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

Anne Bouillard, Laurent Jouhet, Eric Thierry

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

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

Anne Bouillard and Laurent Jouhet and Eric Thierry

N° 7012

Août 2009

Thème COM

ISSN 0249-6399 ISRN INRIA/RR--7012--FR+ENG

INRIA
RENNES

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

Anne Bouillard * and Laurent Jouhet † and Eric Thierry ‡

Thème COM — Systèmes communicants
Projet DistribCom

Rapport de recherche n° 7012 — Août 2009 — 22 pages

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 [27], 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 concave arrival curves and 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.

Key-words: Network Calculus, linear programming

* Anne.Bouillard@bretagne.ens-cachan.fr
† Laurent.Jouhet@ens-lyon.fr
‡ Eric.Thierry@ens-lyon.fr

Unité de recherche INRIA Rennes
IRISA, Campus universitaire de Beaulieu, 35042 Rennes Cedex (France)
Téléphone : +33 2 99 84 71 00 — Télécopie : +33 2 99 84 71 71

Calcul exact de performances pire cas dans des réseaux acycliques

Résumé : Le *Network Calculus* est une théorie visant à évaluer les performances pire-cas dans les réseaux de communication. Il fournit des méthode pour analyser des modèles où le trafic et les services sont contraints par des enveloppes inférieures et/ou supérieures (courbes d'arrivées et de service). Alors que de nouvelle applications émergent, beaucoup de problèmes importants restent ouverts : analyser de manière exacte des réseaux o des flux sont agrégés.

Cette théorie offre des méthodes efficaces pour obtenir des bornes supérieures sur les délais de bout-en-bout maximum et la charge locale maximum. Malheureusement, il a été montré dans un article récent [27] que ces bornes pouvaient tre arbitrairement mauvaises vis-à-vis de bornes effectivement atteintes, et ceci pour des réseaux très simples (deux serveurs et deux flux), lorsque on n'a aucune information *a priori* sur la politique de service entre les flux. Actuellement, seuls des réseaux avec trois flux et trois serveurs, ainsi que les réseaux *sink trees* ont été analysés de manière exacte.

Dans cet article, nous décrivons le premier algorithme qui calcule les bornes de performance exactes (pire délai de bout-en-bout pour un flux donné ou charge maximum pour un serveur donné) pour *tout* réseau acyclique quand on autorise toutes les politiques de service. La complexité de cet algorithme peut paraitre très élevée, mais nous montrons que ce problème est intrinsèquement difficile (NP-difficile). Heureusement, dans certains cas, le calcul devient polynomial, par exemple quand tout les trafics transverses d'un flux interfèrent avec ce flux sur un intervalle de serveurs. Nous comparons finalement notre méthode aux approches existentes.

Mots-clés : *Network Calculus*, programmation linéaire

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 informations about the system are 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 [14] to the analysis of wireless sensor networks [24], System-on-Chip [8], industrial Ethernets [28], critical embedded networks [7]. At the present time, the theory has developed and yield accomplished results which are mainly recorded in two reference books [10, 17]. It is an alternative to other approaches for worst-case performance analysis like holistic methods [32], trajectory methods [19] or model checking [15]. It is believed that Network Calculus and its extensions have advantages like modularity and scalability that will allow valuable analyzes of complex networks [20].

From their premises [9, 11, 12], 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. Then a general philosophy 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 fairly 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 [12], or unidirectional rings [29]. But, this condition is not sufficient for FIFO scheduling since there exists unstable networks at arbitrarily low utilization factors [1]. For general FIFO networks, the best sufficient conditions and associated bounds on delays are provided by [22] 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 [23]). This assumption is lighter 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) [25]. 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. It 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 [27] 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 a 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 concave arrival curves and 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 [27, 25], 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 [27]: 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.

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 [27, 26], 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 [30] and Sensor Calculus [24] (both use the strict service curves). But we do not address stochastic extensions like [13, 16].

2 Model and assumptions

2.1 Network Calculus framework

2.1.1 NC functions and systems

In Network Calculus, flows and services in the network are modeled by non-decreasing functions $t \mapsto f(t)$ where t is *time* and $f(t)$ an amount of *data*. There are different models depending on whether t or $f(t)$ takes discrete or continuous values, *e.g.* in $\mathbb{N}$ or $\mathbb{R}_+$. In this article, we will consider the *fluid model* where time in continuous and data can be divided into arbitrarily small pieces (time and data measures belong to $\mathbb{R}_+$). We will call *bit* the data unit of measure. Note also that we do not exactly use the term *fluid* as in [17] where the *fluid model* adds the condition that the manipulated functions are continuous. Nevertheless we make the usual assumption that our functions are *left-continuous*.

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). Cumulative functions will belong to $\mathcal{F}_\uparrow = \{f : \mathbb{R}_+ \to \mathbb{R}_+ \mid f \text{ non-decreasing, left-continuous, } f(0) = 0\}$. 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 assume that they all belong to $\mathcal{F}_\uparrow$ for commodity, but a careful look will show that all our solutions can be adjusted with no extra cost to deal with some ∞ values.

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

- 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 $\{(F^{in}, F^{out}) \in \mathcal{F}_\uparrow \times \mathcal{F}_\uparrow \mid F^{in} \ge F^{out}\}$. 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.

