Trade-off between Accuracy and Tractability of Network Calculus in FIFO Networks

Anne Bouillard
anne.bouillard@huawei.com

October 27, 2021

Abstract

Computing accurate deterministic performance bounds is a strong need for communication technologies having stringent requirements on latency and reliability. Within new scheduling protocols such as TSN, the FIFO policy remains at work inside each class of communication.

In this paper, we focus on computing deterministic performance bounds in FIFO networks in the Network Calculus framework. We propose a new algorithm based on linear programming that presents a trade-off between accuracy and tractability. This algorithm is first presented for tree networks. Next, we generalize our approach and present a linear program for computing performance bounds for arbitrary topologies, including cyclic dependencies. Finally, we provide numerical results, both of toy examples and realistic topologies, to assess the interest of our approach.

1 Introduction

Some communication technologies aim at providing deterministic services, with strong requirements on buffer occupancy, latency and reliability. An example of such a standard under discussion is Time-Sensitive Networking (TSN), which is part of the 802.1 working group [1] and has potential applications to industrial and automotive networks. In this new communication paradigm, critical traffic (having strong delay and reliability requirements) and best-effort traffic can share switches and routers. Even though scheduling policies have been defined to cope with these heterogeneous traffic classes, it is a necessity to develop tools for accurately dimensioning the bandwidth allocated to each class.

Properly dimensioning a network relies on the ability to compute accurate performance bounds (delay or buffer occupancy) in networks. As far as deterministic performance bounds are concerned, one popular theory is Network Calculus, which is based on the (min, plus) semi-ring. Elements of the network, such as the traffic flows and switches, are described by *curves*, and upper bounds of the performances (delay, buffer occupancy) are computed from this description. This theory has already been successfully applied to various types of networks. One can cite switched network [2], ATM networks [3], AFDX (Avionics Full Duplex) networks [4], TSN/AVB [5, 6].

Different solutions have recently been proposed to analyze these types of networks with Network Calculus. It is first required to give a precise modeling of the scheduling policy – for example, priorities or processor sharing scheduling, such as DRR (Deficit Round Robin) [7] and WRR (Weighted Round Robin) – to deduce network guarantees for flows scheduled in the same *class*, where the FIFO (First In First Out) policy is at work. Being able to compute accurate performance bounds is FIFO networks is then crucial.

Recent works focused on the analysis of FIFO networks, and their main goal was to reduce the computational cost for deriving performance guarantees (upper bounds of worst-case delay). For example, Mohammadpour et al. propose in [8] a modeling of TSN, and the insertion of regulators [9] to control the arrival processes at each router; Thomas et al. compare in [10] the analysis with partial insertion of regulators (from complete to none) using TFA++ (total flow analysis) proposed in [11]. These analyses have a very low complexity, which allows the analysis of large-scale networks, but can have pessimistic bounds.

Other works focused on the accuracy of the bounds computed, in order to get the tightest result possible. From the first paper on Network Calculus, phenomena such as the *pay burst only once* (the burst parameter of the flow under analysis impacts the computation of its end-to-end performance once) and the *pay multiplexing only once* (the burst parameter for any competing flow impacts the computation of the end-to-end performance of a flow once) have been investigated, and each time they led to improvements of the performance bounds. More recently, algorithms based on linear programming have been proposed in [12, 13] to compute tight bounds in FIFO networks, but the complexity of these algorithms is too high to be used in most of the networks. Nevertheless, some networks are not so large that they require very low complexity performance bounds. For example, Zhang et. al present in [14] a TSN industrial network with less than 20 nodes, where every flow crosses at most five routers; Zhao et al. study a variety of networks in [6]. The largest one has 15 switches and each class of AVB traffic has at most 25 flows (the network is presented in [15]). The network of [14] is small enough to be analyzed with the linear-programming techniques from [13]. With 15 switches (resulting in 100 servers in the Network Calculus modeling), the network of [15] might be too large to be directly analyzed that way. However, the performances of this network could benefit from a more accurate analysis at a computational cost that would still be manageable, even though this would be out of reach for larger networks.

Objective and contributions The objective of this paper is to explore a solution between these two extremes, which is both tractable and leads to accurate bounds. We introduce a new polynomial-size linear-programming technique to compute performance bounds in FIFO networks, which presents a good trade-off between complexity and accuracy to analyze medium-size networks. Furthermore, we compare this algorithm with different existing Network Calculus methods. More precisely, our contributions are the following.

1. We first propose a simplified model (with respect to that presented in [12]) for a linear program computing bounds in FIFO trees. This model can also take into account the shaping of transmission links. While losing some accuracy, this algorithm is more tractable, and achieves better performance bounds than the other methods in the literature in most cases (obvious exceptions are when tight bounds with explicit formulas are already known, as in [16]).

2. We generalize the linear-programming technique to networks with cyclic dependencies. This will then improve the stability region compared to previous works that combine the fixed-point method and the more classical analysis techniques (TFA/TFA++ and SFA described in Section 3).

