}(s) \geq \beta(t-s)\}$.

We say that a system $S$ provides a (minimum) *simple service curve* (resp. *strict service curve*) $\beta$ if $S \subseteq \mathcal{S}_{simple}(\beta)$ (resp. $S \subseteq \mathcal{S}_{strict}(\beta)$). A typical example of service curve is the *rate-latency* function: $\beta_{R,T}(t) = R(t-T)_+$ where $R, T \in \mathbb{R}_+$ and $a_+$ denotes $\max(a, 0)$. Note that for all $\beta \in \mathcal{F}$, $\mathcal{S}_{strict}(\beta) \subseteq \mathcal{S}_{simple}(\beta)$ (since for all $(F^{in}, F^{out}) \in \mathcal{S}_{strict}(\beta)$, we have $\forall t \in \mathbb{R}_+$, $F^{out}(t) \geq \min(F^{in}(t), F^{in}(start(t)) + \beta(t - start(t))) \geq (F^{in} * \beta)(t))$. But the converse is not true [19]. An usual assumption for strict service curves is that $\beta$ is super-additive (otherwise it can be replaced by its super-additive closure).

In NC models with multiplexing, the aggregation of all the flows entering the system is often considered as a single flow to which the minimum service is applied (*i.e.* one works with the sum of the cumulative functions). This is the case here.

### 2.1.4 Performance characteristics and bounds

Given a input/output system, bounds for the worst backlog and worst delay can be easily read from arrival and service curves.

Given a flow going through a network, modeled by an input/output system $S$, let $(F^{in}, F^{out})$ be a trajectory of $S$. The *backlog* of the flow at time $t$ is $b(t) =$

$F^{in}(t) - F^{out}(t)$, and the delay endured by data entering at time $t$ (assuming FIFO policy for the flow) is

$$\begin{aligned} d(t) &= \inf\{s \geq 0 \mid F^{in}(t) \leq F^{out}(t+s)\} \\ &= \sup\{s \geq 0 \mid F^{in}(t) > F^{out}(t+s)\}. \end{aligned}$$

Beware that $d(t) \neq \sup\{s \geq 0 \mid F^{in}(t) \geq F^{out}(t+s)\}$. For instance, look at $t = 1$ for $F^{in}(t) = t$ and $F^{out}(t) = 0$ over $[0, 2]$ and $= 1$ over $]2, 4]$ and $= t$ over $]4, +\infty[$.

For the trajectory, the *worst backlog* is $B_{\max} = \sup_{t \geq 0} \left(F^{in}(t) - F^{out}(t)\right)$ and the *worst delay* is $D_{\max} = \sup_{t \geq 0} d(t) = \sup\{t - s \mid F^{in}(s) > F^{out}(t)\}$.

For the system $S$, the *worst backlog* (resp. *delay*) is the supremum over all its trajectories.

The next theorem explains how to derive performance bounds from constraints and how traffic constraints can be propagated.

**Theorem 1** ([12, 19])**.** *Let $S$ be an input/output system providing a simple service curve $\beta$ and let $(F^{in}, F^{out})$ be a trajectory such that $\alpha$ is an arrival curve for $F^{in}$. Then,*

1. $B_{\max} \leq \sup\{\alpha(t) - \beta(t) \mid t \geq 0\} = v(\alpha, \beta)$ *(vertical distance).*
2. $D_{\max} \leq \inf\{d \geq 0 \mid \forall t \geq 0,\ \alpha(t) \leq \beta(t+d)\} = h(\alpha, \beta)$ *(horizontal distance).*
3. $\alpha \oslash \beta$ *is an arrival curve for $F^{out}$.*

Fig. 1 illustrates those bounds.

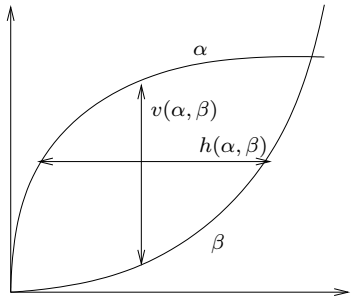


Figure 1: Guaranteed upper bounds on backlog and delay.

## 2.2 Network model

A network will be modeled, without loss of generality, by a directed graph where the flows must follow the edges and the servers (switches, transmission links, routers...) are represented by the vertices.

Servers and flows will be identified by indices. The set of servers is $\mathbb{S} = \{1, \ldots, n\}$ and each server $j$ offers a *strict* service curve $\beta_j \in \mathcal{F}$ which is piecewise affine (with a finite number $|\beta_j|$ of pieces) and convex. Thus it can be written $\beta_j(t) = \max_{1 \le \ell \le |\beta_j|}(r_{j,\ell}t - h_{j,\ell})$ with $r_{j,\ell}, h_{j,\ell} \in \mathbb{R}_+$.

The set of flows is $\mathbb{F} = \{1, \ldots, p\}$. Each flow $i$ corresponds to a couple $(\alpha_i, \mu_i)$ where $\mu_i$ is the (finite) ordered sequence of servers crossed by the flow and $\alpha_i$ is an arrival curve for the cumulative function before entering the first server. We suppose that $\alpha_i$ is non-negative, non-decreasing, piecewise affine (with a finite number $|\alpha_i|$ of pieces) and concave [3]. Thus it can be written $\alpha_i(t) = \min_{1 \le \ell \le |\alpha_i|}(\sigma_{i,\ell} + \rho_{i,\ell}t)$ with $\sigma_{i,\ell}, \rho_{i,\ell} \in \mathbb{R}_+$. Let $i \in \mathbb{F}$, we denote $\text{first}(i)$ (resp. $\text{last}(i)$) the index of the first (resp. last) server encountered by flow $i$. We will by abuse of notation write $j \in i$ to say that server $j$ belongs to the sequence $\mu_i$. Given $j \in i$, we will denote $\text{prec}_i(j)$ the index of the server preceding $j$ in the sequence $\mu_i$ (by convention, $\text{prec}_i(\text{first}(i)) = 0$). These notations are illustrated on Fig. 2. Note that in the following indices of flows will always use the letter $i$ and indices of servers will always use the letter $j$. We write $i \ni j$ to say that flow $i$ crosses server $j$.

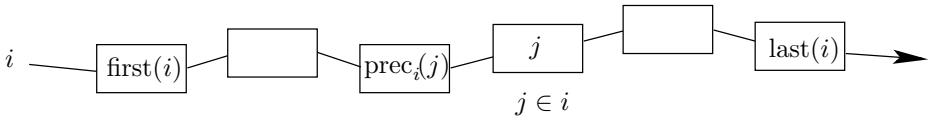


Figure 2: Notations about flow $i$.

The overall network $\mathcal{N}$ is defined by $\mathbb{S}$, $\mathbb{F}$ and the sets $\{\beta_j,\ 1 \le j \le n\}$, $\{(\alpha_i, \mu_i),\ 1 \le i \le p\}$. The directed graph induced by $\mathcal{N}$ is $\mathcal{G}(\mathcal{N}) = (\mathbb{S}, \mathbb{A})$, where $\mathbb{S}$ is the set of vertices and $(j, j') \in \mathbb{A}$ if and only if $j$ and $j'$ are consecutive servers for some sequence $\mu_i$. The sequences $\mu_i$ are paths in the digraph (the converse is not necessarily true). Any path in $\mathcal{G}(\mathcal{N}) = (\mathbb{S}, \mathbb{A})$ will be designated by its ordered sequence of vertices and often written as a *word over the alphabet* $\mathbb{S}$ (**we will often use $\pi$ to designate a path and for instance $j\pi$ will denote the path starting by vertex $j$ followed by the path $\pi$**).

In addition, we will use the following notations (similar to [28]): for all $i \in \mathbb{F}$,

- the cumulative function of flow $i$ at the entry of network is $F_i^{(0)}$;
- for all $j \in i$, the cumulative function of flow $i$ at the output of server $j$ is $F_i^{(j)}$.

A set of cumulative functions $\{F_i^{(j)} \in \mathcal{F} \mid i \in \mathbb{F},\ j \in \mathbb{S},\ j \in i\}$ will be called a *trajectory of the network* $\mathcal{N}$ if it respects the NC constraints of the network:

(T1) $\forall i \in \mathbb{F}, \forall j \in i,\ F_i^{(\text{prec}_i(j))} \ge F_i^{(j)}$;

[3] up to forcing a null value at $t = 0$, it belongs to $\mathcal{F}$, however we do not force this value to keep an expression as a minimum of affine functions (it will be useful for the future LP translation of constraints)

(T2) $\forall i \in \mathbb{F}$, $F_i^{(0)}$ is $\alpha_i$-upper constrained;

(T3) $\forall j \in \mathbb{S}$, $(\sum_{i \ni j} F_i^{(\mathrm{prec}_i(j))}, \sum_{i \ni j} F_i^{(j)}) \in \mathcal{S}_{strict}(\beta_j)$.

Since we work under blind multiplexing, no other relation is imposed. The set of all trajectories of $\mathcal{N}$ is denoted $\mathrm{Traj}(\mathcal{N})$.

Here are our objectives:

1. **Worst end-to-end delay.** Given a flow $i_0$, we wish to compute the worst end-to-end delay endured by data of this flow, that is

$$\sup_{\{F_i^{(j)}\} \in \mathrm{Traj}(\mathcal{N})} \sup_{0 \le s \le t} \{t - s \mid F_{i_0}^{(0)}(s) > F_{i_0}^{(\mathrm{last}(i_0))}(t)\}.$$

   In fact, we will rather compute

$$\sup_{\{F_i^{(j)}\} \in \mathrm{Traj}(\mathcal{N})} \sup_{0 \le s \le t} \{t - s \mid F_{i_0}^{(0)}(s) \ge F_{i_0}^{(\mathrm{last}(i_0))}(t)\}.$$

   This will not change the result, as we always assume that one bit of data can arrive instantaneously (for example, even if $\alpha_{i_0} = 0$, we compute the maximum delay of one bit of data of flow $i_0$ crossing the system).

2. **Worst local backlog.** Given a server $j_0$, we wish to compute the worst backlog endured by this server, that is

$$\sup_{\{F_i^{(j)}\} \in \mathrm{Traj}(\mathcal{N})} \sup_{t \ge 0} \{\sum_{i \ni j_0} F_i^{(\mathrm{prec}_i(j_0))}(t) - \sum_{i \ni j_0} F_i^{(j_0)}(t)\}.$$

Those are supremum over infinite sets, nevertheless they can be computed as shown next. Note that those supremum are not necessarily reached for some trajectory or some instants $s, t$. For example, for a single flow crossing a single server, let $F_1^{(0)}(t) = t$ and $F_1^{(1)}(t) = 2(t-1)_+$ then the worst delay is 1 but it is not reach for any instants $s, t$ (it occurs when $t = 1$ and $s$ tends to 0). In the same way, if $F_1^{(0)}(0) = 0$, $F_1^{(0)}(t) = 1$ for $t > 0$ and $F_1^{(1)}(t) = t$, the worst backlog is 1 but it is not reach for any instant $t$ (it occurs when $t$ tends to 0).

Note also that, when dealing with complexity, we will assume that numerical parameters take their values in $\mathbb{Q}$ rather than $\mathbb{R}$.

# 3 Analysis of general feed-forward networks

**Theorem 2.** *Let $\mathcal{N}$ be a network with $n$ servers and $p$ flows. If its induced graph $\mathcal{G}(\mathcal{N})$ is feed-forward, then given a flow $i$ (resp. a server $j$), there exists a finite set $\Lambda$ of linear programs (LP) with respective optimum values $opt_\lambda$, $\lambda \in \Lambda$, such that $\max_{\lambda \in \Lambda} opt_\lambda$ is the worst end-to-end delay for flow $i$ (resp. worst backlog at server $j$). Each linear program has $\mathcal{O}(p|\Pi|)$ variables and $\mathcal{O}(p|\Pi|^2)$ linear constraints where $\Pi$ is the set of paths ending at end$(i)$ (resp. $j$). We have $|\Pi| \le 2^{n-1}$ and $|\Lambda| \le |\Pi|! 2^{|\Pi|-1}$.*

The description of the different LP instances and the proof of the theorem will be illustrated with the small but typical example of Fig. 3, the *diamond network*.

