Worst-case delay bounds with fixed priorities using network calculus

Anne Bouillard
ENS / INRIA
45 rue d'Ulm
75005 Paris, France
anne.bouillard@ens.fr

Aurore Junier
INRIA / IRISA
Campus de Beaulieu
35000 Rennes, France
aurore.junier@inria.fr

ABSTRACT

Worst-case delay bounds are an important issue for applications with hard-time constraints. Network calculus is a useful theory to study worst-case performance bounds in networks. In this article, we focus on networks with a fixed priority service policy and provide methods to analyze systems where the traffic and the services are constrained by some minimum and/or maximum functions: arrival/service curves. Our approach uses linear programming to express constraints of network calculus.

Our first approach refines an existing method by taking into account fixed priorities. Then we improve that bound by mixing this method with other ones and provide a lower bound of the worst-case delay. Finally, numerical experiments are used to compare those bounds.

1. INTRODUCTION

Network calculus is a theory of deterministic queuing networks in communication networks. Based on the (min, plus) algebra, it aims to compute worst-case performance bounds, such as backlog or delay, for the analysis of critical systems. Applications of this theory can be found in the embedded networks field (the Avionic Full Duplex (AFDX), [3]) and Ethernet networks [5].

Network calculus uses functions (named curves) to described constraints on system. More precisely, *arrival curves* shape the incoming traffic by bounding the amount of data that can arrive during any interval of time and *service curves* give some guarantee about the minimal amount of data that is served. Using the algebraic properties of the (min,plus) operators ([6]), network calculus is an elegant theory that is modular and defines constraints on a system that permits to compute performance bounds.

However, recent studies have shown some limits of this theory: the direct application of the algebraic operators may lead to over-pessimistic bounds. In [10], the *pay multiplexing only once* (PMOO) phenomenon has been exhibited. There, the authors make a first attempt to compute tight delay bounds under arbitrary multiplexing (no assumption is made about the service policy) and focus on tandem networks. This problem has been tackled in [2] for feed-forward networks using linear programming (LP) techniques. The general problem (in feed-forward networks) is shown to be NP-hard, whereas in tandem networks, this problem is polynomial.

Other studies focused on some service policies. One can cite [7], where the authors investigate the First In Fist Out (FIFO) policy in tandem and sink-tree networks. The authors also exhibit some PMOO phenomenon. One can also cite [12], where the authors use RTC (real-time calculus), a variant of network calculus, to study networks with fixed priorities amongst other service policies. There, the authors use the network calculus operators as a basic block, but to get more precise results, they have upper and lower constraints for both the arrival curves and service curves.

was also used in [13] to find an upper bound of the worst case delay in Controller Area Network (CAN). However CAN networks have a restricted topology. Priorities have also been studied with other theories, for example the trajectory method in [9]. This study uses techniques that is not the purpose of this article.

In this article, we study networks with a fixed priority service policy (each flow is assigned a fixed priority) and try to take into account the PMOO phenomenon. A first result will show that not surprisingly, the stability issue in this case boils down to a per-node stability. Our main contribution is to improve the computation of the existing worst-case delay bounds. The first approach refines the approach of [2] in order to take into account fixed priorities. Then we improve that bound by mixing this method with other ones. Finally we provide a method that computes a lower bound of the worst-case delay in order to delimit the worst case-delay. This gives us an idea of the tightness of our results.

The rest of this paper is organized as follows: Section 2 presents the network calculus framework and describes the existing methods for worst-case upper bounds of the delay. Section 3 gives a first bound and the necessary and sufficient condition for stability, then Section 4 extends the results of [2] to the case of strict priorities. The (non)-tightness of our bounds and some improvements are discussed in Section 5 and numerical comparison of our method with the others existing ones is presented in Section 6 before concluding in Section 7.

VALUETOOLS 2011, May 16-20, Paris, France

DOI 10.4108/icst.valuetools.2011.245603

2. THE NETWORK CALCULUS FRAMEWORK

Network calculus is a theory that studies the relations between flows of data in a network. The movements of data are described by *cumulative functions* in $\mathcal{F} = \{f : \mathbb{R}_+ \to \mathbb{R}_+ \cup \{+\infty\} \mid f(0) = 0,\ f \text{ is non-decreasing and left-continuous}\}$. A cumulative function $f(t)$ counts the amount of data that arrives/departs from a network element up to time t.

A second type of functions that is used in network calculus are the *arrival and service curves.* These functions give some constraints to shape cumulative arrival in a server or to guaranty a minimal service of a server. These functions are used for computing worst-case delay bound.

Network calculus computes bounds by applying operators on the arrival and service curves. Those operators are based on the (min,plus) algebra and the main ones, beyond the point-wise minimum and addition, are the convolution and the deconvolution: 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: $f \oslash g(t) = \sup_{u \ge 0} f(t + u) - g(u)$.

2.1 Arrival and service curves

Given a data flow, let $F \in \mathcal{F}$ be its cumulative function at some point, such that $F(t)$ 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 (or F is α-constrained) 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 $\alpha_{\sigma,\rho}(t) = \sigma + \rho t$, $\sigma, \rho \in \mathbb{R}_+$.

Two types of minimum service curves are commonly considered: *simple service curves* and *strict service curves.* A trajectory is a family of functions if $\mathcal{F}$ and describes the amount of input or output packets of a system. We denote (F^{in}, F^{out}) the trajectories of an input/output system. Then, 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 $start(t) = \sup\{u \le t \mid F^{in}(u) = F^{out}(u)\}$. 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:

- $\mathcal{S}_{simple}(\beta) = \{(F^{in}, F^{out}) \in \mathcal{F} \times \mathcal{F} \mid F^{in} \ge F^{out}$ 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 contrary is not true.

In network calculus 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 (that is, one works with the sum of the cumulative functions). This is the case here.

2.2 Network composition, characteristics and bounds

Given an input/output system S, bounds for the worst-case delay can easily be read from the arrival and service curves. Let (F^{in}, F^{out}) be a trajectory of S. The delay endured by data entering at time t (assuming FIFO discipline for the flow) is

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

We denote the maximum horizontal distance between f and g by $hDev(f, g) = \inf\{t \ge 0 \mid f(t) \le g(t + d)\}$. The *worst-case delay* of a trajectory is $D_{\max} = \sup_{t \ge 0} d(t) = hDev(F^{in}, F^{out})$.

For the system S, the *worst-case delay* is the supremum over all its trajectories.

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

THEOREM 1 ([4, 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. *$D_{\max} \le hDev(\alpha, \beta)$.*
2. *$\alpha \oslash \beta$ is an arrival curve for F^{out}.*

EXAMPLE 1. *Given a flow f with arrival curve $\alpha(t) = \sigma + \rho t$ crossing a server that offers a minimal service curve $\beta(t) = R(t - T)_+$, one can compute this bound. If $R \ge \rho$,*

$$\begin{aligned} D_{\max} &\le hDev(\alpha, \beta) = T + \frac{\sigma}{R} \\ \alpha \oslash \beta(t) &= (\sigma + \rho T) + \rho t. \end{aligned}$$

If $R < \rho$, then $D_{\max} = \alpha \oslash \beta(t) = \infty$.

In order to compute bounds for more complex systems than a single server crossed by a single flow, one may use the following theorems.