3. We compare our algorithms against the literature in both toy examples (tandems and rings) and realistic use-cases.

The rest of the paper is organized as follows. First, the Network Calculus framework and our network model are briefly recalled in Section 2. The state of the art on Network Calculus

for FIFO networks is described in Section 3. In Section 4, we present the first contribution of the paper, that is, a new linear programming proposal to compute performance bounds in FIFO tree networks, in polynomial time. This approach is generalized in Section 5 and 6 respectively to the case of feed-forward networks and networks with a general topology, including cyclic dependencies. Finally, we compare the new algorithm with the state of the art in several examples in Section 7 before concluding.

2 Network Calculus framework

In this section, we recall the Network Calculus framework and present the basic results that will be used in the next parts of the paper. More details about the framework can be found in [17, 18, 19].

We will use the following notations: $\mathbb{R}_+$ is the set of non-negative reals, for all $n \in \mathbb{N}$, $\mathbb{N}_n = \{1, \dots n\}$, and for all $x \in \mathbb{R}$, $[x]_+ = \max(0, x)$.

2.1 Arrival and service curves

Data processes and arrival curves Flows of data are represented by cumulative processes. More precisely, if $A : \mathbb{R}_+ \to \mathbb{R}_+$ represents a flow at a certain point in the network, $A(t)$ is the amount of data of that flow crossing that point during the time interval $[0, t)$, with the convention $A(0) = 0$. The cumulative processes are non-decreasing, left-continuous and null at zero. We denote by $\mathcal{F}$ the set of such processes.

A cumulative process A is constrained by the arrival curve α, or is α-constrained, if

$$\forall s, t \in \mathbb{R}_+ \text{ with } s \leq t, \quad A(t) - A(s) \leq \alpha(t - s).$$

In the following we will mainly consider *token-bucket* curves: $\gamma_{b,r} : 0 \mapsto 0;\ t \mapsto b + rt$, if $t > 0$. The *burst* b can be interpreted as the maximal amount of data that can arrive simultaneously and the *arrival rate* r as a maximal long-term arrival rate of data.

Servers and service curves An n-server $\mathcal{S} \subseteq \mathcal{F}^n \times \mathcal{F}^n$ (illustrated for $n = 1$ in Figure 1) is a relation between n arrival processes $(A_i)_{i=1}^n$ and n departure processes $(D_i)_{i=1}^n$ such that $A_i \geq D_i$ for all $i \in \mathbb{N}_n$, whenever $((A_i)_{i=1}^n, (D_i)_{i=1}^n) \in \mathcal{S}$. We assume that the servers are causal. This means in particular that $A_i \geq D_i$, and that $D_i(t)$ only depends on $(A_j(s))_{j \in \mathbb{N}_n, s \leq t}$.

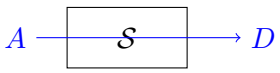


Figure 1: Server model.

The role of a service curve is to constrain the relation between the inputs of a server and its outputs.

We say that $\beta \in \mathcal{F}$ is a *service curve* for 1-server $\mathcal{S}$ if

$$\forall (A, D) \in \mathcal{S}, \quad A \geq D \geq A * \beta, \tag{1}$$

where $*$ is the (min, plus)-convolution: for all $t \geq 0$, $A * \beta(t) = \inf_{0 \leq s \leq t} A(s) + \beta(t - s)$. In the following we will use

- *rate-latency* service curves: $\beta_{R,T} : t \mapsto R[t - T]_+$, where T the *latency* until the server becomes active and R as its minimal *service rate* after this latency;

- *pure delay* service curves: $\delta_d : t \mapsto 0$ if $t \leq d$; $t \mapsto +\infty$ if $t > d$. We have $A * \delta_d(t) = A((t-d)_+)$. In other words, all data are delayed by a delay d.

An n-server $\mathcal{S}$ offers a service curve β if it offers the service curve β for the aggregate flows: for all $((A_i)_{i=1}^n, (D_i))_{i=1}^n \in \mathcal{S}$, $(\sum_{i=1}^n D_i) \geq (\sum_{i=1}^n A_i) * \beta$ and $A_i \geq D_i$ for all $i \in \mathbb{N}_n$. We call the flow with arrival process $\sum_{i=1}^n A_i$ the aggregate process of flows $1, \ldots, n$.

FIFO service policy In this paper, we assume that the service policy in this system is FIFO (First-In-First-Out): data are served in their arrival order. Figure 2 illustrates the FIFO policy in the case of a 2-server. The service of data can be decomposed in three steps: first the arrival processes are aggregated ($A = A_1 + A_2$); second the departure process in computed ($D_1 + D_2 = D \geq A * \beta$); third, D_1 and D_2 are computed using the FIFO property:

$$\begin{aligned} D(t) \geq A(u) &\Leftrightarrow \forall i \in \mathbb{N}_n, D_i(t) \geq A_i(u), \\ D(t) \leq A(u) &\Leftrightarrow \forall i \in \mathbb{N}_n, D_i(t) \leq A_i(u). \end{aligned} \tag{2}$$

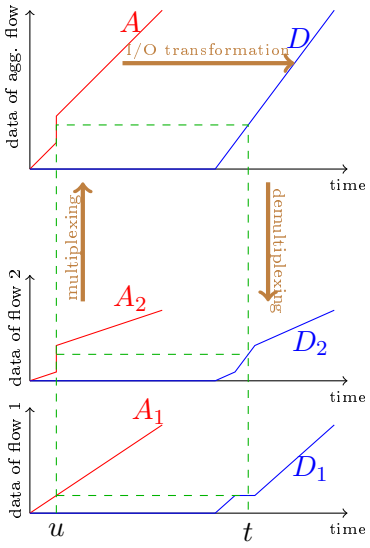


Figure 2: Description of the FIFO policy for a 2-server (from [12]).