2.1.2 NC arrival curves

Given a data flow, let $F \in \mathcal{F}_\uparrow$ 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 $F(0) = 0$. A function $\alpha \in \mathcal{F}_\uparrow$ is an *arrival curve* for F if $\forall\ s, t \in \mathbb{R}_+$, $s \le t$, we have $F(t) - F(s) \le \alpha(t-s)$. 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). Then σ represents the maximal number of packets that can arrive simultaneously (the maximal burst) and ρ the maximal long-term rate of arrivals. A usual assumption is that α is sub-additive (otherwise it can be replaced by its sub-additive closure).

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 $start(t) = \sup\{u \le t | F^{in}(u) = F^{out}(u)\}$. Since F^{in} and F^{out} are left-continuous, we also have

$F^{in}(start(t)) = F^{out}(start(t))$. If $F^{in}(t) = F^{out}(t)$, then $start(t) = t$. 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 $\beta \in \mathcal{F}_\uparrow$, we define:

- $\mathcal{S}_{simple}(\beta) = \{(F^{in}, F^{out}) \in \mathcal{F}_\uparrow \times \mathcal{F}_\uparrow \mid F^{in} \geq F^{out} \text{ and } F^{out} \geq F^{in} * \beta\}$;
- $\mathcal{S}_{strict}(\beta) = \{(F^{in}, F^{out}) \in \mathcal{F}_\uparrow \times \mathcal{F}_\uparrow \mid F^{in} \geq F^{out}$, 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*) β 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 $\beta \in \mathcal{F}_\uparrow$, $\mathcal{S}_{strict}(\beta) \subseteq \mathcal{S}_{simple}(\beta)$ since for all $(F^{in}, F^{out}) \in \mathcal{S}_{strict}(\beta)$, we have for all $t \in \mathbb{R}_+$, $F^{out}(t) \geq F^{in}(start(t)) + \beta(t - start(t))$. But the converse is not true. An usual assumption for strict service curves is that β 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-case backlog and worst-case delay can be read easily 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 discipline for the flow) is

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

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

Remark that if we define $D'_{\max} = \sup\{t - s \mid 0 \leq s \leq t \text{ and } F^{in}(s) \geq F^{out}(t)\}$, we have $D'_{\max} \geq D_{\max}$. Suppose that $D'_{\max}$ is achieved for some arguments $0 \leq s_0 \leq t_0$. If F^{in} is increasing over $[s_0, s_0 + \varepsilon[$ or F^{out} is increasing over $]t_0 - \varepsilon, t_0]$ for some $\varepsilon > 0$, then $D'_{\max} = D_{\max}$. It is not true otherwise (*e.g.* if both functions are constant and equal around s_0 and t_0).

For the system S, the *worst-case 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 ([10, 17])**.** *Let S be an input/output system providing a simple service curve β and let (F^{in}, F^{out}) be a trajectory such that α is an arrival curve for F^{in}. Then,*

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

The worst-case backlog is bounded by the maximal vertical distance between α and β while the worst-case delay is given by the maximal horizontal distance between those two functions. Figure 1 illustrates this fact. Those bounds are tight if α is sub-additive and β a simple service curve (or β a strict super-additive service curve) [17].

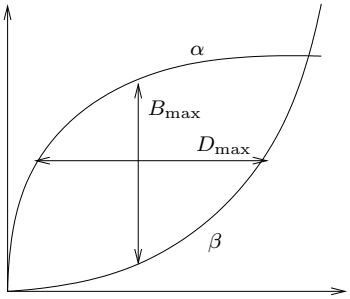

Figure 1: Guaranteed bounds on backlog and delay.

2.2 Network model

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

Servers and flows will be identified by indices. The set of servers is $\mathbb{S} = \{1, \dots, n\}$ and each server j offers a *strict* service curve $\beta_j \in \mathcal{F}_\uparrow$ 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 $\mathbb{F} = \{1, \dots, p\}$. Each flow i corresponds to a couple (α_i, μ_i) where μ_i is the (finite) ordered sequence of servers crossed by the flow and α_i is an arrival curve for the cumulative function before entering the first server. We suppose that $\alpha_i \in \mathcal{F}_\uparrow$ is piecewise affine (with a finite number $|\alpha_i|$ of pieces) and concave. 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 $first(i)$ (resp. $last(i)$) the index of the first (resp. last) server encountered by flow i. We will abusively write $j \in i$ to say that the server j belongs to the sequence μ_i. Given $j \in i$, we will denote $\mathrm{prec}_i(j)$ the index of the server preceding j in the sequence μ_i (by convention, $\mathrm{prec}_i(first(i)) = 0$).

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 μ_i. The sequences μ_i are paths in the digraph (the converse is not necessarily true). Any path on $\mathcal{G}(\mathcal{N}) = (\mathbb{S}, \mathbb{A})$ will be designated by its ordered sequence of vertices and often written as a *word over the alphabet* $\mathbb{S}$ (we will often use π to designate a path and for instance $j\pi$ will denote the path starting by vertex j followed by the path π).

