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

Anne Bouillard, Laurent Jouhet, Éric Thierry

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

Anne Bouillard
ENS Cachan (Brittany) / IRISA
Campus de Ker Lann, 35 170 Bruz
Email: Anne.Bouillard@bretagne.ens-cachan.fr

Laurent Jouhet   Éric Thierry
ENS Lyon / LIP / IXXI, 46, allée d'Italie, 69 007 Lyon
Email: Laurent.Jouhet@ens-lyon.fr
Email: 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 (service/arrival 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 recently, 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.

I. 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 [1] to the analysis of System-on-Chip [2], industrial Ethernets [3], critical embedded networks [4]. At the present time, the theory has developed and yield accomplished results which are mainly recorded in two reference books [5], [6]. It is an alternative to other approaches for worst-case performance analysis like holistic methods [7], trajectory methods [8] or model checking [9]. It is believed that Network Calculus and its extensions have advantages like modularity and scalability that will allow valuable analyzes of complex networks [10].

From the beginning [11], [12], [13], 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 [14]. 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 if 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 [15]. A well-known necessary condition for stability is an utilization factor < 1 at each server. This condition is also sufficient for feed-forward networks [13], or unidirectional rings [16]. But, this condition is not sufficient for FIFO scheduling since there exists unstable networks at arbitrarily low utilization factors [17]. For general FIFO networks, the best sufficient conditions and associated bounds on delays are provided by [18] but they are usually not tight. More thrilling, for simple feed-forward networks like FIFO tandems, a recent paper [19] improving delay 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 [20]). 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) [21]. With a new method taking into account PMOO, 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 [22], thus preserving the NC spirit while being a good candidate for tight analysis of blind multiplexing. However a recent breakthrough paper [23] 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 [23], [21], 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 [23]: 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 II, we describe and analyze our algorithm in Section III where we also set the NP-hardness of the problem. Section IV shows how it applies to tandem networks and compares our solution to previous works namely [23], [24], while experiments are discussed in Section V to assess the gain w.r.t. to older NC methods. Further interesting extensions and open problems are presented in Section VI.

Our results are relevant to Network Calculus and extensions like Real-Time Calculus [25] and Sensor Calculus [26] (both use the strict service curves). But we do not address stochastic extensions like [27], [28].

Due to space constraints, all details and full proofs are postponed to [29].

II. Model and assumptions

A. Network Calculus framework

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 cumulative functions will belong to $\mathcal{F} = \{f : \mathbb{R}_+ \to \mathbb{R}_+ \mid f \text{ non-decreasing, left-continuous, } f(0) = 0\}$. 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 $\mathcal{F}$ 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.

Beyond usual operations like the minimum or the addition of functions, Network Calculus makes use of several classical $(\min, +)$ operations [14] 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. A *trajectory* of the system S is an element (F^{in}, F^{out}) of S.

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 $F(0) = 0$. A function $\alpha \in \mathcal{F}$ 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 $\alpha_{\sigma,\rho}(t) = \sigma + \rho t$, $\sigma, \rho \in \mathbb{R}_+$.

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 $start(t) = t$. Note that for any $t \in \mathbb{R}_+$, $]start(t), t[$ is a backlogged period.

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

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

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 $\max(a, 0)$. For all $\beta \in \mathcal{F}$, we have $\mathcal{S}_{strict}(\beta) \subseteq \mathcal{S}_{simple}(\beta)$ but the converse is not true.

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.

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

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 ([5], [6]): 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}.

B. 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 $\mathbb{S} = \{1, \dots, n\}$ 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 $\beta_j(t) = \max_{1\leq \ell \leq |\beta_j|}(r_{j,\ell}t - h_{j,\ell})$ 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 α_i is piecewise affine (with a finite number $|\alpha_i|$ of pieces) and concave. Thus it can be written $\alpha_i(t) = \min_{1\leq \ell \leq |\alpha_i|}(\sigma_{i,\ell} + \rho_{i,\ell}t)$ 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 by abuse of notation write $j \in i$ to say that the server j belongs to the sequence μ_i. Given $j \in i$, we will denote $\text{prec}_i(j)$ the index of the server preceding j in the sequence μ_i (by convention, $\text{prec}_i(first(i)) = 0$).