It is possible to find service guarantees for individual flows. The following theorem gives the result for 2-server. It can be generalized to an n-server by first aggregating flows $2, \ldots, n$ to compute the residual service curve of flow 1.

Theorem 1 ([18, Proposition 6.2.1]). *Consider a FIFO server with service curve β, crossed by two flows with respective arrival curves α_1 and α_2. For all $\theta \geq 0$, β_θ is a residual service curve for the first flow, with*

$$\beta_\theta = [\beta - (\alpha_2 * \delta_\theta)]_+ \wedge \delta_\theta.$$

One can notice that the service curves computed when θ varies are not comparable (*i.e.*, they might intersect each other), and lead to different performances.

Greedy shapers In most networks, the transmission rate is physically limited by the capacity of a wire or a channel, which limits the quantity of data that can be transmitted to the next server. This phenomenon is taken into account by *greedy shapers*. Let B be a cumulative process, crossing a token-bucket greedy shaper $\sigma = \gamma_{L,C}$. The departure process is $D = B * \sigma$. Here

C represents the maximum capacity of the server, and L can represent a packet length, hence takes into account the packetization effect.

A server whose transmission rate is limited by a token-bucket greedy shaper can then be modeled by a system that is composed of a server β and a greedy shaper σ, as depicted on Figure 3. We will always assume that $\sigma \geq \beta$, which is not a restriction since the service offered to a flows is limited by the physical limitations of the server.

Consider a system consisting in a 1-server with service curve β followed by a greedy-shaper σ. The departure process then satisfies:

$$D = B * \sigma \geq (A * \beta) * \sigma = A * (\beta * \sigma) = A * \beta,$$

where the last equality comes from $\beta \leq \sigma$ and $\beta(0) = \sigma(0) = 0$.

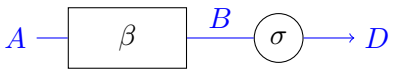


Figure 3: Shaping of the departure process.

As a consequence, the whole system still offers a service curve β.

Output arrival curve A departure process is also characterized by an arrival curve. Such an arrival curve can be computed as a function of the arrival curve of the arrival process and the shaping and service curves of the server.

Theorem 2 ([17, Theorem 5.3]). *Suppose that A is α-constrained and crosses a server offering the service curve β and with greedy shaper σ. Then the departure process D is $(\alpha \oslash \beta) \wedge \sigma$-constrained, where $\oslash$ is the (min, plus)-deconvolution: $\alpha \oslash \beta(t) = \sup_{u \geq 0} \alpha(t+u) - \beta(u)$.*

Functions δ_0 or $\varepsilon : t \mapsto \infty$ are valid greedy-shapers for all servers. In case of a token-bucket arrival curve $\alpha = \gamma_{b,r}$ and rate-latency service curve $\beta = \beta_{R,T}$ with $R > r$, $(\alpha \oslash \beta) \wedge \delta_0 = \gamma_{b+rT,r}$ is an arrival curve for the departure process (the only difference between $\alpha \oslash \beta$ and $(\alpha \oslash \beta) \wedge \delta_0$ is the value at 0).

2.2 Performance guarantees in a server

Backlog and delay Let $\mathcal{S}$ be a 1-server and $(A, D) \in \mathcal{S}$. The backlog of that server at time t is $b(t) = A(t) - D(t)$. The worst-case backlog is then $b_{\max} = \sup_{t \geq 0} b(t)$.

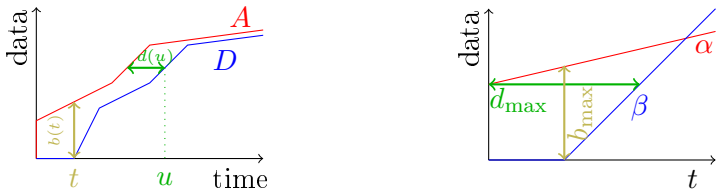
We denote $b_{\max}(\alpha, \beta)$ the maximum backlog obtained for an α-constrained flow crossing a server offering the service curve β. It has been shown to be the maximum vertical distance between α and β. For example, we have $b_{\max}(\gamma_{b,r}, \beta_{R,T}) = b + rT$ if $r \leq R$.

The delay of data exiting at time t is $d(t) = \sup\{d \geq 0 \mid A(t-d) > D(t)\}$. The worst-case delay is then $d_{\max} = \sup_{t \geq 0} d(t)$.