THEOREM 2 (CONCATENATION, [4, 6]). *Consider two servers with respective service curves β_1 and β_2. The system composed of the concatenation of the two servers offers a minimal service curve $\beta_1 * \beta_2$.*

THEOREM 3 (RESIDUAL SERVICE CURVE, [6]). *Let f_1 and f_2 be two flows crossing a server that offers a strict service curve βsuch that f_1 is α_1-constrained. Then a minimal simple service curve offered to f_2 is $(\beta - \alpha_1)_+$.*

Note that the assumption that the server offers a strict service curve is mandatory, but the residual service curve is simple, preventing from formally applying the theorem to the residual service curve. The following theorem states that with fixed priorities, the residual service remains strict.

THEOREM 4 (RESIDUAL CURVE WITH PRIORITIES, [1]). *Let f_1 and f_2 be two flows crossing a server that offers a strict service curve β such that f_1 is α_1-constrained and has the priority over f_2. Then a minimal simple service curve offered to f_2 is $(\beta - \alpha_1)_+$.*

EXAMPLE 2. *Consider two flows f_1 and f_2 crossing a server such that $\alpha_1(t) = \sigma_1 + \rho_1 t$ is an arrival curve of f_1 and the server offers a strict service curve $\beta(t) = R(t-T)_+$. If $R \leq \rho_1$, a service curve offered to f_2 by the server is*

$$\beta'(t) = (R - \rho_1)\left(t - \frac{\sigma_1 + RT}{R - \rho_1}\right)_+.$$

2.3 State of the art: computing worst-case performance bounds using network calculus

Different methods have been derived to compute worst-case performance bounds. The two first methods do not make any assumption on the service policy and the third, Real-Time Calculus (RTC), assumes the existence of priorities. We describe here these methods, using the toy example of Figure 1. This network has three flows and two servers. Flows f_1 and f_3 cross the two servers but f_2 goes only through the first server. It has a service curve β_1, and the other β_2. We are interested in the delay suffered by f_3.

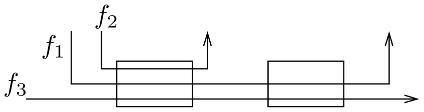

Figure 1: Network with three flows and two servers.

Separate Flow Analysis (SFA) [10].

Here, Theorems 1, 2 and 3 are used to compute the global residual service offered to a flow, and then compute the horizontal maximum distance with the arrival curve to get an upper delay bound. One needs to compute the arrival and residual service curves for each flow on each server of Figure 1. The delay suffered by f_3 is at most

$$d_{SFA} = hDev(\alpha_3, ([\beta_1 - \alpha_1 - \alpha_2]_+ * [\beta_2 - (\alpha_1 \oslash [\beta_1 - \alpha_2 - \alpha_3]_+)]_+).$$

Linear Programming (LP) [2].

Here, the network calculus constraints (service, arrival, causality...) for a network are encoded as linear programs (only one in the case of tandem networks), under some assumptions: piecewise affine concave arrival curves and piecewise affine convex service curves. The solution of the program gives the value of all cumulative functions at a finite number of dates (that are starts of backlogged periods). This method gives exact bounds under arbitrary multiplexing (no assumption is made about the service policy) for acyclic networks, in polynomial time for tandem networks, but is proved to be NP-hard in the general setting. Due to the space limit, we do not describe here the linear program for the network of Figure 1, but a similar method will be studied in Section 4.

Real Time Calculus - fixed priorities (RTC) [12].

This method is quite similar to the SFA one, but makes the assumption that flows have fixed priorities, which enables to tighten the bounds. The residual service curve is computed by progressively subtracting the resources needed by the flows with higher priorities. Here, in the aim of comparing with the other methods, we intentionally forget the maximum service curves and the minimal arrival curves.

We apply this analysis to the example of Figure 1 assuming that flows are ordered by priorities (*i.e.* f_1 has the highest priority).

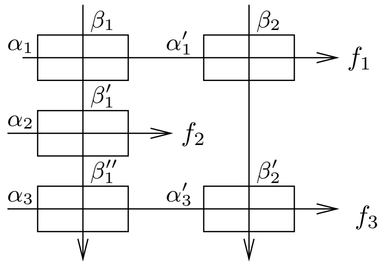

Figure 2: Application of RTC method to the example.

Figure 2 presents the network in the RTC manners. For example, flow f_1 crosses the first server, then we compute the residual service curve of this server β'_1 for f_2 and the arrival curve α'_1 of f_1 for the second server by direct application of Theorems 1 and 4. One gets $\beta'_1 = (\beta_1 - \alpha_1)_+$ and $\alpha'_1 = (\alpha_1 \oslash \beta_1)_+$. Finally,

$$d_{RTC} = hDev(\alpha_3, (\beta_1 - \alpha_1 - \alpha_2)_+ * [\beta_2 - \alpha_1 \oslash \beta_1]_+).$$

Note that with the SFA formula, one would get the same formula if one had assumed fixed priorities.

EXAMPLE 3. *With $\alpha_1(t) = 1+t$, $\alpha_2(t) = 1+t$, $\alpha_3(t) = 1+2t$, $\beta_1(t) = 6(t-1)_+$ and $\beta_2(t) = 4(t-2)$, the residual curves for flow 3 are $\beta_{SFA}(t) = 3(t-6.556)_+$ ($d_{SFA} = 6.889$), $\beta_{LP}(t) = 3(t-5.133)_+$ ($d_{LP} = 5.467$) and $\beta_{RTC}(t) = 3(t-6)_+$ ($d_{RTC} = 6.333$). Then the linear programming method, although not taking into account the service policy can lead to a better delay that RTC. Our aim is to find a method that will do better than all those existing methods.*

3. NETWORK MODEL AND FIRST BOUNDS

In this section, we give a first method for computing an upper delay bound for a given flow in a network. This method is inspired by the SFA and RTC methods. We conclude by a sufficient and necessary condition for the stability.

3.1 Network model

We consider an arbitrary network, where flows are totally ordered by their priorities. More precisely, let n be the number of servers $(1, \ldots, n)$ and m the number of flows $(f_1, \ldots, f_m)$. Each server j offers a strict service curve β_j and the arrival cumulative process of each flow f_i is α_i-upper constrained. Flows satisfy the priority property

$$\text{(PRIO)} \quad i < j \quad \Leftrightarrow \quad f_i \text{ has higher priority than } f_j.$$

Each flow f_i follows an acyclic path $path_i = \langle j_1, \ldots, j_{\ell_i}\rangle$. For each server we define $\mathrm{Fl}(j) = \{i \mid f_i \text{ crosses server } j\}$, the collection of flows that crossed server j and write down $\max\mathrm{Fl}(j) = \max\{i \mid i \in \mathrm{Fl}(j)\}$ the flow that belongs to server j and has the lowest priority. By convention, we set $\mathrm{Fl}(0) = [1, m]$.

In the network, we define a *trajectory* as a family of functions in $\mathcal{F}$, $(F_i^{(j)})_{j \in [1,n], i \in \mathrm{Fl}(j)}$. A trajectory is *admissible* if it satisfies the network calculus' constraints of the network: $\forall i \in [1, m]$, α_i is an arrival curve for $F_i^{(0)}$, and

$\forall j \in [1,n]$, $(\sum_{i\in \mathrm{Fl}(j)} F_i^{(\mathrm{prev}_i(j))}, \sum_{i\in\mathrm{Fl}(j)} F_i^{(j)}) \in \mathcal{S}_{strict}(\beta_j)$, where $\mathrm{prev}_i(j)$ is the server crossed by flow f_i just before server j (and 0 if j is the first server crossed by flow f_i). Thus $F_i^{(0)}$ is the arrival cumulative process of f_i in the system and $F_i^{(j)}$ is the cumulative departure process of flow f_i after server j.