In addition, we will use the following notations (similar to [27]): 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 $\{F_i^{(j)} \in \mathcal{F}_\uparrow \mid i \in \mathbb{F},\ j \in \mathbb{S},\ j \in i\}$ will be called a *trajectory of the network* $\mathcal{N}$ if it respects the NC constraints of the network:

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

(T2) $\forall i \in \mathbb{F},\ F_i^{(0)}$ is α_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. Given a flow i_0, we wish to compute the worst end-to-end delay endured by data of this flow, that is

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

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

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

Those are supremum over infinite sets, nevertheless they can be computed as shown next. Note that those supremum are not necessarily reached for a fixed trajectory or fixed 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). That is why in the paper we will sometimes place ourselves in a state where the worst case *almost* occurs, meaning that we work near the supremum up to a constant which can be chosen arbitrarily small. 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 Λ of linear programs (LP) with respective optimum values opt_λ, $\lambda \in \Lambda$, 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 Π is the set of paths ending at $end(i)$ (resp. j). Cardinalities are bounded by: $|\Pi| \leq 2^{n-1}$ and $|\Lambda| \leq |\Pi|!$.*

The description of the different LP instances and the proof of the theorem will be illustrated with the small but typical example of Fig. 2.

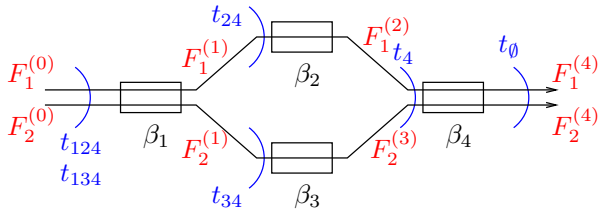

Figure 2: An example of network with two flows and four servers.

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 Π be the set of all paths ending at $end(i)$ (resp j). Here are the variables that will appear in each LP instance:

- t_π for all $\pi \in \Pi$, as well as a spare variable $t_\emptyset$. 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 of t_π 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_π.
- $F_i^{(0)}(t_\pi)$ 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_π.

Example

- The temporal variables are $t_\emptyset$, t_4, t_{24}, t_{34}, t_{124}, t_{134}.
- For flow 1 and server 1, the 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, the 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 each $F_i^{(j)}(t_\pi)$ is a numerical value, *i.e.* just an expression identifying a value, and not some kind of functional variable depending on the t_π values. 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 assignement 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 $t_{\pi_1} = t_{\pi_2}$, $t_{\pi_1} \leq t_{\pi_2}$ or $t_{\pi_1} < t_{\pi_2}$ 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_π values with their interpretation as starts of backlogged periods, we introduce two predicates over the t_π variables:

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

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

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', an ordering $start(t) < start(t') \leq start(t)$ can not occur, leading to predicate (P2).

We say that a set of temporal constraints $\mathcal{T}$ over Π' satisfies the predicates (P1) and (P2) if any solution to $\mathcal{T}$ with real values satisfies both (P1) and (P2). 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_π, $\pi \in \Pi'$, 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).

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 Π' 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 π and π' which is either $=, <, >, \leq$ or $\geq$. This comparison is denoted $\mathcal{T}(\pi, \pi')$. The set of all total orders over Π' 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 [21], 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, it has roughly a $\mathcal{O}(|\mathrm{Tot}(\Pi')|3^{|\Pi'|}|\Pi'|^2)$ complexity. We have not looked for a faster algorithm, there is probably some ways to speed up this step, but it 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 $\mathcal{T}_{i_1}(\pi, \pi') = \mathcal{T}_{i_2}(\pi, \pi')$. It can be checked with a $\mathcal{O}(p|\Pi|^2)$ algorithm. 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 check 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 redondant 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 $\mathcal{T}^1 = \{t_{12} = t_{13} < t_2 = t_3\}$ 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 Π 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):

- $\mathcal{T}^1 = \{t_{124} \leq t_{24} < t_{134} \leq t_{34} \leq t_4 \leq t_\emptyset\}$;

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

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

Example For $\mathcal{T}^1 = \{t_{124} \leq t_{24} < t_{134} \leq t_{34} \leq t_4 \leq t_\emptyset\}$, flow constraints, non-decrease and starts of backlogged periods for flow 1, are depicted on Fig. 3.

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

Figure 3: Flow constraints, non-decrease and starts of backlogged periods for flow 1 and $\mathcal{T}^1$.

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 $(t_\emptyset - u)$, 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 $\mathrm{Tot}(\Pi_{i_0})$:

- *Entry date:* $F_{i_0}^{(0)}(u) > F_{i_0}^{last(i_0)}(t_\emptyset)$.
- *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 et non-decrease:* $t_\pi \leq u$, $F_{i_0}^{(0)}(t_\pi) \leq F_{i_0}^{(0)}(u)$, and let $t_{\pi'}$ be the next element of t_π in $\mathcal{T}_{i_0}$ (if any), $u \leq t_{\pi'}$, $F_{i_0}^{(0)}(u) \leq F_{i_0}^{(0)}(t_{\pi'})$.
  - *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 $F_{i_0}^{(0)}(t_{\pi'}) - F_{i_0}^{(0)}(u) \leq \alpha_{i_0}(t_{\pi'} - u)$.

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 $Start_0 = \{t_{first(i_0)\pi} \mid first(i_0)\pi \in \Pi_{i_0}\}$ the set of all starts of backlogged periods at server $first(i_0)$. 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 $\sum_{i \ni j_0} F_i^{(\mathrm{prec}_i(j_0))}(t_\emptyset) - \sum_{i \ni j_0} F_i^{(j_0)}(t_\emptyset)$. It does not introduce new cases or new linear constraints.