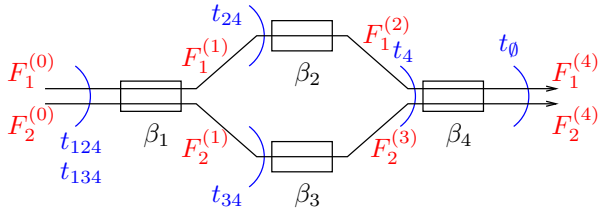


Figure 3: Diamond network: two flows and four servers.

## 3.1 The LP instances

We present a set of LP instances such that any trajectory satisfies at least one of them.

### 3.1.1 Variables

Given the flow of interest $i$ (resp. a server $j$) for which one wants to compute the maximum end-to-end delay (resp. the maximum backlog) over all trajectories of the network, let $\Pi$ be the set of all paths ending at last$(i)$ (resp $j$), including the empty path denoted $\emptyset$. Here are the variables names that will appear in each LP instance:

- $t_\pi$ for all $\pi \in \Pi$. **Interpretation:** $t_\emptyset$ is the instant at which the worst-case occurs (output of data having endured the worst delay or instant of worst backlog). Then for all $j\pi \in \Pi$, $t_{j\pi} = start_j(t_\pi)$, the start of the backlogged period before $t_\pi$ at the server $j$.
- $F_i^{(j)}(t_\pi)$ for all $i \in \mathbb{F}$, $j \in i$, $\pi \in \Pi_i^{(j)} := \{\pi', j\pi' \mid j\pi' \in \Pi\}$. **Interpretation:** the value of the cumulative function $F_i^{(j)}$ at time $t_\pi$.
- $F_i^{(0)}(t_\pi)$ for all $i \in \mathbb{F}$, $\pi \in \Pi_i := \bigcup_{j \in i} \Pi_i^{(j)}$. **Interpretation:** the value of the cumulative function $F_i^{(0)}$ at time $t_\pi$.

**Diamond example**

- If we are studying the worst end-to-end delay for flow 1 (or for flow 2, or the worst backlog at server 4), we have to consider the set of paths ending at server 4, *i.e.* $\Pi = \{\emptyset, 4, 24, 34, 124, 134\}$. Thus the temporal variables are $t_\emptyset$, $t_4$, $t_{24}$, $t_{34}$, $t_{124}$, $t_{134}$.
- For flow 1 and server 1, the set $\Pi_1^{(1)}$ is $\{124, 24, 134, 34\}$. Thus the corresponding variables are $F_1^{(1)}(t_{124})$, $F_1^{(1)}(t_{24})$, $F_1^{(1)}(t_{134})$, $F_1^{(1)}(t_{34})$.

- For flow 1, we have $\Pi_1 = \{\emptyset, 4, 24, 34, 124, 134\}$. Thus the corresponding input variables are $F_1^{(0)}(t_{124})$, $F_1^{(0)}(t_{24})$, $F_1^{(0)}(t_{134})$, $F_1^{(0)}(t_{34})$, $F_1^{(0)}(t_4)$, $F_1^{(0)}(t_\emptyset)$.

---

Note that one could have introduced the variables $F_i^{(j)}(t_\pi)$ for all $\pi \in \Pi$, but many of them do not occur while describing the sequence of events leading to the worst case and are not necessary to rebuild critical trajectories.

In the text, we will carefully distinguish the expression "the variable $x$" which refers to the unassigned variable, from the expression "the value $x$" which refers to an assignment of the variable $x$.

### 3.1.2 Temporal constraints

A *set of temporal constraints* $\mathcal{T}$ over some subset $\Pi' \subseteq \Pi$ is a set of equalities or inequalities of the form $t_{\pi_1} = t_{\pi_2}$ or $t_{\pi_1} \leq t_{\pi_2}$ where $\pi_1, \pi_2 \in \Pi'$, and such that its set of solutions $\mathrm{Sol}(\mathcal{T}) \subseteq \mathbb{R}_+^{\Pi'}$ is non-empty.

To ensure the coherence of the $t_\pi$ values with their interpretation as starts of backlogged periods, we introduce two predicates over the $t_\pi$ variables:

(P1) For all $j\pi \in \Pi$, $t_{j\pi} \leq t_\pi$.

(P2) For all $j\pi_1, j\pi_2 \in \Pi$, $t_{j\pi_1} < t_{j\pi_2} \Rightarrow t_{j\pi_1} \leq t_{\pi_1} \leq t_{j\pi_2} \leq t_{\pi_2}$.

The predicate (P1) comes from the fact that for any trajectory and any instant $t$, at server $j$, we have $start(t) \leq t$. It is also clear that for any instants $t, t'$, at server $j$, an ordering $start(t) < start(t') \leq t$ can not occur, leading to predicate (P2).

We say that a set of temporal constraints $\mathcal{T}$ over $\Pi'$ satisfies the predicates (P1) and (P2) if any solution to $\mathcal{T}$ with real values satisfies both (P1) and (P2). Note that this definition involves an infinite number of tests, but one can easily decide whether $\mathcal{T}$ satisfies (P1) and (P2) with a $\mathcal{O}(|\mathcal{T}||\Pi'|)$ algorithm (one can perform a transitive closure algorithm on $\mathcal{T}$ to get all existing comparisons between the variables $t_\pi$, $\pi \in \Pi'$, with complexity $\mathcal{O}(|\mathcal{T}||\Pi'|)$, then checking (P1) is immediate and checking (P2) comes to check that there is no $t_{j\pi_1} < t_{j\pi_2} \leq t_{\pi_1}$ configuration, it can be done in $\mathcal{O}(|\mathcal{T}|^2)$).

We say that a set of temporal constraints $\mathcal{T}$ is a *total order* over some subset $\Pi' \subseteq \Pi$ if it has the form $\{t_{\pi_1} \lhd_1 t_{\pi_2} \lhd_2 \cdots \lhd_{N-1} t_{\pi_N}\}$ where $\pi_1, \pi_2, \ldots, \pi_N$ is a permutation of all the elements of $\Pi'$ and for all $1 \leq k \leq N-1$, $\lhd_k \in \{=, \leq\}$. Note that for all $\pi, \pi' \in \Pi'$, by considering the transitive closure, $\mathcal{T}$ implies a comparison between $\pi$ and $\pi'$ which is either $=$, $\leq$ or $\geq$. This comparison is denoted $\mathcal{T}(\pi, \pi')$. The set of all total orders over $\Pi'$ which satisfy (P1) and (P2) is denoted $\mathrm{Tot}(\Pi')$. Here is one way to enumerate all the elements of $\mathrm{Tot}(\Pi')$: generate the set of temporal constraints imposed by predicate (P1), it corresponds to a tree-like partial order, then generate all its linear extensions, generate for each linear extension all the possible combinations of comparisons $=$, or $\leq$ and for each one check whether it satisfies (P2). Such an algorithm

works and it has roughly a $\mathcal{O}(|\mathrm{Tot}(\Pi')|3^{|\Pi'|}|\Pi'|^2)$ complexity [23]. We have not looked for a faster algorithm, there is probably some ways to speed up this step, but any algorithm will require at least a $|\mathrm{Tot}(\Pi')|$ complexity which can be exponential w.r.t. $|\Pi'|$.

Now we come back to our network. For each flow $i$, we have associated the set of paths $\Pi_i = \{\pi, j\pi \mid j \in i,\ j\pi \in \Pi\}$ and we know that $\Pi = \bigcup_{1 \leq i \leq n} \Pi_i$. Let $(\mathcal{T}_1, \ldots, \mathcal{T}_p) \in \mathrm{Tot}(\Pi_1) \times \cdots \times \mathrm{Tot}(\Pi_p)$, we say that the total orders $(\mathcal{T}_1, \ldots, \mathcal{T}_p)$ are *mutually compatible* if for all $1 \leq i_1, i_2 \leq p$ and $\pi, \pi' \in \Pi_{i_1} \cap \Pi_{i_2}$, we have $\mathcal{T}_{i_1}(\pi, \pi') = \mathcal{T}_{i_2}(\pi, \pi')$. Note that it can be checked with a $\mathcal{O}(p|\Pi|^2)$ algorithm looking at all pairs $\pi, \pi'$. This condition ensures that there exists a solution $(t_\pi)_{\pi \in \Pi} \in \mathbb{R}_+^\Pi$ to the set of constraints $\mathcal{T}_1 \cup \ldots \cup \mathcal{T}_p$. Moreover one can easily prove that this solution will always satisfy the predicates (P1) and (P2).

Each combination $(\mathcal{T}_1, \ldots, \mathcal{T}_p) \in \mathrm{Tot}(\Pi_1) \times \cdots \times \mathrm{Tot}(\Pi_p)$ of mutually compatible total orders will lead to a set of LP instances. The main issue leading to such a case study is that, unlike (P1), the predicate (P2) cannot be captured by a single set of linear constraints.

Note that to avoid the analysis of redundant cases, one may only consider set of constraints $\mathcal{T}_i$ such that their set of solutions $\mathrm{Sol}(\mathcal{T}_i)$ is maximal for the inclusion (*e.g.* among two eligible sets $\mathcal{T}^1 = \{t_{12} = t_{13} \leq t_2 = t_3\}$ and $\mathcal{T}^2 = \{t_{12} = t_{13} \leq t_2 \leq t_3\}$, only $\mathcal{T}^2$ needs to be considered).

**Example** The set $\Pi$ is $\{\emptyset, 4, 24, 34, 124, 134\}$ and $\Pi_1 = \Pi_2 = \Pi$.

To satisfy predicate (P1), one has the relations:

$$t_{124} \leq t_{24} \leq t_4 \leq t_\emptyset \text{ and } t_{134} \leq t_{34} \leq t_4 \leq t_\emptyset.$$

Now $t_{124}$ and $t_{134}$ need to be ordered. There are four maximal total orders that also satisfy predicate (P2):

- $\mathcal{T}^1 = \{t_{124} \leq t_{24} \leq t_{134} \leq t_{34} \leq t_4 \leq t_\emptyset\}$;
- $\mathcal{T}^2 = \{t_{134} \leq t_{34} \leq t_{124} \leq t_{24} \leq t_4 \leq t_\emptyset\}$;
- $\mathcal{T}^3 = \{t_{124} = t_{134} \leq t_{24} \leq t_{34} \leq t_4 \leq t_\emptyset\}$;
- $\mathcal{T}^4 = \{t_{124} = t_{134} \leq t_{34} \leq t_{24} \leq t_4 \leq t_\emptyset\}$.

---

### 3.1.3 Trajectory constraints

Let $(\mathcal{T}_1, \ldots, \mathcal{T}_p) \in \mathrm{Tot}(\Pi_1) \times \cdots \times \mathrm{Tot}(\Pi_p)$ be some mutually compatible total orders.

Here is the set of equalities and inequalities describing the states of the system for our selected events:

- *Temporal constraints:* $\mathcal{T} = \mathcal{T}_1 \cup \ldots \cup \mathcal{T}_p$.