3.2 Stability condition and bound

In order to compute an upper bound for the delay, one can compute for each server and each flow $\alpha_i^{(j)}$ and $\beta_j^{(i)}$ using Algorithm 1.

Algorithm 1: SFA with priorities

Data: Network description : paths of flows, α_i, β_j.
Result: $\alpha_i^{(j)}$ arrival curve for $F_i^{(j)}$, $\beta_j^{(i)}$ strict service curve for flow f_i at server j.

```
begin
    for j = 1 to n do β'_j ← β_j;
    for i = 1 to m do
        Let path_i = ⟨j_1 ... j_{ℓ_i}⟩; j_0 = 0;
        α_i^(0) ← α_i;
        for k = 0 to ℓ_i − 1 do
            α_i^(j_{k+1}) ← α_i^(j_k) ⊘ β'_{j_k};
            β_{j_k}^(i) ← β'_{j_k};
            β'_{j_k} ← (β_{j_k}^(i) − α_i^(j_k))_+;
end
```

In the next theorem, the arrival curve of each flow f_i is affine: $\alpha_i(t) = \sigma_i + \rho_i t$ and each server j offers a strict service curve $\beta_j(t) = R_j(t - T_j)_+$. The following theorem shows a sufficient and necessary condition for stability.

THEOREM 5. *Consider a network where flows have fixed priorities. This network is stable if and only if*

$$\forall j \in [1,n], \qquad R_j \geq \sum_{i\in Fl(j)} \rho_i.$$

PROOF. We are going to prove this result by induction on the number of flows in the network.

(H_k) The delay of $f_1, \ldots, f_k$ is finite and for each server j, the remaining service curve for flows $f_{k+1}, \ldots, f_m$ is of the form $\beta_j^{(k)}(t) = R_j^{(k)}(t - T_j^{(k)})_+$ with

$$R_j^{(k)} \geq \sum_{i\in\mathrm{Fl}(j)\cap[k+1,m]} \rho_i \text{ and } T_j^{(k)} < \infty.$$

(H_1) holds: the path of flow f_1 is $\langle j_1 \ldots j_{\ell_i}\rangle$. For all $j \in \{j_1, \ldots, j_{\ell_i}\}$, $R_j \geq \rho_1$. From the computations of Examples 1 and 2, the outgoing flows all have rate ρ_1, so f_1 experiences a finite delay and the remaining service curve for the other flows is $R_j^{(1)} = R_j - \rho_1$ and $T_j^{(1)} < \infty$. For the other servers, the service curve is unchanged, so one can set $R_j^{(1)} = R_j$ and $T_j^{(1)} = T_j$.

Suppose that (H_{k-1}) holds. Let us consider flow f_k. For every server j crossed by this flow, $R_j^{(k-1)} \geq \rho_k$. Then the delay experienced by flow k in each server is finite (using Example 1). The remaining service has rate

$$R_j^{(k)} = R_j^{(k-1)} - \rho_k \geq \sum_{i\in\mathrm{Fl}(j)\cap[k+1,m]} \rho_i.$$

The service rate of the other servers does not change: $R_j^{(k)} = R_j^{(k-1)} \geq \sum_{i\in\mathrm{Fl}(j)\cap[k+1,m]} \rho_i$. Then (H_k) holds and (H_m) is exactly what we are looking for. □

Note that if each server of a network satisfies the constraint $\sum_{f_i\in S_j, i\leq \ell} \rho_i < R_j$ then the worst-case delay for flow f_ℓ is finite.

The main drawback of this method is that it does not take into account the "pay multiplexing only once" phenomenon (PMOO). Indeed, consider two servers in tandem, crossed by two flows. The previous computation takes into account a possible burst in both servers, whereas that may not be possible given the arrival curves. This phenomenon has been detailed in [11]. The next section is an attempt to take this phenomenon into account, using linear programming.

4. LINEAR PROGRAMMING APPROACH

In this section, we will develop a method consisting in modeling the constraints by a linear program. Similar work has been done in the context of arbitrary multiplexing (nothing is known about the service policy of the servers) in [2]. We first explain how to take into account the priorities in a backlogged period. Then we present the constraints for tree-topology network and for general topologies. In this paragraph, we always assume that the arrival curves are piecewise affine concave and that the strict service curves are piecewise affine convex.

We first focus on a single server and a single backlogged period and explain how to encode the behavior with linear constraints.

4.1 Encoding the priorities in a single server

Consider a server that offers a strict service curve β and that is crossed by m flows $f_1, \ldots, f_m$ satisfying $(PRIO)$. For each flow f_i, let $F_i^{(0)}$ and $F_i^{(1)}$ be the respective arrival and departure cumulative functions for the server. In this paragraph, we are going to detail how to ensure that the priorities will be respected in this server during a given backlogged period. Given the date t, we study the period $[start(t), t]$. For this, we need to introduce intermediate dates c_i, $0 \leq i \leq m$ such that $c_0 = t$, the date of interest, $c_m = start(t)$ and

$$c_i = \sup\{u \leq t \mid \forall k \in [1,i],\ F_k^{(1)}(u) = F_k^{(0)}(u)\}.$$

Note that c_0 and c_m are compatible with this equation. The date c_i is the last instant at which flow f_{i+1} can be served. From this definition and (PRIO), the following properties hold:

- $c_0 \geq c_1 \geq \cdots \geq c_m$;
- $\forall i \in [0,m]$, $\forall k \leq i$, $F_k^{(0)}(c_i) = F_k^{(1)}(c_i)$;
- $\forall i \in [0,,m]$, $\forall k \geq i+2$, $F_k^{(1)}(c_i) = F_k^{(1)}(c_{i+1})$. Indeed, since from c_{i+1}, flow f_{i+1} is always backlogged, so flow f_{i+2} cannot be served.

Conversely, the following lemma shows that from an admissible trajectory satisfying those constraints, one can construct a new trajectory satisfying the same constraints and respects the priorities.

LEMMA 1. *Let $(F_i^{(0)}, F_i^{(1)})_{1\leq i\leq m}$ be an admissible trajectory. Suppose that there exist two dates $c < c'$ in the same backlogged period and i_0 such that*

$$\begin{aligned} \forall i \leq i_0, \quad & F_i^{(0)}(c) = F_i^{(1)}(c); \\ \forall i < i_0, \quad & F_i^{(0)}(c') = F_i^{(1)}(c'); \\ \forall i > i_0, \quad & F_i^{(1)}(c) = F_i^{(1)}(c'). \end{aligned}$$

Then, there exists an admissible trajectory $(F_i^{(0)}, \tilde{F}_i^{(1)})_{1\leq i\leq m}$ such that $\forall 1 \leq i \leq m$, $\tilde{F}_i^{(1)}(c) = F_i^{(1)}(c)$, $\tilde{F}_i^{(1)}(c') = F_i^{(1)}(c')$, $\forall t \in [c, c']$, $\sum_{1\leq i\leq m} \tilde{F}_i^{(1)}(t) = \sum_{1\leq i\leq m} F_i^{(1)}(t)$ and that respects the priorities between c and c'.

PROOF. In this proof, we will use the equivalence between strict service curves and variable capacity node for the type of curves we use (convex piecewise affine functions ultimately affine) and Lemma 2 of [1].

Let C be the amount of work offered by the server. As c and c' are in the same backlogged period, we have $\forall c \leq t \leq c'$, $C(t) - C(c) = \sum_i F_i^{(1)}(t) - F_i^{(1)}(c)$. In other words, $C(t) - C(c)$ is the global amount of service that is provided between c and t.