The overall network $\mathcal{N}$ is defined by $\mathbb{S}$, $\mathbb{F}$ and the sets $\{\beta_j,\ 1 \leq j \leq n\}$, $\{(\alpha_i, \mu_i),\ 1 \leq i \leq p\}$. 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 [23]): 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 satisfies the NC constraints of the network:

(T1) $\forall i \in \mathbb{F}, \forall j \in i,\ F_i^{(\text{prec}_i(j))} \geq F_i^{(j)}$;
(T2) $\forall i \in \mathbb{F},\ F_i^{(0)}$ is α_i-upper constrained;
(T3) $\forall j \in \mathbb{S},\ (\sum_{i\ni j} F_i^{(\text{prec}_i(j))}, \sum_{i\ni j} F_i^{(j)}) \in \mathcal{S}_{strict}(\beta_j)$.

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

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
$$\sup_{\{F_i^{(j)}\}\in\text{Traj}(\mathcal{N})}\ \sup_{0\leq s\leq t}\{t - s \mid F_{i_0}^{(0)}(s) > F_{i_0}^{(last(i_0))}(t)\}.$$
2) Given a server j_0, we wish to compute the worst backlog endured by this server, that is
$$\sup_{\{F_i^{(j)}\}\in\text{Traj}(\mathcal{N})}\ \sup_{t\geq 0}\{\sum_{i\ni j_0} F_i^{(\text{prec}_i(j_0))}(t) - \sum_{i\ni j_0} F_i^{(j_0)}(t)\}.$$

Those are supremum over infinite sets, nevertheless they can be computed as shown next. Note that those supremum are not necessarily reached for a fixed trajectory or fixed instants s, t.

III. 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). We have $|\Pi| \leq 2^{n-1}$ and $|\Lambda| \leq |\Pi|!2^{|\Pi|-1}$.

The description of the different LP instances and the proof of the theorem will be illustrated with the small but typical example of Fig. 1: the diamond network.

A. The LP instances

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

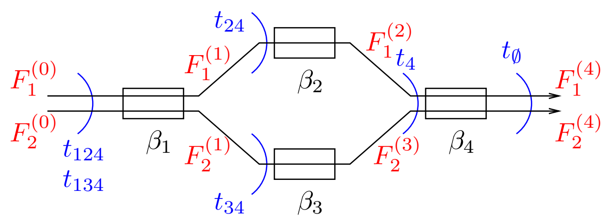

Fig. 1. Diamond network: two flows and four servers.

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 $\forall j\pi \in \Pi$, $t_{j\pi} = start_j(t_\pi)$, 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.

a) Diamond 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 $F_1^{(0)}(t_{124})$, $F_1^{(0)}(t_{24})$, $F_1^{(0)}(t_{134})$, $F_1^{(0)}(t_{34})$, $F_1^{(0)}(t_4)$, $F_1^{(0)}(t_\emptyset)$.

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 $t_{\pi_1} = t_{\pi_2}$, $t_{\pi_1} \le t_{\pi_2}$ 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) $\forall j\pi \in \Pi$, $t_{j\pi} \le t_\pi$.
(P2) $\forall j\pi_1, j\pi_2 \in \Pi$, $t_{j\pi_1} < t_{j\pi_2} \Rightarrow t_{j\pi_1} \le t_{\pi_1} < t_{j\pi_2} \le t_{\pi_2}$.

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

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 $\pi_1, \pi_2, \dots, \pi_N$ is a permutation of all the elements of Π' and for all $1 \le k \le N-1$, $\lhd_k \in \{=, <, \le\}$. 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 $=$,$<$,$>$,$\le$ or $\ge$. 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 [30], generate for each linear extension all the possible combinations of comparisons $=$, $<$ or $\le$ and for each one check whether it satisfies (P2).

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 $\Pi = \bigcup_{1\le i\le n} \Pi_i$. Let $(\mathcal{T}_1, \dots, \mathcal{T}_p) \in \mathrm{Tot}(\Pi_1) \times \cdots \times \mathrm{Tot}(\Pi_p)$, we say that the total orders $(\mathcal{T}_1, \dots, \mathcal{T}_p)$ are *mutually compatible* if for all $1 \le i_1, i_2 \le p$ and $\pi, \pi' \in \Pi_{i_1} \cap \Pi_{i_2}$, we have $\mathcal{T}_{i_1}(\pi, \pi') = \mathcal{T}_{i_2}(\pi, \pi')$. 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 $(\mathcal{T}_1, \dots, \mathcal{T}_p) \in \mathrm{Tot}(\Pi_1) \times \cdots \times \mathrm{Tot}(\Pi_p)$ 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 $\mathcal{T}^2 = \{t_{12} = t_{13} \le t_2 \le t_3\}$, only $\mathcal{T}^2$ needs to be considered).