- *Strict service constraints*: for all $j \in \mathbb{S}$ and $j\pi \in \Pi$, add $\{\sum_{i \ni j} F_i^{(j)}(t_\pi) - \sum_{i \ni j} F_i^{(j)}(t_{j\pi}) \geq \beta_j(t_\pi - t_{j\pi})\}$ (that is $|\beta_j|$ linear inequalities since $\beta_j$ is a maximum of affine functions). Moreover for all $j\pi_1, j\pi_2 \in \Pi$ such that $\mathcal{T}(j\pi_1, j\pi_2)$ is $=$ and $\mathcal{T}(\pi_1, \pi_2) \in \{=, \leq\}$, add $\{\sum_{i \ni j} F_i^{(j)}(t_{\pi_2}) - \sum_{i \ni j} F_i^{(j)}(t_{\pi_1}) \geq \beta_j(t_{\pi_2} - t_{\pi_1})\}$.
- *Starts of backlogged periods:* for all $j \in \mathbb{S}$, $j\pi \in \Pi$ and $i \ni j$, add $\{F_i^{(\mathrm{prec}_i(j))}(t_{j\pi}) = F_i^{(j)}(t_{j\pi})\}$.
- *Flow constraints:* for all $i \in \mathbb{F}$, $j \in i$ and $j\pi \in \Pi$, add $\{F_i^{(0)}(t_{j\pi}) \geq F_i^{(j)}(t_{j\pi}),\ F_i^{(0)}(t_\pi) \geq F_i^{(j)}(t_\pi)\}$.
- *Non-decreasing functions:* for all $i \in \mathbb{F}$, $j \in i$ and $\pi_1, \pi_2 \in \Pi_i^{(j)}$, if $\mathcal{T}(\pi_1, \pi_2)$ is $=$, then add $\{F_i^{(j)}(t_{\pi_1}) = F_i^{(j)}(t_{\pi_2})\}$ and if $\mathcal{T}(\pi_1, \pi_2)$ is $\leq$, then add $\{F_i^{(j)}(t_{\pi_1}) \leq F_i^{(j)}(t_{\pi_2})\}$.
- *Arrival constraints:* for all $1 \leq i \leq p$, for all $\pi_1, \pi_2 \in \Pi_i$ such that $\mathcal{T}(\pi_1, \pi_2) \in \{=, \leq\}$, add $\{F_i^{(0)}(t_{\pi_2}) - F_i^{(0)}(t_{\pi_1}) \leq \alpha_i(t_{\pi_2} - t_{\pi_1})\}$ (that is $|\alpha_i|$ linear inequalities since $\alpha_i$ is a minimum of affine functions).

**Diamond example** For $\mathcal{T}^1 = \{t_{124} \leq t_{24} \leq t_{134} \leq t_{34} \leq t_4 \leq t_\emptyset\}$, flow constraints, non-decreasing constraints and starts of backlogged periods for flow 1, are depicted on Fig. 4.

$$
\begin{array}{ccccccccccc}
F_1^0(t_{124}) & \leq & F_1^0(t_{24}) & \leq & F_1^0(t_{134}) & \leq & F_1^0(t_{34}) & \leq & F_1^0(t_4) & \leq & F_1^0(t_\emptyset) \\
\| & & \wr\vee & & \| & & \wr\vee & & \wr\vee & & \wr\vee \\
F_1^1(t_{124}) & \leq & F_1^1(t_{24}) & \leq & F_1^1(t_{134}) & \leq & F_1^1(t_{34}) & & & & \\
 & & \| & & & & & & F_1^2(t_4) & & \\
 & & F_1^2(t_{24}) & \leq & & & & & \| & & \\
 & & & & & & & & F_1^4(t_4) & \leq & F_1^4(t_\emptyset)
\end{array}
$$

Figure 4: Constraints for flow 1 and $\mathcal{T}^1$ (except service/arrival constraints).

For server 1, strict service constraints are:

- $(F_1^{(1)}(t_{24}) + F_2^{(1)}(t_{24})) - (F_1^{(1)}(t_{124}) + F_2^{(1)}(t_{124})) \geq \beta_1(t_{24} - t_{124})$;
- $(F_1^{(1)}(t_{34}) + F_2^{(1)}(t_{34})) - (F_1^{(1)}(t_{134}) + F_2^{(1)}(t_{134})) \geq \beta_1(t_{34} - t_{134})$.

For flow 1, arrival constraints are:

- $F_1^{(0)}(t_{24}) - F_1^{(0)}(t_{124}) \leq \alpha_1(t_{24} - t_{124})$,

- $F_1^{(0)}(t_{134}) - F_1^{(0)}(t_{124}) \leq \alpha_1(t_{134} - t_{124})$,
- $F_1^{(0)}(t_{134}) - F_1^{(0)}(t_{24}) \leq \alpha_1(t_{134} - t_{24})$, ...

---

#### 3.1.4 Objective

**Worst end-to-end delay for flow $i_0$:** maximize the objective function $(t_\emptyset - u)$, where $t_\emptyset - u$ is the delay endured by data that entered the network at time $u$ and left at time $t_\emptyset$. Consequently one has to add several constraints linked to $u$ and possibly to consider several cases depending on the choice of $\mathcal{T}_{i_0}$ in $\mathrm{Tot}(\Pi_{i_0})$:

- *Arrival time:* $\{F_{i_0}^{(0)}(u) \geq F_{i_0}^{\mathrm{last}(i_0)}(t_\emptyset)\}$.
- *Insertion:* one must position $u$ within the total order $\mathcal{T}_{i_0}$. For each $\pi \in \Pi_{i_0}$, we generate one LP instance by adding the constraints:
  - *Position and monotony:* add $\{t_\pi \leq u,\ F_{i_0}^{(0)}(t_\pi) \leq F_{i_0}^{(0)}(u)\}$, and let $t_{\pi'}$ be the successor of $t_\pi$ in $\mathcal{T}_{i_0}$ (if any), add $\{u \leq t_{\pi'},\ F_{i_0}^{(0)}(u) \leq F_{i_0}^{(0)}(t_{\pi'})\}$.
  - *Arrival curve constraints:* for all $\pi' \in \Pi_{i_0}$, if $\mathcal{T}(\pi', \pi) \in \{=, \leq\}$, add $\{F_{i_0}^{(0)}(u) - F_{i_0}^{(0)}(t_{\pi'}) \leq \alpha_{i_0}(u - t_{\pi'})\}$, otherwise add $\{F_{i_0}^{(0)}(t_{\pi'}) - F_{i_0}^{(0)}(u) \leq \alpha_{i_0}(t_{\pi'} - u)\}$.

**Diamond example** Some possible particular positions of $u$ are $t_{124} \leq u \leq t_{24}$, $t_{24} \leq u \leq t_{134}$, ...

---

As a matter of fact, one can consider fewer cases. First there may be some equal elements in $\mathcal{T}_{i_0}$ which will yield the same additional constraints for $u$. Then more significantly, it is not necessary to consider all the possible positions of $u$ between all the consecutive elements of $\mathcal{T}_{i_0}$. One only has to position $u$ between consecutive elements of $Start_0 = \{t_{\mathrm{first}(i_0)\pi} \mid \mathrm{first}(i_0)\pi \in \Pi_{i_0}\}$ the set of all starts of backlogged periods at server $\mathrm{first}(i_0)$. This general idea is explained in the particular case of Theorem 4, but to ease a little the construction of critical trajectories, we keep all the cases mentioned before.

**Worst backlog at server $j_0$:** maximize the objective function

$$\sum_{i \ni j_0} F_i^{(\mathrm{prec}_i(j_0))}(t_\emptyset) - \sum_{i \ni j_0} F_i^{(j_0)}(t_\emptyset).$$

It does not introduce new cases or new linear constraints.

## 3.2 From network trajectories to LP solutions

Let $\lambda$ be an LP instance (possibly with strict inequalities) with optimal value $opt_\lambda$, we call a *solution* of $\lambda$ an assignation of the variables satisfying the linear constraints, and it is an *optimal solution* if it achieves $opt_\lambda$ (there may be no optimal solution if $\lambda = +\infty$ or when there are some strict inequalities in the constraints).

**Lemma 1.** *Let $\mathcal{N}$ be a feed-forward network and a flow of interest $i_0$ (resp. a server $j_0$). Let $\Lambda$ be the set of LP instances constructed in Section 3.1. Given a trajectory $\{F_i^{(j)}\} \in \mathrm{Traj}(\mathcal{N})$ where some data in flow $i_0$ is enduring an end-to-end delay $d$ (resp. the backlog at $j_0$ becomes $b$), then there exists an LP instance $\lambda \in \Lambda$ admitting a solution such that $t_\emptyset - u = d$ (resp. $\sum_{i \ni j_0} F_i^{(\mathrm{prec}_i(j_0))}(t_\emptyset) - \sum_{i \ni j_0} F_i^{(j_0)}(t_\emptyset) = b$).*

*Proof.* Consider $\{F_i^{(j)}\} \in \mathrm{Traj}(\mathcal{N})$, let $t_\emptyset$ be the instant at which the data enduring the delay $d$ leaves the networks (resp. at which the backlog is $b$). For the delay, there also exists in the trajectory a value $u$ such that that $u \leq t_\emptyset$, $F_{i_0}^{(0)}(u) > F_{i_0}^{(\mathrm{last}(i))}(t_\emptyset)\}$ and $d = t_\emptyset - u$.

Consider the interpretation of the variable $t_\pi$ given in Section 3.1.1 and find out their values from the trajectory (it requires to compute for each server $j$, the functions $\sum_{i \ni j} F_i^{(\mathrm{prec}_i(j))}$ and $\sum_{i \ni j} F_i^{(j)}$ to detect starts of backlogged periods). The values $\{t_\pi,\ \pi \in \Pi\}$ are totally ordered and satisfy the predicates (P1) and (P2). Thus there exists a family of time constraints $(\mathcal{T}_1, \ldots, \mathcal{T}_p) \in \mathrm{Tot}(\Pi_1) \times \cdots \times \mathrm{Tot}(\Pi_p)$ all satisfied by the set of values $\{t_\pi,\ \pi \in \Pi\}$, and which use only $=$ or $<$ constraints (they are just read from the values). As a consequence those time constraints are mutually compatible. Then for all $i$, $j$, $t_\pi$, read the value $F_i^{(j)}(t_\pi)$ from the trajectory. The choice of $(\mathcal{T}_1, \ldots, \mathcal{T}_p)$ (and for delays, the position of $u$ in $\mathcal{T}_{i_0}$) corresponds to a LP instance $\lambda \in \Lambda$.

By a careful but easy checking, the definition of the values $t_\pi$ (and $u$) and the fact that the trajectory satisfies (T1), (T2), (T3) (defined page 7) ensures that all the linear constraints in the LP instance $\lambda$ are satisfied. Moreover by definition of the value $t_\emptyset$ (and $u$), we have $t_\emptyset - u = d$ (resp. $\sum_{i \ni j_0} F_i^{(\mathrm{prec}_i(j_0))}(t_\emptyset) - \sum_{i \ni j_0} F_i^{(j_0)}(t_\emptyset) = b$).

□ □

## 3.3 From LP solutions to network trajectories

**Lemma 2.** *Let $\mathcal{N}$ be a feed-forward network and a flow of interest $i_0$ (resp. a server $j_0$). Let $\Lambda$ be the set of LP instances constructed in Section 3.1. Consider an instance $\lambda \in \Lambda$ and one of its solution. Then there exists a trajectory $\{F_i^{(j)}\} \in \mathrm{Traj}(\mathcal{N})$ where the worst end-to-end delay $d$ for flow $i_0$ (resp. worst backlog $b$ for server $j_0$) satisfies $d = t_\emptyset - u$ ($b = \sum_{i \ni j_0} F_i^{(\mathrm{prec}_i(j_0))}(t_\emptyset) - \sum_{i \ni j_0} F_i^{(j_0)}(t_\emptyset)$).*

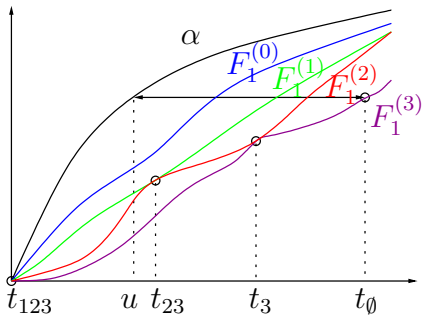


Figure 5: Reading $t_\pi$ on a trajectory for a 1-flow 3-servers scenario.