From that amount of service we are going to define a trajectory $\tilde{F}$ that respects the priorities. As $f_1, \ldots, f_i$, $i \leq i_0$ have the priority over flows $f_{i+1}, \ldots, f_m$ and c is the beginning of a backlogged period for $f_1, \ldots, f_i$, set

$$\sum_{1\leq k\leq i} \tilde{F}_k^{(1)}(t) = \inf_{c\leq s\leq t} \sum_{1\leq k\leq i} F_k^{(0)}(s) + C(t) - C(s). \quad (1)$$

This formula with $i = m$, ensures that $\sum_{1\leq i\leq m} \tilde{F}_i^{(1)} = \sum_{1\leq i\leq m} F_i^{(1)}$. The service offered to flows $f_i, \ldots, f_m$ between s and t is $C_i(t) - C_i(s) = C(t) - \sum_{k=1}^{i-1} \tilde{F}_k^{(1)}(t) - (C(s) - \sum_{k=1}^{i-1} \tilde{F}_k^{(1)}(s))$ so that we also have $\tilde{F}_i^{(1)}(t) = \inf_{c\leq s\leq t} F_i^{(0)}(s) + C_i(t) - C_i(s)$.

Indeed, we then have

$$\begin{aligned} \tilde{F}_i^{(1)}(t) + \sum_{k=1}^{i-1} \tilde{F}_k^{(1)}(t) &= \inf_{c\leq s\leq t} F_i^{(0)}(s) + C(t) - C(s) + \\ \sum_{k=1}^{i-1} \tilde{F}_k^{(1)}(s) &= \inf_{c\leq s\leq t} F_i^{(0)}(s) + C(t) - C(s) + \\ &\inf_{c\leq u\leq s} \sum_{k=1}^{i-1} F_k^{(0)}(u) + C(s) - C(u) = \\ &\inf_{c\leq u\leq s\leq t} F_i^{(0)}(s) + \sum_{k=1}^{i-1} F_k^{(0)}(u) + C(t) - C(u). \end{aligned}$$

The minimum is reached when $s = u$, then we find the original service.

From (1), $C_i = C - \sum_{k=1}^{i-1} \tilde{F}_k^{(1)}$ is non-decreasing. Then, it follows from the properties of variable capacity nodes ([6]) that $\tilde{F}_i^{(1)} \leq F_i(0)$ and $\tilde{F}_i^{(1)}$ is non-decreasing.

Then, the trajectory $(F_i^{(0)}, \tilde{F}_i^{(1)})$ is admissible and satisfies the priorities in the interval of time $[c, c']$. □

As a consequence, applying Lemma 1 to c_i and c_{i-1}, $1 \leq i \leq m$, one can construct from the constraints a trajectory for the backlogged period that respects the priorities, from the values on the trajectories on a finite set of dates.

4.2 Linear program for fixed priorities

We now describe the linear constraints and objective for a tree topology. The main difference with [2] is the existence of the priority constraints, which also request to consider more time variables. We will compute a worst-case delay upper bound for flow f_m when this flow ends at the root of the tree and the underlying structure of the network is a tree directed to that root. First let introduce some notations.

- n is the root server of the network;
- for $j \neq n$, $\text{next}(j)$ is the unique successor to server j, (if $j = n$, set $\text{next}(j) = n + 1$);
- for every server j that is not a leaf of the tree, $\text{prev}_i(j)$ is its predecessor regarding flow f_i (if j is the first server crossed by flow f_i then set $\text{prev}_i(j) = 0$).

Let $\mathcal{N}$ be a tree-network, with the notations used in Section 3.1.

Variables.

We consider two kinds of variables: the dates, denoted by $c_i^{(j)}$ and u; and the values of the trajectories at some dates, chosen amongst the $c_i^{(j)}$ and u and denoted by $F_i^{(j)}(t)$ where t is a date, and $F_i^{(j)}$ denotes the cumulative function of flow f_i after crossing server j. More precisely

- *date variables:* u represents the date at the bit of interest (satisfying the worst-case delay) enters the network and $\forall j$, $\forall i \in \{0\} \cup \text{Fl}(j)$, we have $c_i^{(j)}$. Thus there are at most $(m + 1)n$ date variables.
- *functional variables:* $F_m^{(0)}(u)$ and $\forall j$, $\forall i \in \text{Fl}(j)$, $\forall k \in \text{Fl}(j) \cup \{0\}$, we have variables $F_i^{(0)}(c_k^{(j)})$, $F_i^{(j)}(c_k^{(j)})$ and $F_i^{(j)}(c_k^{(\text{prev}_i(j))})$. Then there are at most $2m(m + 1)n$ functional variables.

The set of date variables is denoted by V_d and the set of functional variables is denoted by V_f.

Objective.

One wishes to maximize the time between the arrival date of the bit of data that suffers a maximum delay and its departure time: $\max c_0^{(n)} - u$.

Linear constraints.

To ensure that the computed delay is the delay suffered by a bit of data, one has $F_m^{(0)}(u) \geq F_m^{(n)}(c_0^{(n)})$.

Time constraints: $\forall j \in \{1, \ldots, n\}$, $i, i' \in \text{Fl}(j) \cup \{0\}$, $i > i'$,

- $c_i^{(j)} \leq c_{i'}^{(j)}$ and $c_0^{(j)} = c_{\max\text{Fl}(\text{next}(j))}^{(\text{next}(j))}$ (at most $n(m + 1)$ constraints). There is in fact no need of two variables in the latter case, but for simplicity, we can choose the notation. Moreover, this variable will also be denoted by $t_{\text{next}(j)}$ later.

Causality and non-decreasing constraints: $\forall j \geq j'$, $\forall t \leq t' \in V_d$ and $F_i^{(j)}(t), F_i^{(j')}(t), F_i^{(j)}(t') \in V_f$,

- $F_i^{(j)}(t) \leq F_i^{(j')}(t)$ and $F_i^{(j)}(t) \leq F_i^{(j)}(t')$ (respectively at most $2nm(m + 1)$ and $2nm^2$ constraints).

Arrival constraints: $\forall c_k^{(j)} \geq c_{k'}^{(j')} \in V_d$, $i \in \text{Fl}(j) \cap \text{Fl}(j')$,

- $F_i^{(0)}(c_k^{(j)}) - F_i^{(0)}(c_{k'}^{(j')}) \leq \alpha_i(c_k^{(j)} - c_{k'}^{(j')})$.

Note that as α_i is a concave piecewise affine function, it can be expressed as a finite minimum of affine functions and thus be expressed with linear constraints. If $|\alpha_i|$ is the number affine functions to describe α_i, then there are at most $\sum_{i=1}^m |\alpha_i|(n(m+1))^2$ constraints).
Service constraints: $\forall j$, $\forall k < k' \in \mathrm{Fl}(j) \cup \{0\}$,

- $\sum_{i \in \mathrm{Fl}(j)} F_i^{(j)}(c_k^{(j)}) - F_i^{(j)}(c_{k'}^{(j)}) \geq \beta_j(c_k^{(j)} - c_{k'}^{(j)})$.

Note that as β_j is a convex piece-wise affine function, it can be expressed as a finite maximum of affine functions and thus be expressed with linear constraints. If $|\beta_j|$ is the number affine functions to describe β_j, then there are at most $\sum_{j=1}^n |\beta_j|(m)^2$ constraints).
Backlog constraints: $\forall j$, $\forall i \in \mathrm{Fl}(j)$,

- $F_i^{(\mathrm{prev}_i(j))}(c_{\max \mathrm{Fl}(j)}^{(j)}) = F_i^{(j)}(c_{\max \mathrm{Fl}(j)}^{(j)})$ (at most mn constraints).