3.2 From network trajectories to LP solutions

Let λ be a LP instance with optimal value opt_λ, we call a *solution* of λ an assignation of the variables satisfying the linear constraints, and it is *optimal* if it achieves opt_λ (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 Λ 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 a LP instance $\lambda \in \Lambda$ admitting a solution such that $t_\emptyset - u = d$ (resp. $\sum_{i \ni j_0} F_i^{(prec_i(j_0))}(t_\emptyset) - \sum_{i \ni j_0} F_i^{(j_0)}(t_\emptyset) = b$).*

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). Consider the interpretation of the variable t_π given in Section 3.1.1 and read their values on 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. For the delay evaluation, set the value u as $\inf\{t \mid t \leq t_\emptyset,\ F_{i_0}^{(0)}(t) > F_{i_0}^{(last(i)}(t_\emptyset)\}$. Then for all i, j, t_π, 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 t_π (and u) and the fact that the trajectory satisfies (T1), (T2), (T3) ensures that all the linear constraints in the LP instance λ are satisfied. Moreover by definition of the value $t_\emptyset$ (and u), we have $t_\emptyset - u = d$ (resp. $\sum_{i \ni j_0} F_i^{(\mathrm{prec}_i(j_0))}(t_\emptyset) - \sum_{i \ni j_0} F_i^{(j_0)}(t_\emptyset) = b$). □

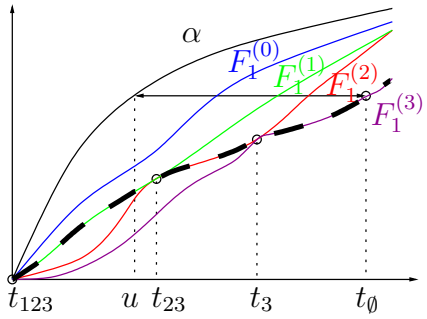

Figure 4: Reading t_π on a trajectory for a 1-flow 3-servers scenario.

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 Λ 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 $d = t_\emptyset - u$ ($b = \sum_{i \ni j_0} F_i^{(prec_i(j_0))}(t_\emptyset) - \sum_{i \ni j_0} F_i^{(j_0)}(t_\emptyset)$).*

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 For the worst backlog, all our LP instances admit at least a solution by choosing a set of values t_π complying to time constraints and all other values $F_i^{(j)}(t_\pi)$ being null. For the worst end-to-end delay, solutions exist if and only if α_{i_0} is not the null function. If $\alpha_{i_0} \neq 0$, since it is concave, $\alpha(u) > 0$ if $u > 0$. Thus choose some value t_π and u complying to time constraints and such that $u > 0$, impose that all other values $F_i^{(j)}(t_\pi)$ are null except $F_{i_0}^{(first(i_0))}(t_\pi) = \alpha(t_\pi)$. It ensures that $F_{i_0}^{(0)}(u) > F_{i_0}^{last(i_0)}(t_\emptyset)$.

The $<$ issue Our LP instance λ may use some strict linear inequalities. However LP solvers work with non-strict inequalities, and thus the LP instance which is actually solved is the instance $\hat{\lambda}$ where all signs $<$ have been replaced by $\leq$. Since the set of solutions of $\hat{\lambda}$ is the closure of the non-empty convex set of solutions of λ, we have $opt_{\hat{\lambda}} = opt_\lambda$ and for any solution of $\hat{\lambda}$ there exists an arbitrarily close solution of λ. Consequently, if the LP solver does not provide a solution achieving opt_λ and satisfying λ, we will work with a sequence of solutions of λ whose objective functions tends to opt_λ. Thus, from now, we will discuss the construction of trajectories for solutions of λ.

The assignation issue for functions Since our LP instance λ uses some non-strict inequalities, the following case may occur in a solution to λ: 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 contraint, thus we still have a solution to λ. 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 λ 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 λ.

Given a solution to λ, 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 substract c_i from all the values $F_i^{(j)}(t_\pi)$. Once done for all flows, we still have a solution to λ, 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 Let α (resp. β) be a concave (resp. convex) function in $\mathcal{F}_\uparrow$. Let $x_1 < x_2 < \cdots < x_k$ 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. $y_{\ell'} - y_\ell \geq \beta(x_{\ell'} - x_\ell)$). Then the function F from $[x_1, x_k]$ into $\mathbb{R}_+$ which is the linear interpolation of the points $(x_1, y_1)(x_2, y_2), \ldots, (x_k, y_k)$, 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. $F(t) - F(s) \geq \beta(t-s)$). The proof is a straightforward use of concavity (resp. convexity).

Reconstruction of cumulative functions for each flow Given a solution to λ, 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 μ_i. Suppose that $F_i^{\mathrm{prec}_i(j)}$ has been constructed and that there exists a path in Π starting at j. To construct $F_i^{(j)}$, we first have to define some intervals where some 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 Π. Then, for all the next j on μ_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 fonctions which belong to $\mathcal{F}_\uparrow$ satisfy (T1), (T2), (T3), thanks to the preceeding remarks et 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 $F_{i_0}^{(last(i_0))}(t) = F_{i_0}^{(0)}(t)$ 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 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 [17].