*Proof.* Before describing the construction of the trajectory $\{F_i^{(j)}\}$, we make a few remarks about our LP instances and their solutions.

**Existence of solutions** The affectation of variables where every variable is set to 0 is a solution of the problem.

**The date ordering issues** Since our LP instances $\lambda$ uses only non-strict inequalities for ordering the dates, the following case can occur: for some $\pi$ and $\pi'$ and $j$ we have the constraint (P2) $t_{j\pi} \leq t_\pi \leq t_{j\pi'} \leq t_{\pi'}$ with an assignation of constraints $t_{j\pi} = t_\pi = t_{j\pi'} < t_{\pi'}$ or $t_{j\pi} < t_\pi = t_{j\pi'} = t_{\pi'}$. In the first case, the equalities $t_{kj\pi} = t_{kj\pi'}$ are not ensured for $k = \mathrm{prec}_i(j)$ thus possibly lead to solution from which a trajectory cannot be reconstruct. In fact, such a solution would not lead to the maximum solution, as there is another LP $\lambda' \in \Lambda$ with $t_{j\pi} = t_{j\pi'}$. This LP $\lambda'$ only differs from $\lambda$, with this constraint and the removal of the variables $t_{\tilde{\pi}}$, where $\pi$ is a suffix of $\tilde{\pi}$. The maximal solution of $\lambda'$ is at least a solution of $\lambda$ where $t_{j\pi} = t_{j\pi'}$. In the second case, the same trick can be used by setting $t_{j\pi} = t_{j\pi'}$ - the service constraints are still satisfied as $F_i^{(j)}(t_{\pi'}) \geq F_i^{(j)}(t_\pi)$ - and removing the variables $t_{\tilde{\pi}}$, where $\pi'$ is a suffix of $\tilde{\pi}$. Form now on, we assume that a solution is such that whenever $t_\pi = t_{\pi'}$, $t_{j\pi} = t_{j\pi'}$.

**The assignation issue for functions** Since our LP instance $\lambda$ uses some non-strict inequalities, the following case may occur in a solution to $\lambda$: for some $i \in \mathbb{F}$, $j \in i$, $\pi, \pi' \in \Pi_i^{(j)}$, we may have the values $t_\pi = t_{\pi'} = t$ while $F_i^{(j)}(t_\pi) < F_i^{(j)}(t_{\pi'})$. A careful look at our LP instances shows that it can occur only if $\{t_\pi \leq t_{\pi'}\}$ was a time constraint (due to non-decrease constraints). Then we can transform our solution by replacing $F_i^{(j)}(t_\pi)$ by the value $F_i^{(j)}(t_{\pi'})$. Another careful look shows that such a transformation will not violate any constraint, thus we still have a solution to $\lambda$. Repeated if necessary, it comes to say that we will finally choose $F_i^{(j)}(t) = \max_{t_\pi = t} F_i^{(j)}(t_\pi)$ which is well-defined for $t$.

**The translation lemma and its application** Given a solution to $\lambda$ and an arbitrary constant $c_i \in \mathbb{R}_+$, let us replace all values $F_i^{(j)}(t_\pi)$ by the values $F_i^{(j)}(t_\pi) - c_i$ (we suppose that $c_i$ is sufficiently small so that those values remain non-negative). Then one can easily check that this new assignment of variables is still a solution to $\lambda$.

Given a solution to $\lambda$, we use this translation lemma for each flow $i$. Each set of values $\{t_\pi,\ \pi \in \Pi_i\}$ admits a minimum $t_{\pi min(i)}$. Due to LP constraints, for all $j \in i$, the value $F_i^{(0)}(t_{\pi min(i)})$ is lower or equal to all the other values $F_i^{(j)}(t_\pi)$. We set $c_i = F_i^{(0)}(t_{\pi min(i)})$ and subtract $c_i$ from all the values $F_i^{(j)}(t_\pi)$. Once done for all flows, we still have a solution to $\lambda$, but now for all $i \in \mathbb{F}$, we have $F_i^{(0)}(t_{\pi min(i)}) = 0$.

This transformation ensures that one can add the point $(0, 0)$ among the points $(t_\pi, F_i^{(0)}(t_\pi))$, $\pi \in \Pi_i$, and still satisfy the arrival curve constraints (since $F_i^{(0)}(t_\pi) - 0 = F_i^{(0)}(t_\pi) - F_i^{(0)}(t_{\pi min(i)}) \leq \alpha(t_\pi - t_{\pi min(i)}) \leq \alpha(t_\pi) - 0)$).

**The linear interpolation lemma**

**Lemma 3.** *Let $\alpha$ (resp. $\beta$) be a concave (resp. convex) function in $\mathcal{F}$. Let $x_1 \leq x_2 \leq \cdots \leq x_k$ and $y_1 \leq y_2 \leq \cdots \leq y_k$ be two sets of values in $\mathbb{R}_+$, such that $\forall 1 \leq \ell \leq \ell' \leq k$, $y_{\ell'} - y_\ell \leq \alpha(x_{\ell'} - x_\ell)$ (resp. $y_{\ell'} - y_\ell \geq \beta(x_{\ell'} - x_\ell)$). Let $F$ from $[x_1, x_k]$ into $\mathbb{R}_+$ be the linear interpolation of the points $(x_1, y_1), (x_2, y_2)$, $\ldots, (x_k, y_k)$. Then $F$ is non-decreasing and satisfies for all $x_1 \leq s \leq t \leq x_k$, $F(t) - F(s) \leq \alpha(t - s)$ (resp. $F(t) - F(s) \geq \beta(t - s)$).*

*Proof.* We first prove the result for a concave function $\alpha$.

Let $t \in [x_p, x_{p+1}]$ for some $p \in \{1, \ldots, k-1\}$. First, $\forall i \leq p$, $F(t) - F(x_i) \leq \alpha(t - x_i)$: the set $C_i = \{(x, y) \mid x \in [x_i, x_{p+1}],\ y \leq y_i + \alpha(x - y_i)\}$ is convex and contains $(x_p, y_p)$ and $(x_{p+1}, y_{p+1})$. Then, it contains the cord between those two points and $(t, F(t))$.

Let $s \in [x_q, x_{q+1}]$ such that $s \leq t$. The set $C = \{(x, y) \mid x \in [x_1, t],\ y \geq F(t) - \alpha(t - x)\}$ is a convex set that contains $(x_q, y_q)$ and $(x_{q+1}, y_{q+1})$. Then, it contains the cord between those two points and $(s, F(s))$.

Then, $F(t) - F(s) \leq \alpha(t - s)$.

The case of convex function is simply proved by replacing $\alpha$ by $-\alpha$ and $F$ by $-F$.

□ □

**Reconstruction of cumulative functions for each flow** Given a solution to $\lambda$, apply first all the transformations described above. Then consider a flow $i$, all functions $F_i^{(j)}$, $j \in i$ or $j = 0$, will be null from 0 to $t_{\pi min(i)}$ the minimum value of $\{t_\pi,\ \pi \in \Pi_i\}$. Now we describe each $F_i^{(j)}(t)$, $j \in i$ or $j = 0$, for $t \geq t_{\pi min(i)}$.

The function $F_i^{(0)}$ is the linear interpolation of the points $\{(t_\pi, F_i^{(0)}(t_\pi)) \mid \pi \in \Pi_i\}$ and remains constant from its maximum point.

Now we construct the functions $F_i^{(j)}$ by induction while moving forward on the path $\mu_i$. Suppose that $F_i^{(\mathrm{prec}_i(j))}$ has been constructed. To construct $F_i^{(j)}$, we first have to define some intervals where data from flow $i$ is backlogged at server $j$. Consider the values $\{t_{j\pi} \mid j\pi \in \Pi_i\}$ which correspond to the starts of such backlogged periods. Some of these values can be equal, just consider the distinct values denoted $t_1 < \cdots < t_m$. By convention we denote $t_{m+1} = +\infty$. Then to each $t_v$ we associate $t_v^+ = \max\{t_\pi \mid t_v \leq t_\pi < t_{v+1},\ \pi \in \Pi_i^{(j)}\}$ and the points $P_v = \{(t_\pi, F_i^{(j)}(t_\pi)) \mid t_v \leq t_\pi < t_{v+1},\ \pi \in \Pi_i^{(j)}\}$. Then $F_i^{(j)}$ is defined as the linear interpolation of the points $P_v$ over each interval $[t_v, t_v^+]$ and is equal to $F_i^{(\mathrm{prec}_i(j))}$ otherwise.

Let $j_{\max}$ be the last server from which there is a path in $\Pi$. Then, for all the next $j$ on $\mu_i$, we set $F_i^{(j)} = F_i^{(j_{\max})}$.

Once these constructions are done for all the flows, one can carefully check that this family of functions which belong to $\mathcal{F}$ satisfies (T1), (T2), (T3) (defined 7), thanks to the preceding remarks and lemmas which enable to extend the trajectory constraints from a finite set of points to functions defined over $\mathbb{R}_+$.

This construction also ensures that the trajectory has a worst end-to-end delay for flow $i_0$ equal to $t_\emptyset - u$ (resp. worst backlog for server $j_0$ equal to $\sum_{i \ni j_0} F_i^{(\mathrm{prec}_i(j_0))}(t_\emptyset) - \sum_{i \ni j_0} F_i^{(j_0)}(t_\emptyset)$). It comes from the fact that $F_{i_0}^{(\mathrm{last}(i_0))}(t) = F_{i_0}^{(0)}(t)$ for $t \geq t_\emptyset$.

□
□

Note that the statements of Lemma 1 and Lemma 2 prove Theorem 2 while avoiding the question of unbounded delays or backlogs. The LP solvers will output a $+\infty$ result if it is actually the worst-case. However if one wish to state a theorem characterizing scenarios with such $+\infty$ worst end-to-end delays or local backlogs, rather than analyzing in details the properties of our set of LP instances, we prefer to refer to the following classical result: in a feed-forward network with a FIFO-per-flow policy, for any flow $i$ (resp. server $j$) the worst end-to-end delay (resp. local backlog) is bounded if and only if each server on a path leading to last($i$) (resp. $j$) has an utilization factor (asymptotic service rate/ sum of asymptotic arrival rates) which is $< 1$, as already proved in [19].

### 3.4 Computational hardness

**Theorem 3.**
- *Computing the worst backlog at a given server in a feed-forward network is NP-hard.*
- *Computing the worst delay at a given server in a feed-forward network is NP-hard.*

*Proof.* We reduce the problem "exact three-cover" (X3C) to our problem. An instance of X3C is a collection $\mathcal{C} = \{c_1, \ldots, c_{3q}\}$ of $3q$ elements and a collection $\mathcal{U} = \{u_1, \ldots, u_s\}$ of $s$ sets of 3 elements of $\mathcal{C}$. The problem is to decide whether

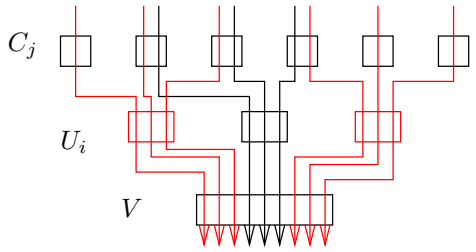


Figure 6: Transformation of an instance of X3C into a network.

there exists a cover of $\mathcal{C}$ by $q$ elements of $\mathcal{U}$. We will reduce this problem to deciding whether a given backlog or delay can be reached in a server of a network.