Priority constraints: $\forall k \in \mathrm{Fl}(j) \cup \{0\}$, $\forall i \in \mathrm{Fl}(j)$,.

- $\forall i \leq k$, $F_i^{(j)}(c_k^{(j)}) = F_i^{(\mathrm{prev}_i(j))}(c_k^{(j)})$ (at most $nm(m+1)$ constraints;
- $\forall i \geq k+2$[1], $F_i^{(j)}(c_k^{(j)}) = F_i^{(j)}(c_{k+1}^{(j)})$ (at most $nm(m+1)$ constraints).

We denote by λ_{prio} the set of linear constraints and by d_{prio} the optimal solution of this linear program.

Note that the number of variables and linear constraints and of is polynomial in the size of the description of the system.

4.3 Building trajectories

In this paragraph we look at an optimal solution of our linear program. We will convert this solution (*i.e.* a trajectory defined for a finite set of points) into a trajectory of the network that satisfy the constraints (arrival and service curves) for the backlogged periods that are considered. More precisely, we show that from the points of the form $F_i^{(0)}(s)$ one can extrapolate an α_i-upper constrained process and from the set $F_i^{(j)}(t_j)$, one can build a trajectory between t_j and $t_{next(j)}$ that guaranties a minimal strict service β_j.

LEMMA 2. *Let α a concave function in $\mathcal{F}$. Given $u_0 \leq u_1 \leq \cdots \leq u_N$ and a set $\{g_i\}_{0 \leq i \leq N}$ such that*

$$i < j \Rightarrow 0 \leq g_j - g_i \leq \alpha(u_i - u_j).$$

Then, one can extrapolate from the g_i's a function F that is non-decreasing and α-constrained such that $F(u_i) = g_i$, $0 \leq i \leq N$.

PROOF. Consider the function F such that

$$F(t) = \min(\min\{g_i + \alpha(t - u_i) \mid u_i \leq t\}, \min\{g_j \mid u_i \geq t\}).$$

Note that for all j such that $u_j \leq t$, $F(t) - g_j \leq \alpha(t - u_j)$. One can easily check that $F(u_j) = g_j$: $F(u_j) = \min_{j \geq i} g_i + \alpha(u_j - u_i) \leq g_j$. The minimum g_j is obtained for $i = j$.

Now consider $t_1 < t_2$ two arbitrary dates. If $F(t_1) \neq F(t_2)$, two cases can occur. Either there exists i such that $t_1 \leq u_i$ and $F(t_1) = g_i$, then $F(t_2) - F(t_1) = F(t_2) - g_i \leq \alpha(t_2 - u_i) \leq \alpha(t_2 - t_1)$.

Or there exists i such that $F(t_1) = g_i + \alpha(t_1 - u_i)$ and $F(t_2) - F(t_1) = F(t_2) - g_i - \alpha(t_1 - u_i) \leq \alpha(t_2 - u_i) - \alpha(t_1 - u_i) \leq \alpha(t_2 - t_1)$. □

[1] $k+2$ here stands for the second lower priority flow than f_k crossing j. This is an abuse of notation.

This lemma can be directly applied to functions $F_i^{(0)}$. We have already checked (Lemma 1) that in a backlogged period, one can construct from a set of points satisfying the linear constraints a trajectory that respects the priorities. It remains to prove that the total amount of service offered is a β strict service curve.

LEMMA 3. *Let β a convex function in $\mathcal{F}$. Given $u_0 \leq u_1 \leq \cdots \leq u_N$ and a set $\{g_i\}_{0 \leq i \leq N}$ such that*

$$i < j \Rightarrow g_j - g_i \geq \beta(u_j - u_i).$$

Then, one can extrapolate from the g_i's a function F that is non-decreasing and β-lower constrained such that $F(u_i) = g_i$, $0 \leq i \leq N$.

PROOF. The proof is similar to the proof of Lemma 2, taking $F(t) = \max\{g_i + \beta(t - u_i) \mid t \geq u_i\}$. □

Finally we have proved the following theorem.

THEOREM 6. *Consider a tree-like network crossed by flows totally ordered by their priority. From the set of constraints λ_{prio}, one can construct a trajectory for the network such that the priorities and arrivals/services constraints are satisfied on the backlogged period of a bit of data studied (every other constraints are satisfied). This set of constraints is polynomial is the description of the network.*

EXAMPLE 4. *The smallest network for which one can observe a difference between the arbitrary multiplexing case and the fixed priority case is depicted in Figure 1. This network is composed of two servers in tandem. Two flows, the ones with highest and lowest priority cross both of them, and the flow with intermediate priority only crosses the first server.*

More precisely, in the case where $\sigma_1 = \sigma_2 = \sigma_3 = \sigma$, $\rho_1 = \rho_2 = \rho_3 = \rho$, $R_1 - \rho > R_2$, and $T_2 = 0$, the formulas for the delays are:

$$\begin{aligned} d_{blind} &= \frac{R_2}{R_2 - \rho}\frac{\sigma + R_1T_1}{R_1 - \rho} + \frac{2\sigma}{R_2 - \rho} \\ d_{prio} &= \frac{2\sigma}{R_2 - \rho} + \frac{\sigma + R_1T_1}{R_1 - 2\rho}. \end{aligned}$$

Computations show that $d_{blind} > d_{prio}$ and the difference between the delays can be made arbitrarily large by choosing adequate parameters.

4.4 From a tree topology to a general topology

Now we have described the constraints that must be added in a single server, we will show that one can transform a network with a general topology into a tree by *unfolding* it. This *unfolded network* will have a greater delay bound than the original one.

Consider a network (we keep the same notations than in Section 3) satisfying (PRIO). A *non-increasing priority path* (nipp) is $\langle (j_0, f_{i_0}), \ldots, (j_p, f_{i_p}) \rangle$ a finite sequence of pairs (server, flow) such that

$$\begin{cases} \langle j_0, \ldots, j_p \rangle \text{ is a path in the network,} \\ \forall 0 \leq k \leq p,\ f_{i_k} \in \mathrm{Fl}(j_k), \\ \forall 0 \leq k < p,\ j_k = prev_{i_k}(j_{k+1}) \\ \forall 0 \leq k < p,\ i_k = \max\{i \leq k_{k+1} \mid i \in \mathrm{Fl}(j_k)\}. \end{cases}$$

Note that the number of nipp is finite. The *unfolded network* is a network where the servers are the nipp ending at $(last(m), f_m)$.

Let $\langle j_1, \dots j_{\ell_i}\rangle$ be the path of flow f_i in $\mathcal{N}$. In the unfolded net, we will have an α_i-constrained flow following the path $\langle \Pi_1, \dots \Pi_q\rangle$ with $\Pi_k = \langle (j_k, f_{i_k}), \dots, (j_q, f_{i_q}), \Pi\rangle$ and Π is either the empty path, or $q = \ell_i$, or $\Pi = \langle (j', f')\dots\rangle$ with $j' \neq j_{q+1}$ and $i_0 > i$.

For example, the unfolding of the network of Figure 3 is depicted in Figure 4.

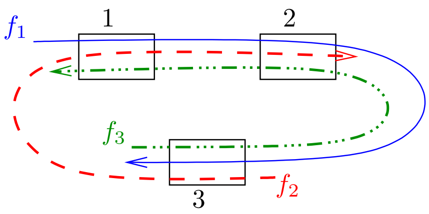

Figure 3: Example of a network with general topology.