Diamond 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:

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

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

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

3) Trajectory constraints: Let $(\mathcal{T}_1, \dots, \mathcal{T}_p) \in \mathrm{Tot}(\Pi_1) \times \cdots \times \mathrm{Tot}(\Pi_p)$ 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 $\{\sum_{i\ni j} F_i^{(j)}(t_\pi) - \sum_{i\ni j} F_i^{(j)}(t_{j\pi}) \ge \beta_j(t_\pi - t_{j\pi})\}$ (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 $\mathcal{T}(\pi_1, \pi_2) \in \{=, \le, <\}$, add $\{\sum_{i\ni j} F_i^{(j)}(t_{\pi_2}) - \sum_{i\ni j} F_i^{(j)}(t_{\pi_1}) \ge \beta_j(t_{\pi_2} - t_{\pi_1})\}$.
- *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 $\{F_i^{(0)}(t_{j\pi}) \ge F_i^{(j)}(t_{j\pi}),\ F_i^{(0)}(t_\pi) \ge F_i^{(j)}(t_\pi)\}$.

- *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 $\mathcal{T}(\pi_1, \pi_2) \in \{\leq, <\}$, 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 α_i is the minimum of affine functions).

Diamond example: For $\mathcal{T}^1 = \{t_{124} \leq t_{24} < t_{134} \leq t_{34} \leq t_4 \leq t_\emptyset\}$, flow constraints, non-decreasing constraints and starts of backlogged periods for flow 1, are depicted on Fig. 2.

$$\begin{array}{ccccccccccc}
F_1^0(t_{124}) & \leq & F_1^0(t_{24}) & \leq & F_1^0(t_{134}) & \leq & F_1^0(t_{34}) & \leq & F_1^0(t_4) & \leq & F_1^0(t_\emptyset) \\
\parallel & & \rotatebox{-90}{$\geq$} & & \parallel & & \rotatebox{-90}{$\geq$} & & \rotatebox{-90}{$\geq$} & & \rotatebox{-90}{$\geq$} \\
F_1^1(t_{124}) & \leq & F_1^1(t_{24}) & \leq & F_1^1(t_{134}) & \leq & F_1^1(t_{34}) & & & & \\
 & & \parallel & & & & & & & & \\
 & & F_1^2(t_{24}) & \leq & & & & & F_1^2(t_4) & & \\
 & & & & & & & & \parallel & & \\
 & & & & & & & & F_1^4(t_4) & \leq & F_1^4(t_\emptyset)
\end{array}$$

Fig. 2. 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:

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

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 $\mathrm{Tot}(\Pi_{i_0})$:

- *Arrival time:* $\{F_{i_0}^{(0)}(u) > F_{i_0}^{(last(i_0))}(t_\emptyset)\}$.
- *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 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_π 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 $t_{124} \leq u \leq t_{24}$, $t_{24} \leq u \leq t_{134}$, etc.

In some cases, one can reduce the number of positions to consider for u. It is explained in the particular case of Theorem 4.

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.

B. 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 III-A. 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. $\sum_{i \ni j_0} F_i^{(\mathrm{prec}_i(j_0))}(t_\emptyset) - \sum_{i \ni j_0} F_i^{(j_0)}(t_\emptyset) = b$).

Sketch of the 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). The interpretation of all the variables has been given in Section III-A1. One just has to find out their values from the trajectory. By comparing those values, one can pick a total order $\mathcal{T}$ fitting the time values and then retrieve an LP instance $\lambda \in \Lambda$ adjusted to the whole trajectory. As the trajectory satisfies (T1),(T2),(T3), it is easy to check that the linear constraints of λ are also satisfied. ■

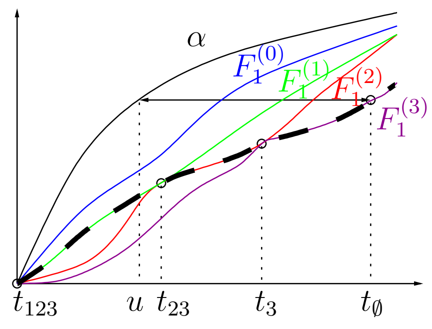

Fig. 3. Reading t_π on a trajectory for a 1-flow 3-servers scenario.