3.4 Computational hardness

Theorem 3. *Computing the worst-case backlog 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 $C = \{c_1, \ldots, c_{3q}\}$ of $3q$ elements and a collection $U = \{u_1, \ldots, u_s\}$ of s sets of 3 elements of C. The problem is to decide whether there exists a cover of S by q elements of C. We will reduce this problem to deciding whether a given backlog can be reached in a server of a network.

The network we use is as shown in Figure 5. The upper stage consists of $3q$ servers $S_1, \ldots, S_{3q}$, all with service curve $\beta_1 : t \mapsto t$. The middle stage consists of s servers $S'_1, \ldots, S'_s$, all with service curve $\beta_2 : t \mapsto 2.1t$ if $t < 1$; $t \mapsto \infty$ otherwise. Finally, the lower stage has only one server S'', 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 S_j, S'_i, S'' 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).

One wants to decide whether the backlog in S'' can be at least $3(s-q) + 0.9s$.

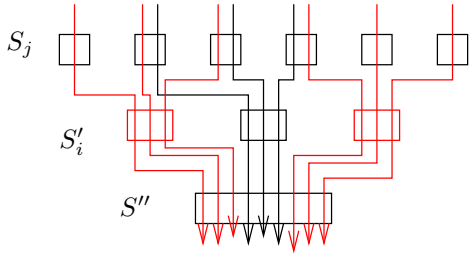

Figure 5: Transformation of an instance of X3C into a network.

The worst-case backlog in server S'' will be obtained when flows arrive according to α (indeed, no burst can never arrive), and when servers S_j and S'_i are exact servers.

Consider an infinitesimal time interval, during which each flow $F_{i,j}$ at the upper level is served at rate $r_{i,j}$, with $\sum_{i:c_j \in u_i} r_{i,j} = 1$. The service rate if S'_i is $\min(2.1, \sum_{j:c_j \in u_i} r_{i,j})$ and then the increase rate of the backlog during that time interval at the middle level is $\sum_{i \in \{1,\dots s\}} (\sum_{j:c_j \in u_i} r_{i,j} - 2.1)_+$. At the upper level, the backlog is created at rate $3s - \sum_{i,j} r_{ij} = 3(s-q)$, which does not depend on the service rates. Maximizing the backlog is equivalent to maximizing the following function:

$$\sum_{i \in \{1,\dots s\}} \Big[\sum_{j:c_j \in u_i} r_{i,j} - 2.1 \Big]_+$$

with the constraints: $\forall j \in \{1, \dots, 3q\}$, $\sum_{i:c_j \in u_i} r_{i,j} = 1$ and $\forall i, j$, $r_{i,j} \geq 0$. Note that one can replace $\sum_{i:c_j \in u_i} r_{i,j} = 1$ by $\sum_{i:c_j \in u_i} r_{i,j} \leq 1$.

Now, discard the constraints of the network, and suppose that $\sum_{i:c_j \in u_i} r_{i,j} \leq 1$ and $r_{i,j} \geq 0$ are the only constraints (U is the set of *all* 3-sets of C). Our problem boils down to the maximization of a convex function on a convex set. The maximum values are then obtained at some vertices of the convex set. The vertices of the convex et 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 $\sum_{j:c_j \in u_i} r_{i,j} = 3$.

Note that any small variation will make the function decrease (for $\epsilon \leq 0.9$, $0 + (3 - 2.1) = 0.9 > (\epsilon - 2.1)_+ + (3 - \epsilon - 2.1)_+ = 0.9 - \epsilon$). Then, the function reaches its maximum only at vertices that are in one-to-one correspondance with the X3C covers when U is the set af all 3-sets of C.

So, if one takes into account the constraints of the network (we get back to any collection U), the backlog is created in the middle level at rate at most 0.9. This maximum is reached if and only if there exists an X3C cover.

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 S_j and S'_i serve all their backlog and the backlog in server S'' is then at most $3(s - q) + 0.9s$. This maximum is reached if and only if there is an X3C cover.

□

The NP-hardness has been proved only for the worst-case backlog. A similar result should hold for the worst-case delay, using the same example, but this should be a little more tricky, as the arrival of the bit of data that reached (or almost reached) the worst-case delay may not arrive at time 1. But we conjecture that this problem is also 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 [25, 27, 18].

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 $\mathcal{O}(np)$ 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. Let us apply Theorem 2 to the tandem network. The induced graph $\mathcal{G}(\mathcal{N})$ is $(\mathbb{S}, \mathbb{A})$ with $\mathbb{S} = \{1, \dots, n\}$ and $\mathbb{A} = \{(j, j+1) \mid j \in \{1, \dots, n-1\}\}$.

The suffix-paths we need to consider are paths $\pi_{j-1} = \langle j, \dots, n\rangle$ and the empty path π_n. As a consequence our problem only has $n+2$ temporal variables t_{π_j}, $j \in \{0, \dots, n\}$ and u. Moreover, from (P1) one must have the following constraints: $t_{\pi_{j-1}} \leq t_{\pi_j}$, $j \in \{1, \dots, n\}$. As a consequence, there is only one possible order for the temporal variables, except for u. If one directly apply Theorem 2, u must be ordered with the t_{π_j}, which leads to at most n LP instances to compute the worst-case delay and one LP instance to compute the worst-case backlog in a server (there is no variable u).

Set $f_1 = first(1) - 1$ and $e_1 = end(1)$.