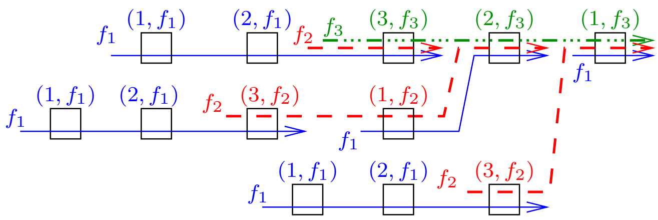

Figure 4: Unfolding of the network of Figure 3. For sake of clarity, only the first pair of each nipp is written.

The intuition of this unfolding is the following: one performs a backward unfolding from the last server crossed by flow f_m. We separate every provenance of the different flows by duplicating them. This results in duplication of both flows and servers. As the flows in the duplicated nodes cannot be influenced by flows with lower priorities, flows with lower priorities can be discarded from some point. As a consequence, this construction will stop.

Doing this, the number of variables and constraints becomes huge as the number of dates to consider is linear in the number of node of the unfolded network.

The constraints to compute the worst-case delay bound will be exactly the same as for networks with a tree topology, except for the arrival constraints: every duplicated flows of the same original flow f_i is in fact the same flow, so one writes the arrival constraints between every date of every branch where the flow is present (dates generated on the same branch can be compared but not dates generated for different branches).

For example of Figure 3, flow f_1 is duplicated several times on the upper branch, and the arrival constraints for flow f_1 will be written for every date defined for a server of that branch.

5. ACCURACY OF THE BOUNDS

In the previous section, we gave means to get more precise bounds than for the arbitrary multiplexing. Nevertheless, we did not prove that the bounds are tight. Indeed, we first show in this section an example where the bound is not tight and, even worse, the bound we compute is not tighter than the one obtained using Algorithm 1, that corresponds to the computations of already existing methods. Then, we show how to improve our linear program to reduce the bounds and outperform Algorithm 1 and finally give cases where our bound is tight.

5.1 Complete study of one simple example

To better understand the advantages and the limits of the solution we proposed, let us first focus on the simple example of Figure 5. The network is composed of three servers in tandem and three flows, f_1 and f_3 cross the three servers, whereas f_2 only crosses server 2.

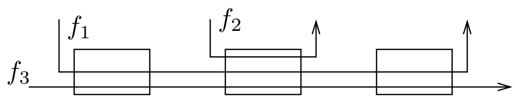

Figure 5: Network for complete study.

5.1.1 Comparison with SFA

As said in Section 3, the best solution to compute worst-case delay bounds is the SFA method with priorities (named SFA in the following), which corresponds to Algorithm 1. In most of the cases used for our comparison (as it will be shown in Section 6), the delay we compute using linear programming is smaller than the one computed using that algorithm. For example, with $\alpha_1 : t \mapsto 2t+2$, $\alpha_2 : t \mapsto 3t+3$, $\alpha_3 : t \mapsto 2$, $\beta_1 : t \mapsto 4(t-5)_+$, $\beta_2 : t \mapsto 8(t-4)_+$, $\beta_3 : t \mapsto 3(t-4)_+$, the delays computed with those two methods are: $d_{prio} = 52$ and $d_{sfa} = 60.6$.

Unfortunately, this is not always the case. Consider now the same example with the following arrival and service curves: $\alpha_1' : t \mapsto t+1$, $\alpha_2' : t \mapsto 2$, $\alpha_3' : t \mapsto 1$, $\beta_1' : t \mapsto 2t$, $\beta_2' : t \mapsto 4t$, $\beta_3' : t \mapsto 2t$. Computations give $d_{prio} = 2$ and $d_{sfa} = 5/6$.

The reason why SFA can reach tighter values than LP with fixed priorities is that we care about the service constraints for each server only for one backlogged period. For example, for the first server, we only considered the backlogged period between t_1 and t_2. Then, between time t_2 and t_3, we did not express the service of server 1 and then authorize some backlog for f_1.

One solution to obtain tighter bounds than for those two methods is to mix them. Indeed, one can encode the arrival constraints for each flow, for each server it crosses. This method will by construction give tighter bounds than both SFA and blind multiplexing exact method, but there is little chance to always get tight bounds, as this is the mixing of two non-tight methods.

The constraints we add are the following: $\forall j \in [1, n]$, $\forall i \in \mathrm{Fl}(j)$, $\forall k \geq k' \in \mathrm{Fl}(j) \cup \{0\}$,

$$F_i^{(j)}(c_{k'}^{(j)}) - F_i^{(j)}(c_k^{(j)}) \leq (\alpha_i^{(j)}(c_{k'}^{(j+1)} - c_k^{(j+1)}),$$

where $\alpha_i^{(j)}$ is computed using Algorithm 1. We denote by λ_{sfa} this set of linear constraints.

THEOREM 7. *Let $d_{prio-sfa}$ be the optimal solution com-*

puted using the linear constraints $\lambda_{prio} \cup \lambda_{sfa}$. Then

$$d_{prio-sfa} \leq d_{sfa} \text{ and } d_{prio-sfa} \leq d_{prio}.$$

Adding λ_{sfa} to the network of Figure 5 gives $d_{prio-sfa} = 49$ with the first set of functions and $d_{prio-sfa} = 5/6$ with the second set of functions (which in fact is the exact worst-case delay).

5.1.2 Adding more linear constraints

Another attempt to get tight bounds would be to add more linear constraints. Indeed, one could encode the service and priority constraints for every backlogged period into our linear program. Up to now, on server j, the service constraints are applied only between t_j and $t_{next(j)}$.

On the example of Figure 5, the key-point would be to take into account the backlogged periods of server 1 between time t_2 and t_3 (indeed, between time t_3 and t_4, to maximize the delay, server 1 and 2 must act as infinite servers, which respects the service constraints). More precisely, one needs to study then backlogged period of t_3 for server 1 and then introduce new dates $t_2' = start_1(t_3)$ and $c_1^{'(1)}$ (for the priority constraints).

But then, several cases can occur:

- $t_1 \leq c_1^{(1)} \leq t_2 \leq t_2' \leq c_1^{'(1)} \leq t_3$ or
- $t_1 = t_2' \leq c_1^{(1)} = c_1^{'(1)} \leq t_2 \leq t_3$ or
- $t_1 = t_2' \leq c_1^{(1)} \leq t_2 \leq c_1^{'(1)} \leq t_3$.

These different orders have to be interwoven with the possible values of $c_1^{(2)}$ and $c_2^{(2)}$, which will finally result in 31 possible orders. One only needs to take into account this backlogged period, as the other will not modify the delay (and one can then consider server 1 between t_2 and t_2' as an infinite server if this interval is non-empty). It is not now possible to compute the delay using a single linear program, but one for each possible order is necessary.

It is also quite clear that this method cannot be efficiently extended to more general cases: the number of backlogged periods to take into account will grow exponentially.

5.2 Some cases of tightness

We show some cases where the bound we compute is exactly the worst-case delay. We only focus here on tree networks. We suppose that the servers are numbered such that $j < \text{next}(j)$.

5.2.1 Shortest Destination First

The Shortest Destination First (SDF) [8] policy is the fixed-priority policy when a flow has higher priority than every flow whose destination is after its own destination. We will show that this policy is in fact the worst service policy regarding our model.