The network we use is as shown in Figure 6. The upper stage consists of $3q$ servers $C_1, \ldots, C_{3q}$, all with service curve $\beta_1 : t \mapsto t$. The middle stage consists of $s$ servers $U_1, \ldots, U_s$, all with service curve $\beta_2 : t \mapsto 2t$. Finally, the lower stage has only one server $V$, with service curve $\beta_3 : t \mapsto Rt$ with $R > 3s$. There are $3s$ flows, each of them crossing three servers from top to bottom. A flow, $F_{i,j}$ crosses servers $C_j$, $U_i$, $V$ if and only if $c_j \in u_i$. Each of those interfering flows has an arrival curve $\alpha : t \mapsto \min(t, 1)$ (note that there can be no burst).

**NP-hardness of the maximum backlog.** One wants to decide whether the backlog in $V$ can be at least $3s - 2q$.

The worst-case backlog in server $V$ will be obtained when flows arrive according to $\alpha$ from time 0, and when servers $C_j$ and $U_i$ are exact servers.

Indeed, fix 0 as the date of arrival of the first bit of data in the system. The worst-case backlog is obtained at time 1. It cannot be obtained at time $s < 1$: every bit of data has not yet been sent into the system, and if some backlog could be created, some more can be created until time 1. Now, suppose that the worst case backlog is obtained at time $s > 1$. Consider the following transformation: remove a flow among those that send their data first and the two other flows that share the same server at the upper stage, and make them arrive from time $s - 1$. Let $q$ be the quantity of data that is served initially by these flows. After being delayed, the quantity of data that is served at the first stage decreases. At the middle stage, less data arrive, and later. Then, the quantity of data that is served also decreases. As a consequence there is a worst-case trajectory with $s = 1$.

Consider an infinitesimal time interval, during which each flow $F_{i,j}$ at the upper stage is served at rate $r_{i,j}$, with $\sum_{i:c_j \in u_i} r_{i,j} = 1$. The service rate of $U_i$ is $\min(2, \sum_{j:c_j \in u_i} r_{i,j})$ and then the increase rate of the backlog during that time interval at the middle stage is $\sum_{i \in \{1,\ldots s\}} (\sum_{j:c_j \in u_i} r_{i,j} - 2)_+$. At the upper stage, the backlog is created at rate $3s - \sum_{i,j} r_{ij} = 3(s - q)$, which does not depend on the flows that are served. Maximizing the backlog is equivalent to maximizing the following function:

$$\sum_{i\in\{1,\ldots s\}} \Big[ \sum_{j:c_j\in u_i} r_{i,j} - 2\Big]_+$$

with the constraints: $\forall j \in \{1,\ldots,3q\}$, $\sum_{i:c_j\in u_i} r_{i,j} = 1$ and $\forall i,j,\ r_{i,j} \geq 0$.

Our problem boils down to the maximization of a convex function on a convex set. The maximum values are then obtained at some extremal vertices of the convex set. The extremal vertices of the convex set are such that $\sum_{i:c_j\in u_i} r_{i,j} = 1$, $r_{i,j} \in \{0,1\}$ and then $\sum_{j:c_j\in u_i} r_{i,j} \in \{0,1,2,3\}$. Maximizing our function is then equivalent to maximizing the number of $i$ such that $\sum_{j:c_j\in u_i} r_{i,j} = 3$.

This number is upper-bounded by $q$, and this maximum is reached if and only if there is an X3C-cover (in fact there is a point-to-point correspondence between the set of extremal vertices that reach $q$ and the set of X3C-covers).

If there exists an X3C-cover, then the backlog in the middle stage increases at rate $q$ (at rate 1 for each server that receive data at rate 3). If there is no X3C cover, then the backlog in the middle stage increases at most at rate $q-1$.

Finally, the reasoning above is valid for the interval of time [0,1]. Indeed, the backlog is sub-additive: if an arrival cumulative function $F_1$ defined on the interval $[0,s]$ in a server creates a backlog $b_1$ at time $s$, and if the arrival cumulative function $F_2$ such that $F_2(s) = 0$ creates a backlog $b_2$ at time $t \geq s$ (when the server is empty at time $s$), then the process $F_1 + F_2$ creates a backlog $b \leq b_1 + b_2$ at time $t$. As a consequence, there is no advantage in changing the service rates $r_{i,j}$ during the interval of time $[0,1]$. At time 1, servers $C_j$ and $U_i$ serve all their backlog and the backlog in server $V$ is then at most $3s - 2q$. This maximum is reached if and only if there is an X3C cover. If there is no X3C-cover, then the backlog is at most $3s - 2q - 1$.

**NP-hardness of the delay.** We keep the same scheme of reduction. There is an additional flow crossing server $V$ only and we are now interested in computing the worst-case delay for a bit of data of that flow (one can take $\alpha'(t) = \rho t$ with $\rho < R - 3s$ and the maximum delay can be obtained for the first bit of data arriving in the system).

The worst-case scenario for that network is first to create a backlog in server $V$ and then consider every other server as a infinite capacity server. To create the backlog, we use the previous construction.

Backlog at a server of the upper stage can be created if data arrive at rate more than one.

Backlog at a server of the middle stage can be created if there is an arrival rate of $r > 2$ at a server of the middle stage. The backlog grows at rate $r - 2$. If the arrival rate is exactly 3, then the flows arrive at rate 1, blocking the flows at the upper level, so this is a scenario that builds the greatest backlog.

Suppose that exactly $k$ servers of the middle stage have an arrival rate of 3 between time 0 and $t_k$. Without loss of generality, the servers that create backlog are $U_1,\ldots,U_k$. The backlog created is $(N_k - 2k)t_k$, where $N_k = \#\{F_{i,\ell} \mid \exists j \leq k,\ c_i \in u_j \cap u_\ell\}$ and for the other flows, either they are idle, either data are transmitted, but no backlog is created at the middle stage, but some backlog can

be build at the upper stage at rate $\sum_{\ell\,:\,c_j\in u_\ell}(r_{\ell,j}-1)_+$ for each server $C_j$. One can upper-bound the backlog created by using the inequality $N_k \leq 3s-3(q-k)$ and lower bounding the remaining flows by $3(q-k)$, and consider them as idle (they will start transmitting data at time $t_k$). Note that if $k=q$, this is not an approximation.

As a consequence, at time $t \geq t_k$, the quantity of data that arrived at server $V$ between $t_k$(start of the backlogged period) and $t$ is at most

$$Q_k(t) = (3s-3q+k)t_k + 3s(\min(t,1)-t_k) + 3(q-k)(\max(t,1)-1).$$

the delay is then less than $t-t_k$ such that $Q_k(t) = R(t-t_k)$.

If $t \leq 1$, then $t-t_k = \frac{3(s-q)+k}{R-3s}t_k$ and $t-t_k$ in increasing with $t_k$. If $t \geq 1$, then $t-t_k = \frac{3(s-q-k)-2kt_k}{R-3(q-k)}$ and $t-t_k$ is decreasing in $t_k$. So to obtain a maximum delay, $t_k$ has to be chosen so that $t=1$. Then $t_k = \frac{R-3s}{R-3q+k}$ and $d_k = \frac{3(s-q)+k}{R-3q+k}$ is an upper bound of the delay. As $R > 3s$, $d_k$ increases with $k$. And $d_q$ can effectively be achieved for $k=q$ as there is no approximation in that specific case.

If there exists an X3C-cover, the maximum delay is $\frac{3s-2q}{R-2q}$, and if there is no X3C-cover, the maximum delay that can be achieved is $\frac{3s-2q-1}{R-2q-1}$. So the question whether a delay greater or equal to $\frac{3s-2q}{R-2q}$ can be achieved is NP-hard. □

# 4 The tandem scenario: a polynomial algorithm

We study here a special class of feed-forward networks: the *tandem networks*, *i.e.* networks $\mathcal{N}$ such that the induced digraph $\mathcal{G}(\mathcal{N})$ is a directed path with no shortcut. It implies that any flow follows a sequence of consecutive servers in the path. Such scenarios have been highlighted by [26, 28, 20].

For this class of networks, the worst-case computation boils down to solving a single LP instance with a polynomial number of variables and constraints, and thus with a polynomial complexity. Moreover, we show for each flow how to reconstruct a minimum end-to-end service curve which is optimal in some sense.

## 4.1 The algorithm for the tandem scenario

**Theorem 4.** *Let $\mathcal{N}$ be a tandem network with $n$ servers and $p$ flows. Then, given a flow $i$ (resp. a server $j$), there exists one LP instance with $\mathcal{O}(pn)$ variables and $\mathcal{O}(pn^2)$ constraints such that the optimum is the worst end-to-end delay for flow $i$ (resp. the worst backlog at server $j$).*

*Proof.* Without loss of generality, we can assume that $i=1$ and $\text{last}(i)=n$ (resp. $j=n$) since what happens after the last server of interest does not impact on the dynamics of the first part of the network (servers after the last server of interest do not appear in the LP formulations of the problem).

A direct application of Theorem 2 to tandem networks induces a single order on the $n+1$ variables $t_{\pi_j}$, for the paths $\pi_{j-1} = j \cdots n$ and $\pi_n = \emptyset$. This order is simply $t_{\pi_0} \leq t_{\pi_1} \leq \cdots \leq t_{\pi_n}$. So, computing the worst backlog at server $n$ boils down to a single LP instance, while computing the worst end-to-end delay introduces the variable $u$ and possibly $n$ LP instances depending on the location of $u$.

Set $f_1 = \text{first}(1) - 1$ and $e_1 = \text{last}(1) = n$. We now show that it is useless to consider several LP instances to compute the worst delay. There is no need to consider all the positions to insert $u$ within the order $t_{\pi_{f_1}} \leq \cdots \leq t_{\pi_{e_1}}$. We only need one LP instance and the constraints where $u$ must appear are: $t_{\pi_{f_1}} \leq u \leq t_{\pi_{e_1}}$, $F_1^{(0)}(u) > F_1^{(n)}(t_{\pi_{e_1}})$ and $F_1^{(0)}(u) - F_1^{(0)}(t_{\pi_{f_1}}) \leq \alpha_1(u - t_{\pi_{f_1}})$. The objective and the other constraints remain unchanged. Remark that the maximization of the objective function will lead to the equality $F_1^{(0)}(u) - F_1^{(0)}(t_{\pi_{f_1}}) = \alpha_1(u - t_{\pi_{f_1}})$.

To prove this, we proceed by contradiction. Consider a worst-case trajectory for a tandem network. It is obtained as a solution of our $n$ LP instances of the general method. If that worst-case trajectory is not obtained by the alternative single linear program, which means that $F_1^{(0)}(u) - F_1^{(0)}(t_{\pi_{f_1}}) < \alpha_1(u - t_{\pi_{f_1}})$, one can replace $F^{(0)}_{1|[t_{\pi_{f_1}},u]}$ by $\alpha_{1|[t_{\pi_{f_1}},u]}$. Doing this the trajectory remains valid for the system: flow constraints, non-decrease, arrival and strict service are still satisfied. As far as the starts of backlog period are concerned, the additional data that arrived in a server are transmitted at the beginning of the backlogged period of the next server (ensuring that the input cumulative function in that next server is not less than the original input cumulative function and then ensuring that the backlogged period will not end before the backlogged period of the original cumulative functions. Since $F_1^{(0)}(u) < \alpha_1(u - t_{\pi_{f_1}})$ and the cumulative function $F_1^{(e_1)}$ is non-decreasing, $t_{\pi_{e_1}}$ must increase, hence one can obtain a longer delay than in the original trajectory.