We denote $d_{\max}(\alpha, \beta)$ the maximum delay that can be obtained for an α-constrained flow crossing a server offering the service curve β. It can be shown to be the maximum horizontal distance between α and β. For example, we have $d_{\max}(\gamma_{b,r}, \beta_{R,T}) = T + \frac{b}{R}$ if $r < R$.

Backlog and delay are illustrated on Figures 4a and 4b.

From performance bounds to output arrival curves It is also possible to compute alternative arrival curves of the output processes using delay and backlog upper bounds of the servers.



(a) Processes (b) Performances

Figure 4: Processes and worst-case performance.

Theorem 3. *Consider a FIFO server crossed by an α-constrained flow, among others. Suppose that d is an upper bound of the delay of this flow. Then $\alpha \oslash \delta_d$ is an arrival curve for the departure process.*

Theorem 3 is a fundamental result for the TFA method described in Paragraph 3.1 and is proved in A. The next theorem will be useful to compute performances in feed-forward networks and networks with cyclic dependencies in Sections 5 and 6.

We need the following additional hypotheses:

(H_1) The arrival process of flow of interest (f.o.i.) is α-constraint with $\alpha = \gamma_{b,r}$, and α is the only arrival constraint for this flow, meaning that all arrival processes that are α-constrained are possible.

(H_2) The last server crossed by the f.o.i. offers a convex service curve β_n. Moreover, for any aggregate arrival process A_n to server n, for any aggregate departure process D_n from server n, for all $t \geq 0$, there exists an admissible departure process D'_n from server n such that 1) $\forall s \leq t$, $D_n(s) = D'_n(s)$; 2) D'_n is right-continuous at t.

Theorem 4. *Consider a system crossed by a flow of interest satisfying (H_1) and (H_2). Let B be the largest backlog that can be achieved by the f.o.i., for any of its possible arrival and departure processes. Then $\alpha' = \gamma_{B,r}$ is an arrival curve for the departure process.*

A similar result has already been proved in [20] in a slightly different setting (strict service curves) and in [21] when a service curve for the whole system is known for the α-constrained flow. For the sake of completeness, we provide the proof in B and show a useful case where (H_2) is satisfied (this is the case Theorem 4 will be applied in Section 5).

2.3 Network model

Consider a network $\mathcal{N}$ composed of n servers numbered from 1 to n and crossed by m flows named $f_1, \ldots, f_m$, such that

- each server j guarantees a service curve β_j and has a greedy shaper σ_j. The service policy is FIFO;
- each flow f_i is α_i-constrained and circulates along an acyclic path $\pi_i = \langle \pi_i(1), \ldots, \pi_i(\ell_i) \rangle$ of length ℓ_i.

We will always assume in the following that arrival curves and greedy shapers are token-bucket ones and the service curves rate-latency ones. We will use the following additional notations:

- $F_i^{(j)} \in \mathcal{F}$ is the cumulative process of flow i entering server j. The departure process after the last server crossed by flow f_i is be denoted $F_i^{(n+1)}$;

- the arrival curve of $F_i^{(j)}$ is denoted $\alpha_i^{(j)} = \gamma_{b_i^{(j)},r_i}$. In particular, $F_i^{(\pi(1))}$ is α-constrained and $b_i^{(\pi(1))} = b_i$;
- the service curve of server j is $\beta_j = \beta_{R_j,T_j}$ and the shaping curve is $\sigma_j = \gamma_{L_j,C_j}$;
- for a server j, we define $\mathrm{Fl}(j) = \{i \mid \exists \ell,\ \pi_i(\ell) = j\}$ the set of indices of the flows crossing server j and $\mathrm{Fl}(j,h) = \{i \mid \exists \ell,\ (\pi_i(\ell), \pi_i(\ell+1)) = (j,h)\}$ the set of indices of the flows crossing servers j and h in sequence;
- for all flows f_i, for $j \in \pi(i)$, we denote by $\mathrm{succ}_i(j)$ is the successor of server j in flow f_i. If $j = \pi(\ell_i)$, then $\mathrm{succ}_i(j) = n+1$. For all servers j, $\mathrm{prec}(j)$ is the set of predecessors of server j.

We call the family of cumulative $(F_i^{(j)})_{i \in \mathbb{N}_m, j \in \pi_i \cup \{n+1\}}$ a trajectory of the network, and an admissible trajectory if it satisfied all the Network Calculus constraints described above: arrival, service and shaping constraints and FIFO scheduling (in particular arrival and departure processes of each server satisfy the properties of Equation (2)).