We now show that it is useless to consider several LP instances to compute the worst-case delay, and the only constraints where u must appear are: $F_1^{(0)}(u) \geq F_1^{(n)}(t_{\pi_{e_1}})$ 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 $F_1^{(0)}(u) - F_1^{(0)}(t_{\pi_{f_1}}) = \alpha_1(u - t_{\pi_{f_1}})$.

We proceed by contradiction. Consider a worst-case trajectory for a tandem network. It is obtained by solving our few LP instances. 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 begin 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 $\beta \in \mathcal{F}_\uparrow$, we say that β is an *end-to-end (simple) service curve* (or *left-over service curve* [25, 27]) if $F_i^{out} \geq \beta * F_i^{in}$. It is called a *universal* end-to-end service curve if β is independent of α_1 (*i.e.* β remains an end-to-end service curve for any choice of α_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 α_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 σ (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 σ, we compute the function $d : \sigma \mapsto d(\sigma)$, which is trivially non-decreasing. We will first show that the 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 $\mathbf{H}(first(1))$ and server 1 has to be replaced by server $first(1)$). Our induction hypothesis is:

$\mathbf{H}(n)$ Let $\{F_i^{(j)} \in \mathcal{F}_\uparrow \mid i \in \mathbb{F},\ j \in \mathbb{S},\ j \in i\}$ be a trajectory for a system with n servers in tandem. For every $t \in \mathbb{R}_+$, there exists t_0 such that there exists a trajectory $\{\tilde{F}_i^{(j)} \in \mathcal{F}_\uparrow \mid i \in \mathbb{F},\ j \in \mathbb{S},\ j \in i\}$ 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;
2. $\forall i,\ \forall j \in i,\ \tilde{F}_i^{(j)}(s) = F_i^{(0)}(s)$ if $s \leq t_0$;
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 $\{\tilde{F}_i^{(j)}\}$.

$\mathbf{H(1)}$ 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 $\{F_i^{(j)} \in \mathcal{F}_\uparrow \mid i \in \mathbb{F},\ j \in \{1, \ldots, n-1\},\ j \in i\}$ 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 $\{\tilde{F}_i^{(j)} \in \mathcal{F}_\uparrow \mid i \in \mathbb{F},\ j \in \{1, \ldots, n-1\},\ j \in i\}\}$ 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 $\tilde{F}_1^{(n)}(s) = F_1^{(n)}(s)$. Then $\mathbf{H}(n)$ holds.

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 do 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. Indeed, if there existed a minimal service curve β' 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 σ appear are for the arrival curve constraints for flow 1 and are $F_1^{(0)}(t_{\pi_j}) - F_1^{(0)}(t_{\pi_{j'}}) \leq \sigma$, with $0 \leq j' < j \leq n$. Moreover σ 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 σ. From the strong duality theorem ([33]), the following problem has the same solution:

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

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

□

Beware that although this curve β is maximum among universal end-to-end service curves for flow 1, nothing ensures that for any arrival curve α_1 for flow 1, the horizontal distance between α_1 and β will be the exact worst end-to-end delay (except for $\alpha_1(t) = \sigma$ where it is guaranteed by definition of β). This distance is an upper bound, but it could be loose. However we conjecture that this distance is always tight. Indeed, from the LP instance used to compute the maximum delay in Theorem 4 with a general piecewise affine concave arrival curve α_1, the arrival cumulative function of flow 1 is exactly α_1 from time t_0 to time u, so the worst-case delay is the distance between α_1 and the departure cumulative function.

Note that such an optimal end-to-end service curve does not necessarily exist for other policies. For instance, in FIFO networks [17, 18], even for a single server, there is not a unique end-to-end service curve, but an infinity which are not comparable (and the maximum of those service curves is of course not a service curve).

4.3 Related work

The study of worst-case delays in tandem networks, in the NC framework and under blind multiplexing, was initiated by [25] and the first tight bounds in the presence of several flows were achieved in [27]. In this later paper, the authors provide closed-form formulas (with disjunctions of cases) for the worst end-to-end delay in two scenarios:

- a particular tandem network with three servers and three flows, with leaky-bucket arrival curves and rate-latency strict service curves;
- sink-tree tandem networks where all flows end at the last server of the path, with leaky-bucket arrival curves and rate-latency strict service curves.

They suggest an idea to deal with concave/convex arrival/service curves, by a decomposition and recomposition scheme. In [26], they suggest some ways to extend their ideas to any tandem network, but some details are not settled. We now discuss similarities and differences between our approaches.

First of all, on the scenarios treated in [27], we have checked numerically on many examples that our algorithm using LP solutions gave the same results as their formulas (but we have not tried to solve by hand the LP instances to check whether we could reach the same disjunction of cases and closed-form formulas).

Then 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 in [27], but it is also true for piecewise affine concave/convex curves as seen before).