□ □

## 4.2 From delays to end-to-end service curves

Let $\mathcal{N}$ be a network and flow 1 be the flow of interest, for now we have investigated a way to compute the worst delay for fixed constraints $(\alpha_i)_{i\in\mathbb{F}}$ and $(\beta_j)_{j\in\mathbb{S}}$. One may want to measure how the global network acts upon flow 1, in particular whether some minimum end-to-end service curve can be guaranteed. Given $\beta \in \mathcal{F}$, we say that $\beta$ is an *end-to-end (simple) service curve* (or *left-over service curve* [26, 28]) if $F_i^{out} \geq \beta * F_i^{in}$. It is called an *universal* end-to-end service curve if $\beta$ is independent of $\alpha_1$ (*i.e.* $\beta$ remains an end-to-end service curve for any choice of $\alpha_1$). Precomputing such an universal curve can be useful to quickly compute a bound on end-to-end delays for flow 1 for several different curves $\alpha_1$ (thanks to the horizontal distance of Theorem 1). In the case of tandem networks, we now prove that one can compute an universal end-to-end service curve which is optimal in some sense.

**Theorem 5.** *Let $\mathcal{N}$ be a tandem network with $n$ servers and $p$ flows. Then one can compute an universal end-to-end service curve for the flow 1, which is the maximum of all universal end-to-end service curves.*

*Proof.* We prove that this service curve can effectively be computed, using the dual problem of an LP instance.

To compute an end-to-end service curve for flow 1, the idea is to send a burst of size $\sigma$ (with $\alpha_1(t) = \sigma$) and compute the worst-case delay for that flow thank to our linear program. Let $d(\sigma)$ be that maximal delay. Doing this for every $\sigma$, we compute the function $d : \sigma \mapsto d(\sigma)$, which is trivially non-decreasing. We will first show its pseudo-inverse $\beta : t \mapsto \inf\{\sigma \geq 0 \mid d(\sigma) \geq t\}$ is a service curve for flow 1 and then we will show how to compute $d$.

We make an induction on the number of servers and are interested in flow 1 that crosses every server (this assumption is only for the ease of the presentation, exactly the same can be done for any flow, the only change is that the initialization step is $\mathbf{H}(\text{first}(1))$ and server 1 has to be replaced by server first(1)). Our induction hypothesis is:

**H($n$)** Let $\{F_i^{(j)} \in \mathcal{F} \mid i \in \mathbb{F},\ j \in \mathbb{S},\ j \in i\}$ be a trajectory for a system with $n$ servers in tandem. For every $t \in \mathbb{R}_+$, there exists $t_0 \leq t$ such that there exists a trajectory $\{\tilde{F}_i^{(j)} \in \mathcal{F} \mid i \in \mathbb{F},\ j \in \mathbb{S},\ j \in i\}$ for the system such that

1. $\tilde{F}_1^{(0)}(s) = F_1^{(0)}(t)$ if $s \geq t_0$ and $\tilde{F}_1^{(0)}(s) = F_1^{(0)}(s)$ otherwise (burst arrival at $t_0$);
2. $\forall i$, $\forall j \in i$, $\tilde{F}_i^{(j)}(s) = F_i^{(0)}(s)$ if $s \leq t_0$ (infinite server before $t_0$);
3. Let $t_n = start_n(t)$, $\tilde{F}_1^{(n)}(s) = F_1^{(n)}(s)$ if $s \geq t_n$ and $t_n$ is still the beginning of the backlogged period of $t$ in server $n$ in the trajectory $\{\tilde{F}_i^{(j)}\}$.

**H(1)** is true: let $t \in \mathbb{R}_+$. Let $t_1 = start_1(t)$. Here, $t_0 = t_1$ and the trajectory $\tilde{F}_i^{(j)}$ with $\forall i$,

- $\tilde{F}_i^{(0)}(s) = F_i^{(0)}(t)$ if $s \geq t_1$ and $\tilde{F}_i^{(0)}(s) = F_i^{(0)}(s)$ otherwise;
- $\tilde{F}_i^{(1)}(s) = F_i^{(0)}(s)$ if $s \leq t_1$ and $\tilde{F}_i^{(1)}(s) = F_i^{(1)}(s)$ otherwise

is a trajectory for the system: before time $t_0$, the system behaves as an infinite server and then has the same behavior as in the original system. As the server is strict, the behavior in a backlogged period only depends on the trajectory during that period. Moreover, at time $t_0$, a burst arrives, so that a backlogged period begins, and as during that period and until time $t$, $\tilde{F}_i^{(0)}(s) = F_i^{(0)}(t) > F_i^{(1)}(t) \geq F_i^{(1)}(s) = \tilde{F}_i^{(1)}(s)$, then that backlogged period cannot end before time $t$, and as the server is strict, we have a trajectory for the system.

Suppose that $\mathbf{H}(n-1)$ holds. Consider a tandem network with $n$ servers and a trajectory $\{F_i^{(j)} \in \mathcal{F} \mid i \in \mathbb{F},\ j \in \{1, \ldots, n-1\},\ j \in i\}$ of that system.

Let $t \in \mathbb{R}_+$ and $t_n = start_n(t)$. Apply $\mathbf{H}(n-1)$ to the $n-1$ first servers and to $t_n$. Let $a = F_1^{(n)}(t) - F_1^{(n)}(t_n)$. We modify the trajectories, up to time $t_n$, $\tilde{F}_1^{(j)}$ in the following way: if $\tilde{F}_1^{(j)}(s) \geq F_1^{(n)}(t_n)$, then $\tilde{F}_1^{(j)}(s) := \tilde{F}_1^{(j)}(s) + a$, and remains unchanged otherwise. In other words, we add a burst of size $a$ at time $t_0$, and serve it as a burst when the data arrived at time $(\tilde{F}_1^{(0)})^{-1}(F_1^{(n)}(t_0))$ are served, in each server $S_1, \ldots, S_{n-1}$. It should be obvious that this is still a trajectory for the system.

We now deduce $\{\tilde{F}_i^{(n)}\}$ from the trajectory $\{\tilde{F}_i^{(j)} \in \mathcal{F} \mid i \in \mathbb{F},\ j \in \{1, \ldots, n-1\},\ j \in i\}\}$ for the $n-1$ first servers satisfying the three conditions. The first condition is already satisfied, as it only concerns $F_1^{(0)}$. The second condition can also be satisfied, as from the induction hypothesis and flow constraints we have $\tilde{F}_1^{(n)}(s) \leq \tilde{F}_1^{(n-1)}(s) = \tilde{F}_1^{(0)}(s)$, one can set $\tilde{F}_1^{(n)}(s) = \tilde{F}_1^{(0)}(s)$ if before time $t_0$ each server serves data as an infinite capacity server.

Now, consider the $n$-th server from time $t_n$. By construction, we have $F_1^{(n-1)}(t_n) = F_1^{(n)}(t_n) + a$, and $t_n = start_n(t)$. The third condition can then be satisfied by setting $\tilde{F}_1^{(n)}(s) = F_1^{(n)}(s)$, $\forall s \geq t$. Then $\mathbf{H}(n)$ holds.

The construction of $\tilde{F}_1^{(j)}$ is depicted on Figure 7 for one flow (for sake of simplicity).

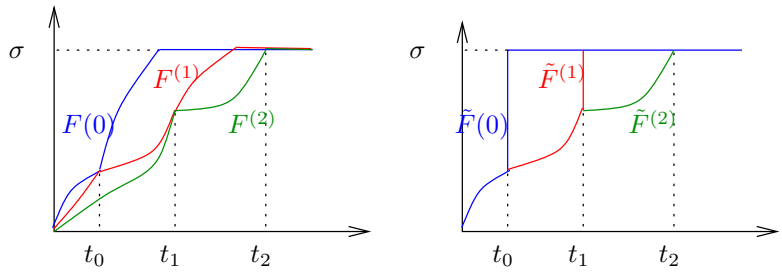


Figure 7: Construction of $\tilde{F}_1^{(j)}$.

Now look at the trajectory $\{\tilde{F}_i^{(j)}\}$. Until time $t_0$, servers act as infinite servers. Then, the departure function after time $t_0$ does not depend on the trajectory before time $t_0$. At time $t_0$ one has a burst of size $b = F_1^{(n)}(t) - F_1^{(0)}(t_0)$. So, the maximum delay for packets entering at time $t_0$ can be computed by our linear problem with $\alpha_1 : s \mapsto b$. So, as $\tilde{F}_1^{(n)}(t) = F_1^{(n)}(t)$ and $\tilde{F}_1^{(0)}(t_0) = F_1^{(0)}(t_0) + b$, we have $F_1^{(n)}(t) \geq F_1^{(0)}(t_0) + \beta(d^{-1}(b))$.

Note that this service curve is optimal in the sense that it is the maximum of all universal end-to-end service curves for flow 1. Indeed, if there existed an universal end-to-end service curve $\beta'$ for that flow and $\sigma \in \mathbb{R}_+$ such that $\beta^{-1}(\sigma) > \beta'^{-1}(\sigma)$, this would invalidate the fact that the delay bound computed with our linear program is tight when the arrival curve for flow 1 is $\alpha_1 : t \mapsto \sigma$.

Let us now go back to our linear problem when objective is to maximize the delay in a system where the arrival curve of our flow 1 is $\alpha_1 : t \mapsto \sigma$. It should be

clear that the only constraints where $\sigma$ appears are the arrival curve constraints for flow 1, that is $\forall f_1 \leq j' < j \leq e_1 = n$, $F_1^{(0)}(t_{\pi_j}) - F_1^{(0)}(t_{\pi_{j'}}) \leq \sigma$ and $F_1^{(0)}(u) - F_1^{(f_1)} \leq \sigma$. Moreover $\sigma$ does not appear in the objective constraint. So, one can express our optimization problem with matrices by

$$\begin{array}{c} \text{Maximizing } AX \\ \text{Subject to } BX \leq C(\sigma) \text{ and } X \geq 0, \end{array}$$

where only $C$ depends on $\sigma$. From the strong duality theorem (see for instance [34]), the following problem has the same solution:

$$\begin{array}{c} \text{Minimizing } C^t(\sigma)Y \\ \text{Subject to } B^tY \leq A^t \text{ and } Y \geq 0. \end{array}$$

Here, only the objective function depends on $\sigma$. The constraints $B^tY \leq A^t$ and $Y \geq 0$ define a convex polyhedron. For any linear objective function, the optimum value is obtained at an extremal vertex of the polyhedron. Thus, the delay can be computed in function of $\sigma$ by computing $\min_Y C^t(\sigma)Y$, where $Y$ belongs to the extremal vertices of the polyhedron. Note that this number of vertices is finite but can be exponential in the number of constraints.

□
□

Beware that although this curve $\beta$ is maximum among universal end-to-end service curves for flow 1, nothing ensures that for any arrival curve $\alpha_1$ for flow 1, the horizontal distance between $\alpha_1$ and $\beta$ will be the exact worst end-to-end delay (except for $\alpha_1(t) = \sigma$ where it is guaranteed by definition of $\beta$). This distance is an upper bound, but it could be loose. As a matter of fact, we conjecture that this distance is always tight, but we haven't proved this result yet.

The validity of this conjecture would be strongly related to blind multiplexing, since such universal and optimal end-to-end service curves do not necessarily exist for other policies. For instance, in FIFO networks, even for a single server with one cross-traffic flow, there is an infinity of incomparable simple service curves and their maximum is not a service curve [19, 20].

## 4.3 Related work

Initiated in [26], the study of tandem networks under blind multiplexing significantly improved in a recent breakthrough paper [28]. In this article the authors compute tight end-to-end delay bounds for some tandem networks, with detailed computations for a network with three servers and three flows and for sink-tree networks. For those particular cases, the authors manage to provide closed-form formulas (with disjunctions of cases). A method for general tandem networks is suggested in the corresponding technical report [27] but some details are not fully settled. We now discuss similarities and differences between our approaches.

Note that for the scenarios treated in [28], we have checked numerically on several examples that our algorithm using LP solutions gave the same results as their formulas (but we have not tried to solve by hand our LP instances to check whether we could reach the same disjunction of cases and closed-form formulas).