The induced graph $G_{\mathcal{N}} = (\mathbb{N}_n, \mathbb{A})$ is the directed graph whose vertices are the servers and the set of arcs is

$$\mathbb{A} = \{(\pi_i(k), \pi_i(k+1)) \mid i \in \mathbb{N}_m, k \in \mathbb{N}_{\ell_i - 1}\}.$$

We will consider different types of topologies:

- if no assumption is made about the induced graph, we say that the network has a *general topology*;
- if the induced graph $G_{\mathcal{N}}$ is acyclic, we say that the network is feed-forward;
- if the induced graph $G_{\mathcal{N}}$ contains cycles, we say that the network has cyclic dependencies (or is not feed-forward);
- if the induced graph $G_{\mathcal{N}}$ is an out-tree [22, page 207] (the graph is simply connected and all vertices have out-degree 1 except one, named the root, that has out-degree 0), we say that the network is a tree network;
- if the induced graph $G_{\mathcal{N}}$ is linear (it is an out-tree and moreover all vertices have in-degree at most one), we say that the network is a tandem network;

Since we will focus on the performances of a flow (either its maximum end-to-end delay or the maximum amount of data in transit along its path), we can restrict the analysis to a sub-part of the network. The network induced by flow f_i, denoted $\mathcal{N}(f_i)$ is a sub-network of $\mathcal{N}$ defined as follows:

- the servers are the servers j of $\mathcal{N}$ for which there exists a path in $G_{\mathcal{N}}$ from j to $\pi_i(\ell_i)$, the destination of flow f_i;
- the flows are the flows of $\mathcal{N}$, with paths restricted to the servers of $\mathcal{N}(f_i)$. Remark that the new paths are necessarily prefixes of the original paths;
- service curves and greedy shapers of the kept servers remain unchanged, as well as the arrival curves of the flows. If the path of a flow becomes empty, it can be safely removed from the network.

Figure 5 illustrates a tree network, with induced graph is $(\{1,2,3,4,5\}, \{(1,3),(2,3),(3,5),(4,5)\})$. In this example, $\pi_1 = \langle 2,3,5\rangle$ and $\mathrm{Fl}(5) = \{1,3,4\}$. The network induced by flow f_2, $\mathcal{N}(f_2)$, is composed of servers $\{1,2,3\}$, and the paths become $\pi'_1 = \langle 2,3\rangle$, $\pi'_2 = \langle 1,3\rangle$, $\pi'_3 = \langle 3\rangle$ and $\pi'_4 = \langle\rangle$. Flow f_4 can be removed from $\mathcal{N}(f_2)$.

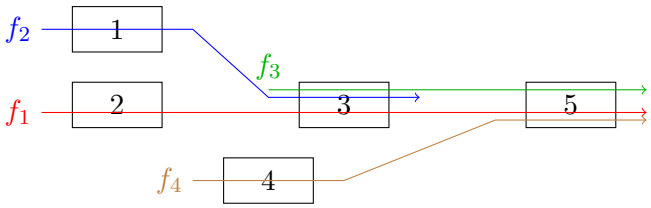


Figure 5: Example of a tree network.

Stability We will also be interested in the network stability.

Definition 1 (Global stability)**.** *A network is* globally stable *if there exists $B < \infty$ such that the backlog of each server is bounded by B.*

Note that we define the global stability as being able to find a bound on the backlog in each server. In the following, we will rather bound the maximum *end-to-end backlog* of a flow (*i.e.*, the maximum amount of data in transit at any time in a flow). End-to-end backlogs can also be used as a global stability condition: if B' is a bound on the end-to-end backlog for each flow, then mB' is a backlog bound for each server. Conversely, if B is a bound of the backlog of each server, then nB is a bound for the end-to-end backlogs. This definition is also equivalent to that of [23], where the condition is that the total backlog in the system is bounded.

Deciding if a network is stable is an open problem in the Network Calculus setting, and only partial results exist. A necessary condition is that the arrival rate in each server is less than the service rate, but this condition is not sufficient: Andrews showed in [24] that there exists FIFO networks with arbitrary small *local* loads that can be unstable. Other works concern the definition of sufficient conditions for stability. For example Charny and Le Boudec [25] derive a sufficient stability condition (and delay bound) that depends on the maximum length h of a flow. They show that the network is stable if the load of each server is less than $a/(b(h-1)+1)$, where a and b depend on the service and shaping rates. Rizzo and Le Boudec [26] derive another sufficient stability condition based on the *route interference number* (RIN), the number of interfering flows at each server. Finally, Rizzo and Le Boudec improve in [23] the stability condition based on the fixed point of a (non-linear) operator. Another, more classical, direction is to compute arrival curves of each flow at the input of each server it crosses as a fixed point. This technique has mainly been applied to networks with arbitrary multiplexing [20, 27], and more recently to FIFO networks [10].

Local stability refers to the arrival rate being less than the service rate in every server of the network. In the following, we will always assume local stability. In our setting, this means that for all server j, $\sum_{i \in \mathrm{Fl}(j)} r_i \leq R_j$.

3 State of the art on computing bounds in FIFO networks in Network Calculus

In this section, we describe the state-of-the-art methods to compute performance bounds in FIFO networks using Network Calculus. For all those methods, we assume that the network is locally stable.