From that point, our approaches differ. First while we directly try to solve this set of constraints using linear programming up to adding some constraints (*e.g.* about time events) and considering several cases, the approach of [26, 27] consists in:

- Performing algebraic manipulations (in particular using a lemma about the convolution of rate-latency functions) to mix the set of equalities and inequalities into an inequality looking like a simple end-to-end service curve constraint, that is for flow 1 in their 3-servers 3-flows example: $F_1^{(2)}(t_3) \geq F_1^{(0)}(t_0) + \beta_{R,T}(t_3 - t_0)$ 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 α_1 the arrival curve of flow 1.
- This set of simple universal end-to-end service curves $\beta_{R,T}$ admits a maximum β (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 they show that for any arrival curve α_1, the horizontal distance between α_1 and β, which is an upper bound on the worst end-to-end delay, is actually tight, *i.e.* there exists a trajectory achieving this bound. They fully describe such a critical trajectory for sink-tree tandems, and in one case for their 3-servers 3-flows example (it is claimed that the the constructions for the other cases are easier).

A way to extend this to any tandem network is suggested in [26]. It appeals to some arguments about priorities between flows which might require further details and it aims at computing an universal simple end-to-end service curve providing tight bounds but, as explained in the preceding subsection, the existence of such a curve is not straightforward (even for leaky-bucket and rate-latency functions).

The second point 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 [27] 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 the worst end-to-end delay.

As pointed out in the article, the decomposition/recomposition scheme is costly ($\prod_{i=1}^{p} |\alpha_i| \prod_{j=1}^{n} |\beta_j|$ 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 β_1 and β_2, crossed by two flows, one with arrival curve α and the other, the flow of interest consisting of only one infinitesimal bit (that can be modeled by the null arrival curve as our model is fluid). We are interested in the delay needed for that bit to cross the system.

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-case 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 6 and illustrate the loss when considering only d_1 and d_2.

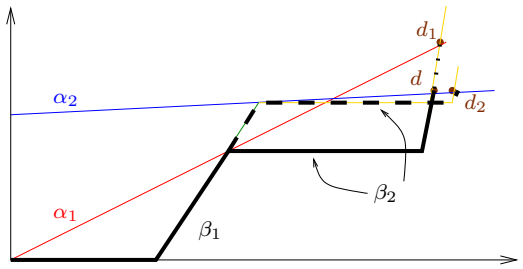

Figure 6: Decomposing arrival curves does not achieve tight bounds.Bold lines are the output processes. Continuous line: the arrival curve is α; dotted line: the arrival curve is α_1; dashed line: the arrival curve is α_2.

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 $\beta_1 = \alpha_1 : t \mapsto Rt$, $\beta_2 : t \mapsto 2R(t-T)_+$ and $\alpha_2 : t \mapsto RT$, we have $d = 3/2T$, $d_1 = d_2 = 2T$. the difference is then $T/2$ that grows infinitely large when T grows (see Fig. 7).

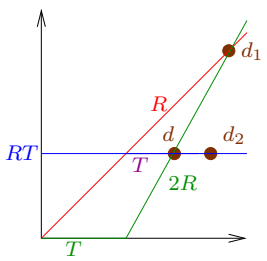

Figure 7: Decomposing service curves can be arbitrarily bad.

5 Numerical results

We here compare our results with the other existing methods. Up to now, two kinds of methods have been used (see [27, 26] for detailed explanations): the *total flow analysis* (TFA), that consists in computing the worst-case delay 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 and then compute the convolution of those service curves and the delay using that convolution and 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://perso.bretagne.ens-cachan.~bouillar/NCbounds/`.

Here are the formula we use for TFA and SFA when arrival curves are affine and service curves are rate-latency: consider a server (Fig. 8) offering a strict service curve $\beta : t \mapsto R(t-T)_+$ crossed by two flows 1 and 2 (in the following, if we are interested in flow 1, flow 2 represents the aggregation of every other flow crossing that server).

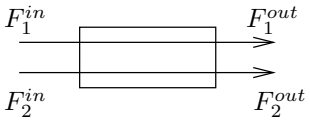

Figure 8: Server crossed by two flows.

If F_1^{in} is $\alpha_1 : t \mapsto \sigma_1 + \rho_1 t$ upper-constrained and F_2^{in} is $\alpha_2 : t \mapsto \sigma_2 + \rho_2 t$ upper-constrained, then a service curve for F_1 is $\beta_1 = (\beta - \alpha_2)_+$ and a service curve for F_2 is $\beta_2 = (\beta - \alpha_1)_+$.

Then, 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:

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

Now, the F_1^{out} (symmetric formula for F_2^{out}) is upper constrained by

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

To get the bounds it suffices to use those formulas on every node of the network sorted in a topological order.

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[1]. It generates a linear program from a small file describing the tandem network. The linear program can the be solved using lp_solve[2], 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. 9. 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.

Fig. 10 shows the delay obtained for each of the three methods (TFA, SFA 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, for an utilization rate of 20%.

Fig. 11 depicts the variation between SFA 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.

[1] http://caml.inria.fr/ocaml
[2] http://lpsolve.sourceforge.net/5.5/

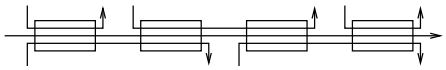

Figure 9: Tandem scenario with 4 servers.

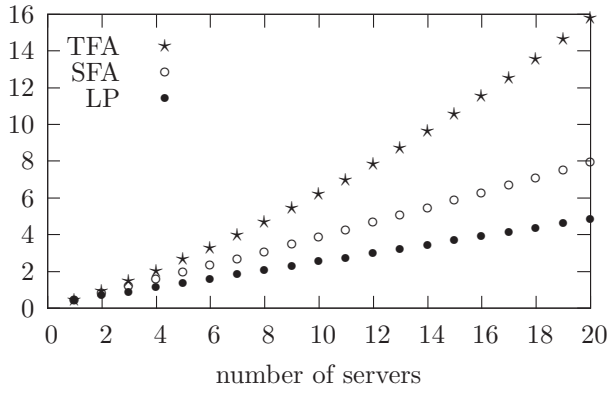

Figure 10: Upper bounds for the delay of the scenario of Fig. 9.