The main similarity concerns the choice of variables and set of constraints describing the system. We both start by writing down the system constraints over cumulative functions at starts of backlogged periods, leading to the same set of equalities and inequalities for tandem networks, up to some renaming: for instance if $n = 2$, $t_2, t_\emptyset$ here is $t_1, t_0$ there, and $\{F_2^{(1)}(t_2) - F_2^{(0)}(t_\emptyset) \leq \alpha_2(t_2 - t_\emptyset)\}$ here is $\{F_2^{(1)}(t_1) - F_2^{(0)} = \alpha_2(t_1 - t_0) - s_2^{(1)};\ s_2^{(1)} \geq 0\}$ there. It appears that those equalities and inequalities are linear (for leaky-bucket and rate-latency curves as noticed in [28], for piecewise affine concave/convex curves as noticed in the previous sections).

From that point, two major differences distinguish our approaches. First, while we directly try to solve this set of constraints using linear programming up to adding some constraints and considering several cases (due to the ordering of time events), the approach of [27, 28] for leaky-bucket/rate-latency arrival/service curves consists in:

- Performing algebraic manipulations (in particular using a lemma about the convolution of rate-latency functions) to mix the set of initial equalities and inequalities into an inequality looking like a simple end-to-end service curve constraint, that is for flow 1 in the 3-servers 3-flows scenario of [28]: $F_1^{(2)}(t_3) \geq F_1^{(0)}(t_0) + \beta_{R,T}(t_3 - t_0)$ where the rate $R$ is fixed, but the latency $T$ depends on free variables $s_2^{(1)}$ and $s_3^{(1)}$. Those parameters $R$ and $T$ do not depend on $\alpha_1$ the arrival curve of flow 1.

- This set of simple universal end-to-end service curves $\beta_{R,T}$ admits a maximum $\beta$ (since $R$ is fixed). It is achieved for a good choice of values $s_2^{(1)}$ and $s_3^{(1)}$ which optimize $T$, it can be computed by solving an LP instance.

- Then it is shown that for any arrival curve $\alpha_1$, the horizontal distance $h(\alpha_1, \beta)$ between $\alpha_1$ and $\beta$, which is an upper bound on the worst end-to-end delay, is actually tight, *i.e.* there exists a trajectory achieving this bound. Such a critical trajectory is fully described for sink-tree tandems, and in one case for the 3-servers 3-flows scenario (the other cases are left to the reader).

The second difference concerns the treatment of concave/convex arrival/service curves, if one already knows how to deal with leaky-bucket/rate-latency curves. Unlike the polynomial algorithm of Theorem 4, the scheme in [28] is to decompose the curves into maximum (resp. minimum) of rate-latency (resp. leaky-bucket) service (resp. arrival) curves, then compute the left-over service curves obtained for each combination of rate-latency/affine service/arrival curves, and finally compute the maximum of all those curves. One should notice that such

a process does not guarantee that this maximum of simple service curves is still a simple service curve or that it will provide tight bounds. It only ensures that the horizontal distance between the arrival curve of the flow of interest and this maximum curve is an upper bound on delays.

As pointed out in the article, the decomposition/recomposition scheme has a cost ($\prod_{i=1}^{p} |\alpha_i| \prod_{j=1}^{n} |\beta_j|$ combinations to consider).

Moreover there are some examples where this method does not give tight bounds : Take a system composed of two servers in tandem with respective service curve $\beta_1$ and $\beta_2$, crossed by two flows: the cross-traffic flow with arrival curve $\alpha$ and the flow of interest where one only needs to consider an arbitrarily small amount of data. We are interested in the delay needed for that data to cross the system. This small data, that can be called *infinitesimal data* (or *infinitesimal bit*), is sometimes used in the reasoning as an existing data (it moves and endures delays, it may have priority or not, its presence in a queue is considered as a backlogged period, ...) but which has a null measure (it adds 0 to cumulative functions for arrival or service constraints). This notion is very convenient to describe critical trajectories and provides right behaviors at the end. It is especially useful for concise explanations.

However to be very rigorous one must reason with small positive amounts of data like bursts of $\varepsilon > 0$ data, then perform the analysis and finally let $\varepsilon \to 0$ (otherwise with $\varepsilon = 0$ data directly, an application of the $D_{\max}$ definition for delays will give 0 whatever is the network, whereas we usually give arguments showing that the delays can be large).

For commodity we will describe the examples below with the convenient infinitesimal data of null measure. To get the rigorous version, replace it by an *infinitesimal data* $\varepsilon > 0$ which is a burst of $\varepsilon$ data at time $t = 0$, perform the computations and then let $\varepsilon \to 0$. For the first numerical example, the good news is that this limit is obtained by assuming that the flow of interest has a null arrival curve. For the second example with parameters, with the same critical trajectories, one should compute closed-form formulas including $\varepsilon$ and then let $\varepsilon \to 0$ to get the results that we give.

Now take for example $\beta_1(t) = 1.5(t-6)_+$, $\beta_2(t) = 6(t-8)_+$ and $\alpha(t) = \min(0.5t, 6 + 0.05t)$. Using the implementation of our solution, computations give that the worst delay is $d = 17.4$. If $\alpha(t) = 0.5t$, then the worst delay is $d_1 = 17.7$ and if $\alpha(t) = 6 + 0.05t$, then the worst delay is $d_2 = 18.4$ and $d < \min(d_1.d_2)$. Three critical trajectories are shown on Fig 8 and illustrate the loss when considering only $\alpha_1$ and $\alpha_2$.

Note that the difference between $d$ and $\min(d_1, d_2)$ can be arbitrary large. Here is another example with $\alpha = \min(\alpha_1, \alpha_2)$, and $\beta_1(t) = Rt$, $\beta_2(t) = 2R(t - T)_+$ and $\alpha_2 : t \mapsto RT$, we have:

- For $\alpha$, $d = 3/2T$. *Critical trajectory*: all data $\alpha(t)$ is served instantaneously at server 1 including our infinitesimal data which arrives at server 2 at time $t = 0$, then at server 2 priority is given to the cross-traffic flow and our data is served at the end of the backlogged period which lasts $3/2T$ (intersection of $\alpha$ and $\beta_2$).

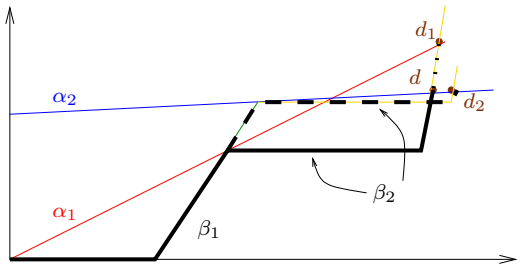


Figure 8: Decomposing arrival curves does not achieve tight bounds. Bold lines are the output processes. Continuous line: the arrival curve is $\alpha = \min(\alpha_1, \alpha_2)$; dotted line: the arrival curve is $\alpha_1$; dashed line: the arrival curve is $\alpha_2$.

- For $\alpha_1$, $d_1 = 2T$. *Critical trajectory*: same as the previous one but the backlogged period at server 2 lasts $2T$ (intersection of $\alpha_1$ and $\beta_2$).
- For $\alpha_2$, $d_2 = 2T$. *Critical trajectory*: $\alpha_2(t)$ bits arrive at server 1 including our infinitesimal data (that is a $RT$ burst at time $t = 0$), server 1 gives priority to the cross-traffic and the output is served instantaneously by server 2, then at time $t = T$ our infinitesimal data is finally served and enters server 2 where it starts a backlogged period which allows server 2 to add a delay $T$ to our infinitesimal data. It endures an overall delay $2T$.

The difference between $\min(d_1, d_2)$ and $d$ is then $T/2$ that grows infinitely large when $T$ grows (see Fig. 9).

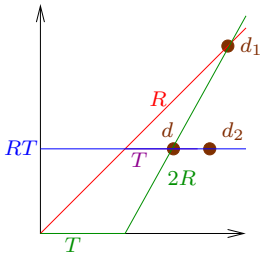


Figure 9: Arbitrarily large gaps between the decomposition/recomposition scheme and exact worst cases.

# 5 Numerical results

We compare our results with the other existing methods. Up to now, two kinds of methods have been used (see [28, 27] for detailed explanations): the *total flow analysis* (TFA), that consists in computing a delay upper bound for each server crossed by the flow of interest and then take the sum, and the *separate flow*

*analysis* (SFA), that consists in computing a left-over service curve for every server on the path of the flow of interest, then compute the convolution of those service curves to get a left-over service curve for the whole path and finally compute an end-to-end delay bound using Theorem 1. To our knowledge, these are the only two systematic methods available for general feed-forward networks.

The linear program files used for our experiments can be found following this link: `http://www.di.ens.fr/~bouillar/NCbounds/`.

Here are the formulas we use for TFA and SFA with leaky-bucket arrival curves and rate-latency strict service curves. Consider a server (Fig. 10) offering a strict service curve $\beta(t) = R(t-T)_+$ crossed by two flows 1 and 2 (it is often used with flow 1 being the flow of interest and flow 2 representing the aggregation of all the other flows crossing that server).

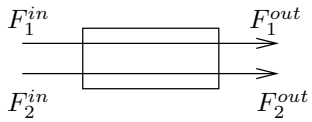


Figure 10: A server crossed by two flows.

If $F_1^{in}$ is upper-constrained by $\alpha_1(t) = \sigma_1 + \rho_1 t$ and $F_2^{in}$ is upper-constrained by $\alpha_2(t) = \sigma_2 + \rho_2 t$, then a simple service curve for $F_1$ is $\beta_1 = (\beta - \alpha_2)_+$.

An upper bound for the delay (used for TFA) for $F_1$ is

$$d_1 = T + \frac{\sigma_1 + \sigma_2 + \rho_2 T}{R - \rho_2}$$

The service curve (used for SFA) for $F_1$ is:

$$\beta_1(t) = (R - \rho_2)\left(t - T - \frac{\sigma_2 + \rho_2 T}{R - \rho_2}\right)_+.$$

Now, the cumulative function $F_1^{out}$ is upper constrained by

$$\alpha_1'(t) = \sigma_1 + \rho_1\left(T + \frac{\sigma_2 + \rho_2 T}{R - \rho_2}\right) + \rho_1 t.$$

Of course, since we are under blind multiplexing and by symmetry, those results apply to flow 2 up to inverting indices 1 and 2 in the formulas.

To get the bounds for the whole network, one can use those formulas for every node of the network sorted by a topological order.

In [8], it is proved that the worst-case delay in tandem networks is obtained for the SDF (*shortest-to-destination-first*) policy: flow have static priorities, the flow of interest has the lowest priority and flow $i_1$ has a higher priority that flow $i_2$ if $\text{last}(i_1) < \text{last}(u_2)$. SFA can be adapted using these priorities: for a given flow, no delay is induced by flows with lower priorities (computations are made as if those low priority flows did not exist).

## 5.1 Tandem scenario

In order to generate the linear program files associated to a tandem network to compute worst-case delay and backlog bounds, we wrote a program, that can be downloaded from the web-page mentioned above. This program has been written in Ocaml[4]. It generates a linear program from a small file describing the tandem network. The linear program can the be solved using solvers like lp_solve[5] for example.

We first compare our results for a tandem scenario, where flows intersect two servers (except at the extremities) and the flow of interest crosses every server. An example is depicted in Fig. 11. Servers have the same characteristics: they have a latency of 0.1s, and a service rate of 10Mbps. Flows have a maximum burst of 1Mb and a arrival rate of 0.67Mbps.

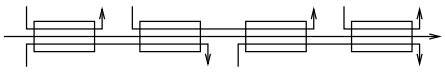

Figure 11: Tandem scenario with 4 servers.