3.1 TFA (Total flow analysis) and TFA++

the FIFO versions of TFA and TFA++ are based on Theorem 3: the worst-case delay in a FIFO server is the same for the aggregate flow and all flows crossing it. Servers are treated in

the topological order, as described in Algorithm 1, specified for token-bucket arrival curves and rate-latency service curves. First an arrival curve α of the aggregate arrival process is computed – only the burst parameters are useful, thus only the sum of the burst parameters is computed (line 3); then an upper bound on the delay of the server d_j is computed as the horizontal distance between α and the service curve β_j (line 4): δ_{d_j} is a residual service curve for all flows crossing server j; last, the arrival curve of the departure processes are computed using Theorem 3 (line 5): $\alpha_i^{(\mathrm{succ}_i(j))} = \alpha_i^{(j)} \oslash \delta_{d_j} \wedge \delta_0 = \gamma_{b_i^{(j)}+r_i d_j, r_i}$. Again, as the arrival rate remains r_i, we only need to compute the burst parameter of the arrival curves computed.

Finally, the delay of flow f_i is computed by summing all the delays of the servers on its path.

Algorithm 1: TFA analysis: delay of flow f_i

```
1 begin
2     foreach server j in the topological order do
3         b ← Σ_{i∈Fl(j)} b_i^(j);
4         d_j ← T_j + b/R_j;
5         foreach flow f_i ∈ Fl(j) do b_i^(succ_i(j)) ← b_i^(j) + r_i d_j;
6     return Σ_{j∈π_i} d_j
```

Assuming a topological order is given (otherwise, it can be computed in time linear in the size of the induced graph, or memoization technique can be used), the number of operations required is $O(\sum_i \ell_i)$, the sum of the lengths of the flows (total number of operations of line 5). In the worst case, $\sum_i \ell_i = nm$, but is usually much less. The bound computed will be very loose: first, it is computed as the sum of the upper delay bounds of each server, without taking into account the Pay-burst-only-once principle. Second, the residual service curve computed is a pure delay (as in Theorem 3), which introduces some pessimism in the computation of the arrival curve of the departure processes.

TFA++ is similar to TFA except that it takes into account the shaping rate of the greedy-shaper of the preceding servers. It has first been introduced in Grieux's PhD thesis [28] and then popularized by Mifdaoui and Leydier under the name TFA++ in [11]. In short, between Algorithm 1 and 2, lines 3 and 4 differ. Under the technical assumption of [11] on the rates of the shapers, the complexity of computing delay d_j is now linear in the number of predecessors of server j, but the time complexity of the algorithms remains $O(\sum_i \ell_i)$. As we will see in Section 7, the delay bounds are drastically improved with TFA++ and can be quite accurate in some cases.

The case with cyclic dependencies is studied in [10], and will be commented in more details in Section 6.

Algorithm 2: TFA++ analysis: delay of flow f_i

```
1 begin
2     foreach server j in the topological order do
3         α ← Σ_{h∈prec(j)} min(σ_h, Σ_{i∈Fl(h,j)} α_i^(j)) + Σ_{i | π_i(1)=j} α_i;
4         d_j ← d_max(α, β_j);
5         foreach flow f_i ∈ Fl(j) do b_i^(succ_i(j)) ← b_i^(j) + r_i d_j;
6     return Σ_{j∈π_i} d_j
```

3.2 SFA (Separated flow analysis)

SFA is the technique that exploits the Pay-burst-only-once principle through the use of (min, plus) operators. We give here a possible algorithm when the network is FIFO, by choosing a particular value of θ in Theorem 1. This choice is locally optimal: from Theorem 4, the backlog bound characterizes the maximum burst of the output arrival curves, so θ is chosen to minimize the backlog bound for each flow at each server.

Corollary 1 (of Theorem 1)**.** *Consider a FIFO server with service curve $\beta = \beta_{R,T}$, crossed by two flows f_1 and f_2 with respective arrival curves $\alpha_1 = \gamma_{b_1,r_1}$ and $\alpha_2 = \gamma_{b_2,r_2}$. A residual service curve for flow f_1 is*

$$\beta' : t \mapsto (R - r_2)(t - (T + b_2/R))_+.$$

The output arrival curve is $\alpha'_1 = \alpha_1 + (T + b_2/R)r_1$.

This is Theorem 1 with $\theta = T + b_2/R$ (see also [18, Corollary 2.6.3]). Note that Boyer et al. in [29] use the SFA method with $\theta = T$. In that case the residual service curve computed has a larger latency: $T + b_2/(R - r_2)$.

Algorithm 3 describes the procedure to compute the delay of a flow with the SFA method, when arrival curves are token-bucket ones and service curves rate-latency ones. In line 4, the aggregate burst of the cross-traffic of flow $f_{i'}$ is computed. A residual curve $\beta_{i'}^{(j)} = \beta_{R_{i'}^{(j)},T_{i'}^{(j)}}$ is computed for each server and each flow crossing it (lines 5 and 6). Then the burst parameter of the output arrival curve is computed (line 7). Finally, line 8 the end-to-end delay is computed. The end-to-end service curve for flow f_i is the convolution of all service curves computed for flow f_i: $\beta_i^{(\pi_i(1))} * \cdots * \beta_i^{(\pi_i(\ell_i))} = \beta_{R,T}$ with $T = \sum_{j\in\pi_i} T_i^{(j)}$ and $R = \min_{j\in\pi_i} R_i^{(j)}$.