More formally, we set the following priorities:

$$\text{(SDF)} \qquad i_1 < i_2 \Rightarrow last(f_{i_1}) \leq last(f_{i_2}),$$

where $last(f_i)$ is the last server crossed by flow f_i. Note that we still assume that our flow of interest f_m ends at server n and has the lowest priority.

THEOREM 8. *The worst-case delay for a tandem network with arbitrary multiplexing can be obtained with an SDF policy.*

PROOF. To prove this theorem, let us consider trajectory $(F_i^{(j)})$ that reaches the worst-case delay. We are going to modify this trajectory into an new one that satisfies the SDF policy.

Let t_{n+1} be the date of exit of the bit of data that satisfies the maximum delay for flow f_m and $t_j = start_j(t_{\text{next}(j)})$. We first modify the trajectory $(F_i^{(j)})$ into $(\tilde{F}_i^{(j)})$ such that 1) before time t_j, ancestors k of server j act like infinite servers: $\forall t \leq t_j$, $\tilde{F}_i^{(k)}(t) = \tilde{F}_i^{(k-1)}(t)$; 2) after time $t_{\text{next}(j)}$, descendants k of server j act as infinite servers: $\forall t \geq t_{\text{next}(j)}$, $\tilde{F}_i^{(k)}(t) = \tilde{F}_i^{(k-1)}(t) = F_i^{(0)}(t)$; 3) during the backlogged period $]t_j, t_{\text{next}(j)}]$, the global service of server j is the same service provided for $(F_i^{(j)})$, and the service is made according to the SDF policy.

It is clear, since the dates t_j are not modified, that the delay computed with that trajectory is the same as in the original one. It is also clear that if the trajectory is modified according to the two first points, the trajectory remains admissible (the backlog in the unique backlogged period of each server only can only increase compared to the original trajectory). So, without loss of generality, one can assume that $(F_i^{(j)})$ already satisfies the two first points. It remains to prove that the trajectory $(\tilde{F}_i^{(j)})$ is admissible, that is $]t_j, t_{\text{next}(j)}]$ is actually a backlogged period for server j.

We show by induction on the number of servers that the backlogged that is transmitted is at least the one transmitted with the original trajectory. Set $\text{Fl}(j)_{\geq k} = \text{Fl}(j) \cap [k, n]$. More precisely, we show that for each server j, $\forall k$,

$$\sum_{i \in \text{Fl}(j)_{\geq k}} (\tilde{F}_i^{(\text{prev}_i(j))}(t_j) - \tilde{F}_i^{(j)}(t_j)) \geq \sum_{i \in \text{Fl}(j)_{\geq k}} (F_i^{(\text{prev}_i(j))}(t_j) - F_i^{(j)}(t_j)).$$

If j is a leaf, this is trivial since $\text{prev}_i(j) = 0$. Now, suppose that this is true all the $\cup_{i \in \text{Fl}(j)} \text{prev}_i(j)$. To prove this for server j, it suffices to show that for each $\text{prev}_i(j)$, the previous inequality holds at time t_j (and not at time $\text{prev}_i(j)$). Let $j' = \text{prev}_i(j)$. For an arbitrary k, let us respectively denote H, $\tilde{H}$, L and $\tilde{L}$ the burst transmitted at time $t_{j'}$ to server j' for the flows with priority higher (lower than k) for trajectory $(F_i^{(j)})$ ($(\tilde{F}_i^{(j)})$). We know that $L \leq \tilde{L}$ and $H + L \leq \tilde{H} + \tilde{L}$. Using the variable capacity node formulation, if $\tilde{H} - H \geq 0$, then the service given to the highest priority flow increases and the service given to the lower priority flows decreases. Otherwise, the difference of service provided to the higher priority flows between $(F_i^{(j)})$ and $(\tilde{F}_i^{(j)})$ is less than $H - \tilde{H}$. Then, the service provided to the lower priority can only increase by $H - \tilde{H} \leq \tilde{L} - L$, which ends the proof. □

5.2.2 Shortest to Destination Last of length 2

Another case of tightness is the case where every cross-traffic flow is of length at most 2 and the priorities satisfy: $i_1 \leq i_2 \Rightarrow last(i_1) \geq last(i_2)$ (2SDL). Figure 9 is an example of such a network.

THEOREM 9. *The optimal solution of Λ is exactly the worst-case delays when (2SDL) is satisfied.*

PROOF. Let $(F_i^{(j)})$ be the trajectory that is constructed from an optimal solution of Λ. The delay is not changed if this trajectory is modified as follow: server j is an infinite server after time $t_{next(j)}$. The global service is not changed during $[t_j, t_{next(j)}[$. The service of the flow with higher priority is also unchanged, as it is a new flow for server j. The only change in the trajectory is that it may have an additional service at time t_j for the flow with intermediate priority, and that flow leaves the network at that point, so there is no influence for the next servers.

Then, one can construct an admissible trajectory whose delay is exactly the optimal solution of Λ. □

5.3 Lower bound on the worst-case delay

In order to have an idea of the difference between the bounds we compute and the actual worst-case delay, one solution is to exhibit a trajectory that respects the NC constraints and compute its worst-case delay. This trajectory has to be well-chosen (trajectory that has a chance to meet the worst-case delay). Since the main problem with the method above is that we cannot efficiently model every backlogged period, a solution is to consider that every server is an infinite server except during the backlogged period that we model so that each server has only one backlogged period. This will give us a lower bound (d_{lowb}) of worst-case delay.

Consider a network with n servers and m flows. To describe an infinite service for each server after the first backlogged period we use the following linear constraints:

- $\forall j \in [1, n]$, $\forall i \in [1, m]$ if $i \in \text{Fl}(j)$, $\forall k \in \text{Fl}(next(j)) \cup \{0\}, F_i^j(c_k^{(\text{next}(j))}) = F_i^0(c_k^{(\text{next}(j))})$.

Note that this set of constraints is polynomial that does not increase the global complexity of the linear program.

EXAMPLE 5. *This method on the network represented in Figure 5, with the curves' set* α'_1, α'_2, α'_3, β'_1, β'_2, β'_3, *defined above, gives* $d_{lowb} = 39.25$. *With the best previous method we found* $d_{prio-sfa} = 49$.

In this section, we gave several ways to improve the worst-case delay bounds. The algorithmic cost of such improvements is very small, using Algorithm 1. One can also use those bounds to design more efficient algorithms to the price of loosening the bounds. For example, one can use the linear program encoding the blind multiplexing and add the SFA constraints of Algorithm 1. In a general topology, one can unfold up to a certain level and use the SFA constraints in order to keep the cost tractable. Another mean of improving our bounds is to take into account the constraints of RTC, namely maximal service curves and minimal arrival curves. These constraints can be used instead of those in λ_{sfa}.

6. NUMERICAL RESULTS

In this section, we aim to compare the bounds obtained with the different methods described in this article. We will study two topologies, one with nested flows and the other one with non-nested flows.

6.1 Nested-flow network

The network we study here has n servers in tandem and $n+1$ flows. Flow f_i crosses servers 1 to $n-i+1$, and flow f_{n+1} crosses every server.

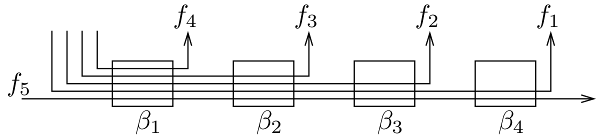

Figure 6: Nested-flow tandem network.