Fig. 12 shows the delay bound obtained for each of the four methods (TFA, SFA, SDF and the tight LP method). Unsurprisingly, the three methods give the same result when there is only one server. For a network with 20 servers, the LP method reduces the SFA bound by a factor $8/5 = 1.6$, for an utilization rate of 20%.

Fig. 13 depicts the variation between SFA, SDF and LP methods when the utilization rate of the servers varies and when the number of servers is 20. Only the arrival rate varies, according to the utilization rate. When the utilization factor grows, the gain becomes huge. Moreover, the execution time of our program is less that 0.3 s. for a similar network with 50 servers.

## 5.2 Feed-forward scenario

We illustrate our result on a small example, depicted in Fig. 14. There are four servers (with the same characteristics as in the previous example, and four flows, each having the same characteristics: a maximum burst of 1Mb, and we make the arrival rate vary from 0.5Mbps to 4.5Mbps (so the utilization rate in each server vary from 10% to 90%).

We compute delay bounds for flow $F_1$ with four methods: (TFA), then (SFA), then (LP) by generating 11 linear programs, one for each possible order for our dates (as explained in Section 3), then the fourth method (ULP) is obtained by solving an unique linear program, where only the common constraints of the 11 programs are taken, *i.e.* constraints yielded by predicate (P1). In particular date are not totally ordered and the optimal solution that is found may not lead to non-decreasing cumulative functions. The ULP method does not obtain

[4] http://caml.inria.fr/ocaml

[5] http://lpsolve.sourceforge.net/5.5/

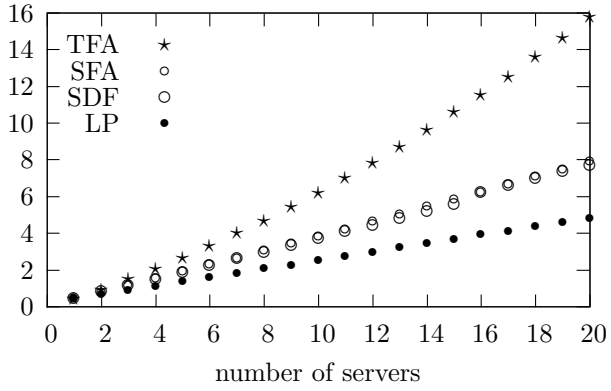


Figure 12: Upper bounds for the delay of the scenario of Fig. 11.

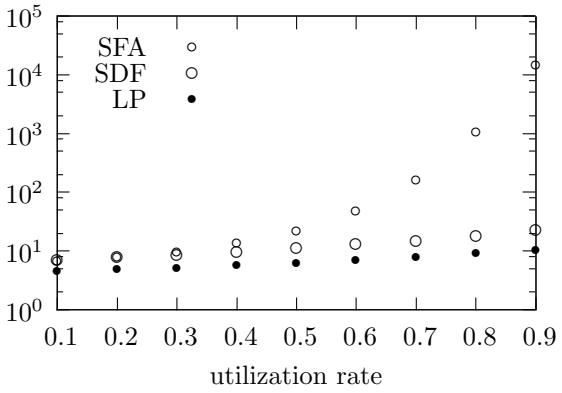


Figure 13: Upper bounds for the delay of the scenario of Fig. 11 for 20 servers and when the arrival rate varies.

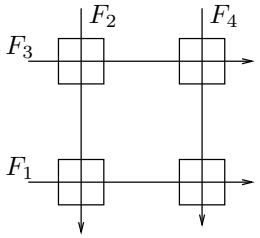


Figure 14: The Square network.

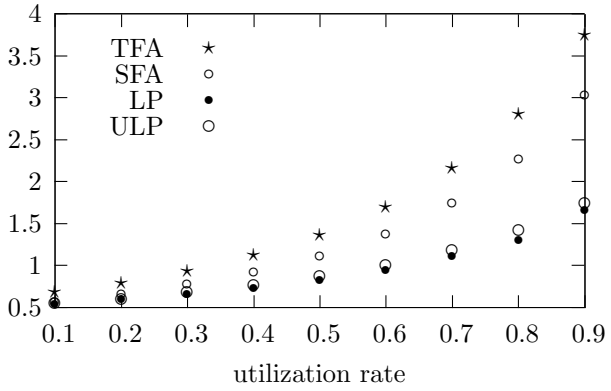


Figure 15: Upper bound on delays for flow $F_1$ in the Square network of Fig. 14.

the tight bounds, but the results are far better than TFA and SFA. Thus it is an interesting candidate as an approximation algorithm. All the results are depicted on Fig. 15.

# 6 Conclusion

We have shown that one can compute the exact worst local backlogs and end-to-end delays in the NC framework for feed-forward networks under blind multiplexing [6]. The number and the size of linear programs one has to solve can be small or extremely large depending on the network. Although we have shown that the problem is intrinsically difficult, one direction is to reduce this number of linear programs as well as their size. For example, our instances may present some redundant inequalities (*e.g.* between non-decrease and minimum service) or one could fix the value of $t_\emptyset$ (since the set of trajectories and thus the set of LP solutions is invariant by time shifts). These are very small improvements, but one may hope that there exist more significant reductions.

Another way to bypass NP-hardness is to look for fast approximation algorithms or exact algorithms which are fast on average.

Here are also a few features that can be added to refine the model:

- Add some other network elements which can not be modeled by strict service curves (like fixed delays).
- Take into account maximum (resp. minimum) strict service (resp. arrival) curves as in RTC [32] (preventing instantaneous propagation (resp. starvation) of data).
- Take into account packet lengths.

[6] The full implementation of our method for any feed-forward network as input is under development.

- Use curves with different shapes like ultimately pseudo-periodic curves [7].

Anyway, even without those additional features, the challenge of computing exact worst-case performances of general networks under blind multiplexing, or even feed-forward networks under other policies like FIFO, remains open.

# References


[1] M. Andrews. Instability of FIFO in the permanent sessions model at arbitrarily small network loads. In *Proc. of SODA'07*, 2007.

[2] F. Baccelli, G. Cohen, G.Y. Olsder, and J.P. Quadrat. *Synchronization and linearity*. Wiley, 1992.

[3] L. Bisti, L. Lenzini, E. Mingozzi, and G. Stea. Estimating the worst-case delay in FIFO tandems using network calculus. In *Proc. of Valuetools'2008*, 2008.

[4] A. Borodin, J. M. Kleinberg, P. Raghavan, M. Sudan, and D. P. Williamson. Adversarial queuing theory. *J. ACM*, 48(1):13–38, 2001.

[5] A. Bouillard, B. Gaujal, S. Lagrange, and E. Thierry. Optimal routing for end-to-end guarantees using network calculus. *Performance Evaluation*, 65(11-12):883–906, 2008.

[6] A. Bouillard, L. Jouhet, and E. Thierry. Tight performance bounds in the worst-case analysis of feed-forward networks. In *Proceedings of Infocom'2010*, 2010.

[7] A. Bouillard and E. Thierry. An algorithmic toolbox for network calculus. *Discrete Event Dynamic Systems*, 18(1):3–49, 2008.

[8] Anne Bouillard and Aurore Junier. Worst-case delay bounds with fixed priorities using network calculus,. In *Proceedings of Valuetools'11*, 2011.

[9] M. Boyer and C. Fraboul. Tightening end to end delay upper bound for AFDX network calculus with rate latency FCFS servers using network calculus. In *Proc. of WFCS'2008*, 2008.

[10] S. Chakraborty, S. Künzli, L. Thiele, A. Herkersdorf, and P. Sagmeister. Performance evaluation of network processor architectures: Combining simulation with analytical estimation. *Computer Networks*, 41(5):641–665, 2003.

[11] C.-S. Chang. A filtering theory for deterministic traffic regulation. In *Proc. of INFOCOM'97*, pages 436–443, 1997.

[12] C. S. Chang. *Performance Guarantees in Communication Networks*. TNCS, Springer-Verlag, 2000.

[13] R. L. Cruz. A calculus for network delay, part i: Network elements in isolation. *IEEE Transactions on Information Theory*, 37(1):114–131, 1991.

[14] R. L. Cruz. A calculus for network delay, part ii: Network analysis. *IEEE Transactions on Information Theory*, 37(1):132–141, 1991.

[15] M. Fidler. A network calculus approach to probabilistic quality of service analysis of fading channels. In *Proc. of GLOBECOM'2006*, 2006.

[16] V. Firoiu, J.-Y. Le Boudec, D. Towsley, and Zhi-Li Zhang. Theories and models for internet quality of service. *Proc. of the IEEE*, 90(9):1565–1591, 2002.

[17] M. Hendriks and M. Verhoef. Timed automata based analysis of embedded system architectures. In *Proc. of IPDPS*, 2006.

[18] Y. Jiang and Y. Liu. *Stochastic Network Calculus*. Springer, 2008.

[19] J.-Y. Le Boudec and P. Thiran. *Network Calculus: A Theory of Deterministic Queuing Systems for the Internet*, volume LNCS 2050. Springer-Verlag, 2001. revised version 4, May 10, 2004.

[20] L. Lenzini, E. Mingozzi, and G. Stea. A methodology for computing end-to-end delay bounds in FIFO-multiplexing tandems. *Performance Evaluation*, 65(11-12):922–943, 2008.

[21] S. Martin, P. Minet, and L. George. End-to-end response time with fixed priority scheduling: trajectory approach versus holistic approach. *Int. J. Communication Systems*, 18(1):37–56, 2005.

[22] S. Perathoner, E. Wandeler, L. Thiele, A. Hamann, S. Schliecker, R. Henia, R. Racu, R. Ernst, and M. G. Harbour. Influence of different system abstractions on the performance analysis of distributed real-time systems. In *Proc. of EMSOFT'2007*, 2007.

[23] G. Pruesse and F. Ruskey. Generating linear extensions fast. *SIAM Journal on Computing*, 23(2):373–386, 1994.

[24] G. Rizzo and J.-Y. Le Boudec. Stability and delay bounds in heterogeneous networks of aggregate schedulers. In *Proc. of INFOCOM'2008*, 2008.

[25] G. Rizzo and J.-Y. Le Boudec. "pay bursts only once" does not hold for non-FIFO guaranteed rate nodes. *Performance Evaluation*, 62(1-4):366–381, 2005.

[26] J. B. Schmitt and F. A. Zdarsky. The disco network calculator: a toolbox for worst case analysis. In *Proc. of Valuetools'2006*, 2006.

[27] J. B. Schmitt, F. A. Zdarsky, and M. Fidler. Delay bounds under arbitrary multiplexing. Technical report, University of Kaiserslautern, 2007.

[28] J. B. Schmitt, F. A. Zdarsky, and M. Fidler. Delay bounds under arbitrary multiplexing: When network calculus leaves you in the lurch ... In *Proc. of INFOCOM'2008*, 2008.

[29] T. Skeie, S. Johannessen, and O. Holmeide. Timeliness of real-time IP communication in switched industrial ethernet networks. *IEEE Tansactions on Industrial Informatics*, 2:25–39, 2006.

[30] L. Tassiulas and L. Georgiadis. Any work-conserving policy stabilizes the ring with spatial re-use. *ACM Transactions on Networking*, 4(2):205–208, 1996.

[31] L. Thiele, S. Chakraborty, M. Gries, A. Maxiaguine, and J. Greutert. Embedded software in network processors models and algorithms. In *Proc. of EMSOFT'2001*, 2001.

[32] L. Thiele, S. Chakraborty, and M. Naedele. Real-time calculus for scheduling hard real-time systems. In *Proc. of ISCAS'2000*, 2000.

[33] K. Tindell and J. Clark. Holistic schedulability analysis for distributed hard real-time systems. *Microprocess. Microprogram.*, 40(2-3):117–134, 1994.

[34] R. J. Vanderbei. *Linear Programming, Foundations and extensions*. Kluwer Academic Publisher, 2000.