C. 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 III-A. Consider an instance $\lambda \in \Lambda$ and one of its solution (if any). 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 $d = t_{resp.\emptyset} - u$ (resp. $b = \sum_{i \ni j_0} F_i^{(\mathrm{prec}_i(j_0))}(t_\emptyset) - \sum_{i \ni j_0} F_i^{(j_0)}(t_\emptyset)$).

Sketch of the proof: Given a solution, the first step consists in shifting the $F_i^{(j)}(t_\pi)$ values such that every cumulative function will start from 0 at some non-negative instant. Then for each flow i, one will compute the functions $F_i^{(0)}, \ldots, F_i^{(last(i))}$ by induction, by moving forward along the path μ_i. The first function $F_i^{(0)}$ is the linear interpolation of the values $F_i^{(0)}(t_\pi)$ which are known. Then each step consists in constructing $F_i^{(j)}(t_\pi)$ from $F_i^{(\mathrm{prec}_i(j))}(t_\pi)$ by identifying

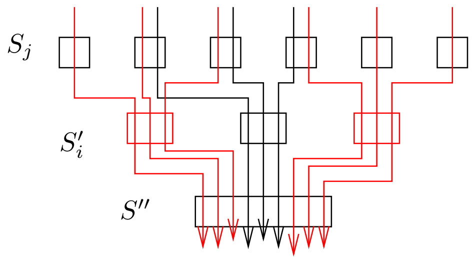

Fig. 4. Transformation of an instance of X3C into a network.

intervals where some data from flow i is backlogged at server j, performing linear interpolations on these intervals and elsewhere setting $F_i^{(j)}(t_\pi) = F_i^{(\text{prec}_i(j))}(t_\pi)$. It uses a lemma stating that the linear interpolation over a set of point preserves arrival/service constraints if the constraint curves are concave/convex.

■

We need another lemma to end the proof of Theorem 2 since LP solvers work with non-strict inequations. The proof works by exhibiting simple solutions to the LP instance (like all null $F_i^{(j)}(t_\pi)$ values).

Lemma 3: Let Λ be the set of LP instances constructed in Section III-A, $\lambda \in \Lambda$ and $\widehat{\lambda}$ obtained by replacing $<$ signs by $\leq$ in λ. Then $opt_{\widehat{\lambda}} = opt_\lambda$.

Note that the LP solvers will output a $+\infty$ result if it is actually the worst-case. As mentioned in the introduction, the classical criterion characterizing bounded worst-case performances for feed-forward networks remains an asymptotic utilization factor < 1 at each server preceding the server under observation, as proved *e.g.* in [6].

D. Computational hardness

Theorem 3: Computing the worst-case backlog in a feed-forward network is NP-hard.

Sketch of the 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 4. 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 $\alpha : t \mapsto \min(t, 1)$.

We show that the backlog in S'' can be at least $3(s-q)+0.9s$ if and only if there exists an X3C cover for the corresponding instance of the problem.

■

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 happen at time 1. But we conjecture that this problem is also NP-hard.

IV. 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 in [21], [23], [31].

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.

A. 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).

Sketch of the proof: A direct application of Theorem 2 to tandem networks induces a single order on the $n+1$ variables t_{π_j}, with $\pi_{j-1} = j \cdots n$ and $\pi_n = \emptyset$. So, computing a maximum backlog boils down to a single LP instance and only the insertion of u generates several (n) orders for the maximum delay.

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 $\alpha_{1|[t_{\pi_{f_1}},u]}$. Then the trajectory remains valid for the system: flow constraints, monotony, arrival and strict service constraints 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.

■

B. *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 β is an *end-to-end (simple) service curve* (or *left-over service curve* [21], [23]) 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). For 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.

Sketch of the proof: The proof consists in two steps:

- We first show that the function $\beta : t \mapsto \inf\{\sigma \geq 0 \mid d(\sigma) \geq t\}$ (where $d(\sigma)$ is the solution of the linear program with $\alpha_1(t) = \sigma$, $\forall t$) is a universal end-to-end service curve for flow 1.
- We then show that $\sigma \mapsto d(\sigma)$, the pseudo-inserse of β, can be computed as an infimum of a finite number of affine functions by applying the strong dual theorem to our linear problem where σ is a non-assigned parameter.