Algorithm 3: SFA analysis: delay of flow f_i

```
1 begin
2   foreach server j in the topological order do
3     foreach flow i' ∈ Fl(j) do
4       b ← Σ_{k∈Fl(j)−i'} b_k^(j);
5       T_{i'}^(j) ← (T_j + b/R_j);
6       R_{i'}^(j) ← R_j − Σ_{k∈Fl(j)−i'} r_k;
7       b_{i'}^(succ_{i'}(j)) ← b_{i'}^(j) + T_{i'}^(j) r_{i'};
8   return Σ_{j∈π_i} T_i^(j) + b_i/(min_{j∈π_i} R_i^(j))
```

Given a topological order on the servers, this algorithm requires a constant number of operations per server and flow crossing it (note that the cost of computing b does not have to be paid at each inner loop, as the sum of all bursts arriving at a server can be pre-computed). The time complexity of this algorithm is then also $O(\sum_i \ell_i)$. SFA always compute better performance bounds than TFA (smaller arrival curves for the departure process are computed, and SFA benefits from the Pay-burst-only-once principle). However, the SFA algorithm described here does not take into account the shaping effect, so it is usually outperformed by TFA++. Whereas the combination with greedy shapers is possible, no study in the literature, to our knowledge, takes the shaping of the cross-traffic into account in the SFA analysis of a FIFO network, so we will only compare our work to the SFA algorithm presented in Algorithm 3.

Networks with cyclic dependencies can be analyzed with SFA by applying the fixed-point method [17].

3.3 Deborah

Deborah(DElay BOund Rating AlgoritHm) [30] is a software designed to compute delay bounds in FIFO tandem networks. It is based on the optimization of θ parameters that appear in Theorem 1. For nested networks (the path of each flow is either contained in, contains or is disjoint from the path of any other flow), it defines one θ parameter per flow, and the parameter of one flow depends on the parameters of flows nested in this flow. These parameters are then optimized to compute the delay bound, called LUDB, of a flow. Lenzini et al. showed in [16] that the delay bounds are tight for sink-tree tandems, and more generally for sink-trees (all flows end at the last server). However, Bisti et al. exhibit in [31] a very small example for which LUDB does not achieve tightness for other nested tandems. The general case of tandems can be tackled by cutting the tandem into nested sub-tandems [32, 33]. An additional optimization step is performed to find the optimal decomposition into sub-tandems. In [34], the *single tandem analysis* (STA) is introduced: non-nested tandem are transformed into a nested one by cutting the competing flows but not the flow of interest, resulting in more accurate delay bounds, but a larger optimization problem.

Computing the LUDB in a nested tandem requires optimizing the θ-parameters, which can be done using piecewise-linear programming, when explicit formulas are not derived. The number of decompositions to test in sub-tandems highly depends on the topology. Experimentally, the computation of the LUDB grows exponentially with the size of the tandem. Nevertheless, when non-nested tandem are cut (including the flow of interest), the algorithm becomes scalable.

Deborah requires token-bucket arrival curves and rate-latency service curves, and does not take into account the shaping effect of a maximal service curve.

3.4 Linear programming

The linear programming approach developed in [12, 13] consists in writing the Network Calculus relations as linear constraints. If the arrival curves are piecewise linear and concave and the service curves, piecewise linear and convex, then the exact worst-case performance bounds in feed-forward networks can be computed by a MILP (Mixed-integer linear program). However, this solution is very costly as the number of variables is exponential and there are integer variables. The MILP can be relaxed by removing the integer variables and their corresponding constraints. While this relaxation gives accurate bounds (better than other methods), the number of constraints is still too high to be able to compute bounds in large network. In the following, we call ELP the relaxed linear program.

3.5 Linear-programming and algebraic methods under arbitrary multiplexing

The methods described above have their counterparts under arbitrary multiplexing (no scheduling policy is assumed among the flows crossing the servers, while still assuming per-flow FIFO scheduling). Under arbitrary multiplexing, it was shown that TFA is always worse than SFA (the blind multiplexing version of TFA must replace the worst-case delay of the aggregate server by the length of its backlogged period, TFA++ does not provide any improvement in most cases). In addition to SFA and TFA methods, another bound has been studied, named PMOO[1], was obtained by multi-dimensional (min, plus)-convolution [36, 37, 38]. In short, the PMOO method first reduces the analysis of a tandem network to the analysis of a single end-to-end 1-server,

[1]For nested tandem, the LUDB analysis, before the optimization of the parameters is similar to a PMOO analysis. For more general topologies (non-nested tandems), there is currently no equivalent to the PMOO method for FIFO scheduling. The formula given in [35] has been proved incorrect in [31].

whose characteristics depend on the servers and of the competing flows. Then the computation of the delay or the backlog is done using this 1-server. Schmitt et al. showed in [39] that SFA and PMOO cannot be compared (there are networks where one method is better that the other and conversely).