First, we study the precision of the results when the utilization rate of the servers varies. Server j guaranties a strict service curve $\beta_j : t \mapsto 3(n+1-j)(t-0.1)_+$: the latency is 0.1 s and the service rate is affine in the number of flows crossing it. Each flow has the same characteristics: the maximum burst is 1 Mbits and we make the long term arrival rate vary from 0.3 Mbit/s to 2.9 Mbit/s. Only last flow has a different arrival curve that have just a maximum burst of 1 Mbits. Figure 6 represents such a network for 4 servers.

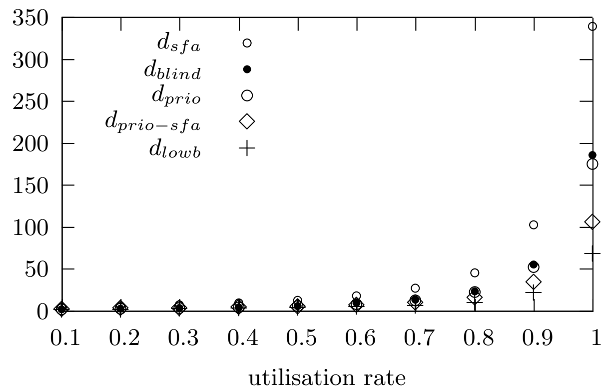

Figure 7: Delay bounds for the nested-flow tandem network with 8 servers.

Figure 7 gives the upper bounds on the delay found using different methods: SFA with priorities (d_{sfa}), linear programming with blind multiplexing (d_{blind}), linear programming with fixed priorities (d_{prio}), linear programming using priorities and SFA constraints ($d_{prio-sfa}$), and linear programming that returns a lower bound on the delay (d_{lowb}) when there are 8 servers in function of the arrival rate of the flows.

Figure 8 gives the delay bounds found with these different methods when network size grows from 2 to 10 servers. Server j guaranties the same strict service curve β_j as previously and arrival curves of flows is $\alpha_i : t \mapsto 1 + 2.5t$, $i \neq m$ and for the lowest priority flow, $\alpha_m : t \mapsto 1$.

We first notice that the SFA method is over-pessimistic method in both cases. Indeed, d_{sfa} can be five times higher than d_{prio}. Secondly, one can check that $d_{prio-sfa} \leq d_{prio} \leq d_{blind}$. Moreover, d_{lowb} is close to these values and thus in this case with those methods the bounds are not that far from the actual worst-case delay.

6.2 Study of a second topology of network

We now study a network composed of n servers in tandem and $n+2$ flows. Each flow intersects two servers (except at the extremities), and the flow of interest (f_{n+2}) crosses every server. Figure 9 represents such a network with 4 servers. Servers have the same service curve, $\beta : t \mapsto 10(t-5)_+$. Flows also have the same arrival curve, $\alpha_i : t \mapsto 1 + 2t$,

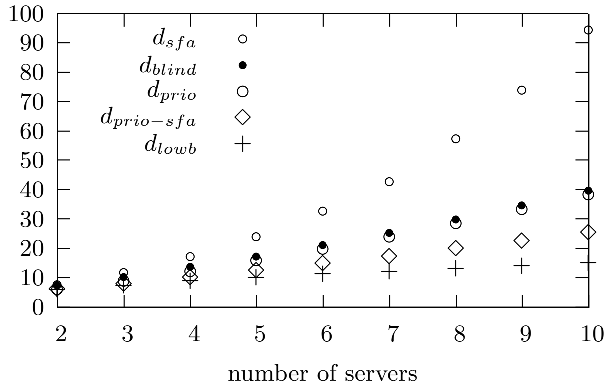

Figure 8: Delay bounds for the nested-flow tandem network when the number of servers varies.

$i \neq 1$, except for f_1, $\alpha_1 : t \mapsto 1 + 4t$.

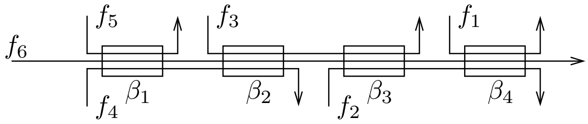

Figure 9: Non-nested-flow tandem network.

Delays computed with the different methods are depicted on Figure 10.

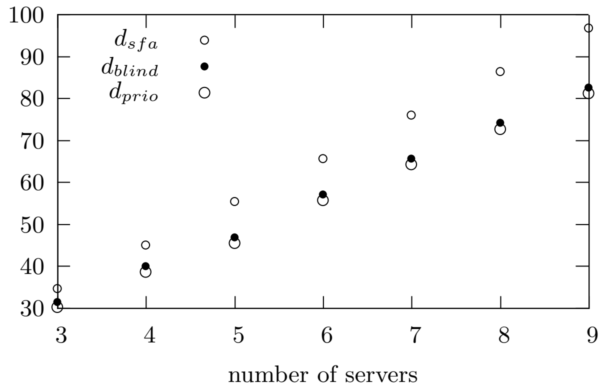

Figure 10: Delay bounds in the non-nested-flow tandem network.

Note that $d_{prio-sfa}$ and d_{lowb} are not represented since we are in the case of application of Theorem 9 and they are equal d_{prio}, which is the actual worst-case delay. We still observe that SFA method is over-pessimistic. Furthermore d_{blind} is very closed to worst-case delay.

We can conclude that the linear programming with SFA constraints is an efficient method to compute worst-case delay in those networks. Furthermore in some cases it reaches the worst-case delay.

7. CONCLUSION

This article presented several methods to compute worst-case delay upper bounds in tandem networks, where flows are ordered according to fixed priorities. Furthermore we implement a method that computes a lower bound of the worst case delay. This study leads to tighter bounds than older methods applied on network with fixed priorities service policy.

The algorithms developed here have a polynomial-time complexity; however they can be intractable for large network. But a trade-off between algorithmic efficiency and complexity can be made.

Future work will include the extension of these results to arbitrary topologies and will combine the linear programming methods with network calculus methods to other service policies (FIFO, GPS, etc.).

8. REFERENCES

[1] A. Bouillard, L. Jouhet, and É. Thierry. Service curves in Network Calculus: dos and don'ts. Research Report RR-7094, INRIA, 2009.

[2] A. Bouillard, L. Jouhet, and É. Thierry. Tight Performance Bounds in the Worst Case Analysis of Feed Forward Networks. In *Proc. of INFOCOM'10*. IEEE, 2010.

[3] 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.

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

[5] J.-P. Georges, E. Rondeau, and T. Divoux. Evaluation of switched Ethernet in an industrial context by using the Network Calculus. In *Proc. of 4th IEEE Workshop on Factory Communication Systems*, pages 19–26, 2002.

[6] 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.

[7] L. Lenzini, L. Martorini, E. Mingozzi, and G. Stea. Tight end-to-end per-flow delay bounds in FIFO multiplexing sink-tree networks. *Performance Evaluation*, 63(9-10):956–987, 2006.

[8] M. Mavronicolas. Stability in Routing: Networks and Protocols. *Bulletin of the EATCS*, 74:119–134, 2001.

[9] L. Saïdane, S. Azzaz, S. Martin, and P. Minet. FP/FIFO Scheduling: Deterministic Versus Probabilistic QoS Guarantees and P-Schedulability. In *ICC*, pages 518–523, 2007.

[10] J. B. Schmitt, F. A. Zdarsky, and M. Fidler. Delay Bounds under Arbitrary Multiplexing. Technical report, University of Kaiserslautern, 2007.

[11] 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.

[12] 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.

[13] H. Thomas, H. Kai-Steffen, K. Ulrich, and G. Reinhard. Stochastic and deterministic performance evaluation of automotive CAN communication. *Comput. Netw.*, 53:1171–1185, June 2009.