■

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 [6], [31], even for a single server, there is not an infinity of incomparable simple service curves and their maximum is not a service curve.

C. *Related work*

The study of tandem networks under blind multiplexing has already been addressed in [23]. 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. A method for general tandem networks is suggested in the corresponding technical report [24] but some details are not fully settled. Here we point out two major differences between our approaches.

We both start by a similar choice of variables and set of constraints. While we directly try to find a solution, they perform algebraic manipulations mixing the constraints to find some end-to-end simple service curves with free parameters. They choose the best one thanks to linear programming and then show that it enables to compute worst delays. Their result strongly relies on the fact that their arrival/service curves are leaky-bucket/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 [23] 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.

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.

With $\beta_1(t) = 1.5(t-6)_+$, $\beta_2(t) = 6(t-8)_+$ and $\alpha(t) = \min(0.5t, 6+0.05t)$ and 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 5 and illustrate the loss when considering only d_1 and d_2.

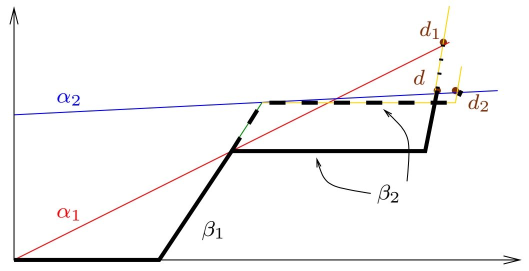

Fig. 5. Decomposing arrival curves does not achieve tight bounds.Bold lines are the output processes. Continuous line: the arrival curve is α; dotted line: the arrival curve is α_1; dashed line: the arrival curve is α_2.

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.

V. Numerical results

We here compare our results with the other existing methods. Up to now, two kinds of methods have been used (see [23], [24] 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.fr/~bouillar/NCbounds/`.

A. 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 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. 6. 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.

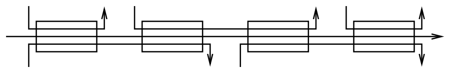

Fig. 6. Tandem scenario with 4 servers.

Fig. 7 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. 8 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. Moreover, the execution time of our program is less that 0.3 s. for a similar network with 50 servers.

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

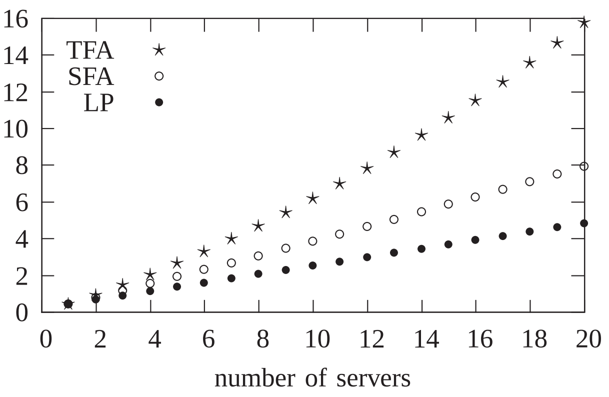

Fig. 7. Upper bounds for the delay of the scenario of Fig. 6.

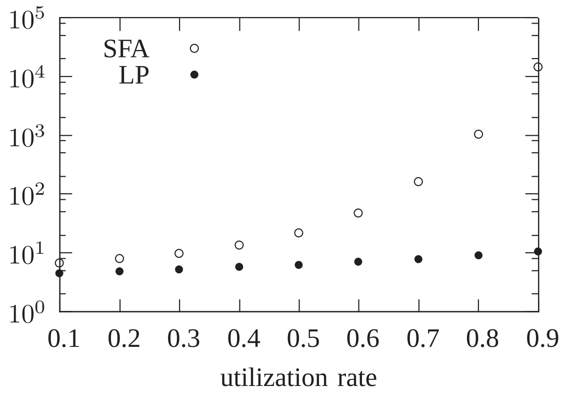

Fig. 8. Upper bounds for the delay of the scenario of Fig. 6 for 20 servers and when the arrival rate varies.

B. Feed-forward scenario

We illustrate our result on a small example, depicted in Fig. 9. 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%).

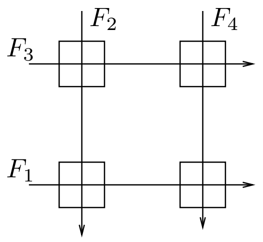

Fig. 9. The Square network.