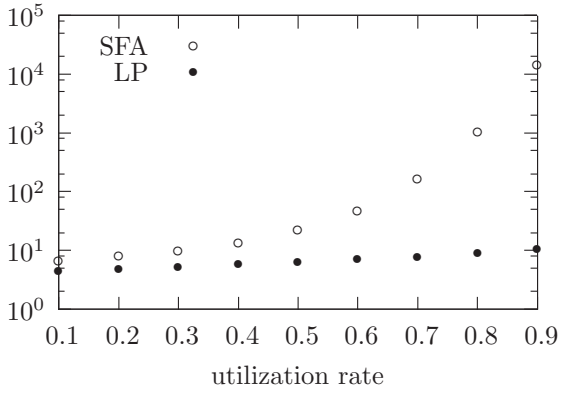

Figure 11: Upper bounds for the delay of the scenario of Fig. 9 for 20 servers and when the arrival rate varies.

5.2 Feed-forward scenario

We illustrate our result on a small example, depicted in Fig. 12. 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%).

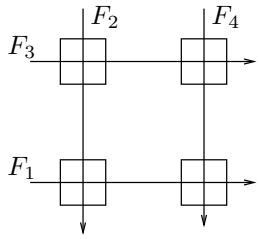

Figure 12: Feed-forward scenario.

We compute delay bounds for flow F_1 with four methods: TFA, SFA, by generating 11 linear programs, one for each possible order for the times (LP); the fourth method is obtained by solving a unique linear program, where only the common constraints of the 11 programs are taken (ULP). Doing this, we do not obtain tight bound, but the results is far better than TFA and SFA. Results are depicted in Fig. 13.

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. 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 inequations (*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 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:

- Mix the servers with strict service curves with some servers or links which only add a constant or bounded delay, as in [22]. It can be modeled by simple service curves but not by strict ones.
- Take into account maximum (resp. minimum) strict service (resp. arrival) curves as in RTC [31]. It prevents instantaneous propagation (resp. starvation) of data.

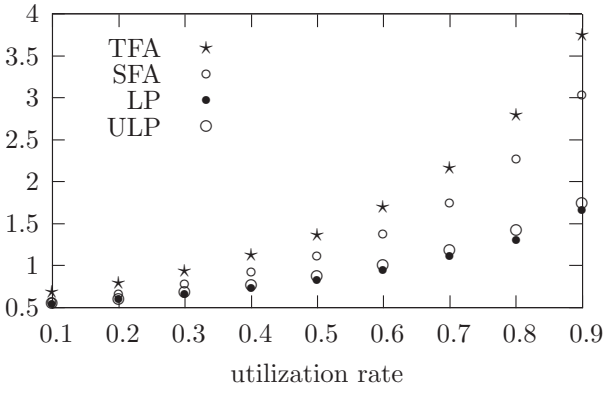

Figure 13: Upper bound for the worst-case delay of flow F_1 with the network of Fig. 12.

- Take into account packet lengths.
- Allow curves whose shape is different from convex/concave, *e.g.* ultimately pseudo-periodic curves [6].

Analyzing such generalizations of our models is of course at least as difficult, however one can focus on particular tractable classes. There is also a chance that some features could be expressed as linear constraints and added to our Linear Programming scheme (*e.g.* maximum strict service curves if they are piecewise affine and concave).

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

[7] 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.

[8] 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.

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

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

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

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

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

[14] 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.

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

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

[17] 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.

[18] 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.

[19] 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.

[20] 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.

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

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

[23] 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.

[24] J. B. Schmitt and U. Roedig. Sensor network calculus: A framework for worst case analysis. In *Proc. of 1st International Conference on Distributed Computing in Sensor Systems*, 2005.

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

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

[27] 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.

[28] 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.

[29] 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.

[30] 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.

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

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

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

Unité de recherche INRIA Rennes
IRISA, Campus universitaire de Beaulieu - 35042 Rennes Cedex (France)

Unité de recherche INRIA Futurs : Parc Club Orsay Université - ZAC des Vignes
4, rue Jacques Monod - 91893 ORSAY Cedex (France)
Unité de recherche INRIA Lorraine : LORIA, Technopôle de Nancy-Brabois - Campus scientifique
615, rue du Jardin Botanique - BP 101 - 54602 Villers-lès-Nancy Cedex (France)
Unité de recherche INRIA Rhône-Alpes : 655, avenue de l'Europe - 38334 Montbonnot Saint-Ismier (France)
Unité de recherche INRIA Rocquencourt : Domaine de Voluceau - Rocquencourt - BP 105 - 78153 Le Chesnay Cedex (France)
Unité de recherche INRIA Sophia Antipolis : 2004, route des Lucioles - BP 93 - 06902 Sophia Antipolis Cedex (France)

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt, BP 105 - 78153 Le Chesnay Cedex (France)
http://www.inria.fr
ISSN 0249-6399