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. 10.

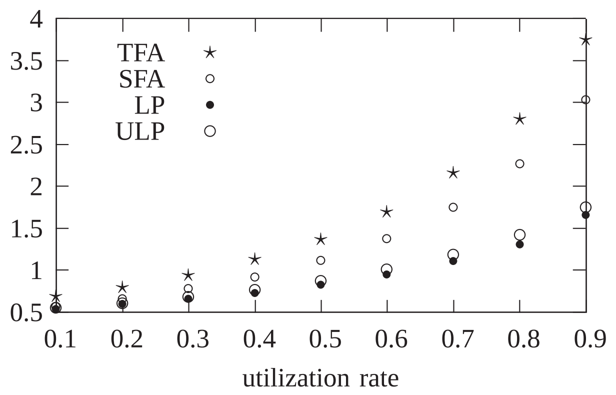

Fig. 10. Upper bounds on delays for flow F_1 in the Square network (Fig. 9).

VI. 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. 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 modelled 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, use curves with different shapes like ultimately pseudo-periodic curves [33].

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] V. Firoiu, J.-Y. Le Boudec, D. Towsley, and Z.-L. Zhang, "Theories and models for internet quality of service," *Proc. of the IEEE*, vol. 90, no. 9, pp. 1565–1591, 2002.

[2] S. Chakraborty, S. Künzli, L. Thiele, A. Herkersdorf, and P. Sagmeister, "Performance evaluation of network processor architectures: Combining simulation with analytical estimation," *Computer Networks*, vol. 41, no. 5, pp. 641–665, 2003.

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

[4] 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.

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

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

[7] K. Tindell and J. Clark, "Holistic schedulability analysis for distributed hard real-time systems," *Microprocess. Microprogram.*, vol. 40, no. 2-3, pp. 117–134, 1994.

[8] 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*, vol. 18, no. 1, pp. 37–56, 2005.

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

[10] 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.

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

[12] R. L. Cruz, "A calculus for network delay, part i: Network elements in isolation." *IEEE Transactions on Information Theory*, vol. 37, no. 1, pp. 114–131, 1991.

[13] ——, "A calculus for network delay, part ii: Network analysis." *IEEE Transactions on Information Theory*, vol. 37, no. 1, pp. 132–141, 1991.

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

[15] A. Borodin, J. M. Kleinberg, P. Raghavan, M. Sudan, and D. P. Williamson, "Adversarial queuing theory," *J. ACM*, vol. 48, no. 1, pp. 13–38, 2001.

[16] L. Tassiulas and L. Georgiadis, "Any work-conserving policy stabilizes the ring with spatial re-use," *ACM Transactions on Networking*, vol. 4, no. 2, pp. 205–208, 1996.

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

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

[19] L. Bisti, L. Lenzini, E. Mingozzi, and G. Stea, "Estimating the worst-case delay in fifo tandems using network calculus," in *Proc. of Value-tools'2008*, 2008.

[20] G. Rizzo and J.-Y. Le Boudec, ""pay bursts only once" does not hold for non-fifo guaranteed rate nodes," *Performance Evaluation*, vol. 62, no. 1-4, pp. 366–381, 2005.

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

[22] A. Bouillard, B. Gaujal, S. Lagrange, and E. Thierry, "Optimal routing for end-to-end guarantees using network calculus," *Performance Evaluation*, vol. 65, no. 11-12, pp. 883–906, 2008.

[23] 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.

[24] ——, "Delay bounds under arbitrary multiplexing," University of Kaiserslautern, Tech. Rep., 2007.

[25] 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.

[26] 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.

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

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

[29] A. Bouillard, L. Jouhet, and E. Thierry, "Tight performance bounds in the worst-case analysis of feed-forward networks," INRIA, Tech. Rep. 7012, August 2009.

[30] G. Pruesse and F. Ruskey, "Generating linear extensions fast," *SIAM Journal on Computing*, vol. 23, no. 2, pp. 373–386, 1994.

[31] L. Lenzini, E. Mingozzi, and G. Stea, "A methodology for computing end-to-end delay bounds in fifo-multiplexing tandems," *Performance Evaluation*, vol. 65, no. 11-12, pp. 922–943, 2008.

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

[33] A. Bouillard and E. Thierry, "An algorithmic toolbox for network calculus," *Discrete Event Dynamic Systems*, vol. 18, no. 1, pp. 3–49, 2008.