A first step into the computation of tight bounds in feed-forward network was obtained by using linear programming [40]. It was shown that, under technical assumptions, computing tight bounds in tandem networks is polynomial, while it is a NP-hard problem in general feed-forward networks. Following this first result, explicit tight delay and backlog bounds for tree networks have been derived in [41, 20] for networks with token-bucket arrival curves and rate-latency service curves. The time complexity is $O(n^2 + m)$.

Because the general problem is NP-hard, the linear-programming bounds cannot be used directly for analyzing general feed-forward networks. Another thread of research consists in using the algebraic methods (e.g., SFA and PMOO) smartly to approach the exact worst-case bound. In [42], Bondorf et al. backtrack in the network to test every (min, plus)-decomposition of the network using flow aggregation [43, 44] to better bound the cross-traffic in addition to PMOO and SFA. The delays obtained are quasi-tight. While the time complexity of this approach is exponential for tandem networks, it scales well on the tested networks.

We will compare our contribution with the FIFO methods described in this section. From the point of view of the complexity, we call *scalable* the methods that have a low time complexity, as TFA++ and SFA: they can be used to analyze large networks (at least a few hundred servers). On the contrary, we call *untractable* the methods that have a (super)-exponential complexity, such as the linear programming methods for FIFO networks (including ELP). They can only be used to analyze network with a few nodes (less than 10 servers) in reasonable time (a few minutes). We are now interested in in-between methods: they might not scale as well as scalable methods but can be used to analyze networks with tens of servers and lead to better performance bounds than the scalable methods. Basically, their complexity is polynomial. In the rest of the paper, we call these methods *tractable*. The term *performance* or *performance bound* always refers to the end-to-end delay or backlog of a flow. The term *accuracy* will be used to define the quality or tightness of this bounds and the terms *scalability* or *tractability* will refer to the algorithmic complexity or execution time of the methods.

4 A polynomial-size linear program for tree networks

In this section, we propose an alternative to the linear program of [12] that keeps the number of constraints and variables polynomial in the size of the network for a tree network. Simply removing constraints from the linear program proposed in [12] can make the bounds more pessimistic than SFA or TFA. Therefore, we propose to incorporate these latter bounds to improve the tightness of our new proposal. We also adapt the linear program so that it can take into account the shaping of the cumulative processes due to the link capacities. We first describe the linear program and prove that its optimal solution is an upper bound of the performance in Paragraph 4.1, and then comment it in Paragraph 4.2. In particular, we compare it with the previous MILP proposed in [12].

4.1 A linear program to compute upper performance bounds

In this paragraph, we describe a new linear proposal to compute performance upper bounds.

To give the intuition of these variables and constraints, we apply the construction on the small network for Figure 6. The linear program is given in Table 1, and the schematic view of the constraints in Figure 7.

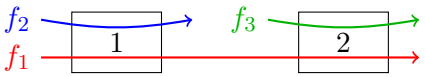


Figure 6: Toy network for the linear program description.

For the general description of the linear program, we assume that the network is a tree, and that we are computing the performance of a flow of interest exiting at the root of the tree (by considering the sub-network induced by this flow). The *bit of interest* is the bit of data we are computing the performance of.

As the network is a tree, each server j, except the root, has a unique successor that we denote $\mathrm{succ}(j)$. We also set $\mathrm{succ}(n) = n+1$. In the example of Figure 6, $\mathrm{succ}(1) = 2$ and $\mathrm{succ}(2) = 3$. To simplify the notations, we will use $F_i^{(\mathrm{succ}(\pi_i(\ell_i)))}$ instead of $F_i^{(n+1)}$ for the departure processes of the flows. Let us denote $d(j)$ the depth of server j. We set the depth of the root $d(n) = 1$, and $d(j) = d(\mathrm{succ}(j)) + 1$.

We now describe the linear program. In the following, we write the variables in bold letters to differentiate them from times and cumulative processes.

Variables

- *Time variables:* we introduce one variable $\mathbf{t}_{(n+1,0)}$ representing the departure time of the bit of interest. For each server j, there are $d(j)+1$ variables $\mathbf{t}_{(j,k)}$ for $k \in \{0, \dots, d(j)\}$. Intuitively, $\mathbf{t}_{(j,k)}$ corresponds to a time when arrival processes at server j will be evaluated in the linear program;

- *Process variables:* we introduce variables $\mathbf{F}_i^{(j)}\mathbf{t}_{(j,k)}$ for all $i \in \mathbb{N}_m$, $j \in \pi_i \cup \{\mathrm{succ}(\pi_i(\ell_i))\}$ and $k \in \{0, \dots, d(j)\}$. Intuitively, $\mathbf{F}_i^{(j)}\mathbf{t}_{(j,k)}$ corresponds to the values of the arrival cumulative processes of flow f_i at server j and at time $t_{(j,k)}$.

Example 1. *In our example, we have 6 time variables:* $\mathbf{t}_{(3,0)}$ *for the departure time of the bit of interest,* $\mathbf{t}_{(2,0)}$ *and* $\mathbf{t}_{(2,1)}$ *are defined for arrivals at server 2, and* $\mathbf{t}_{(1,0)}$, $\mathbf{t}_{(1,1)}$, $\mathbf{t}_{(1,2)}$ *are defined for arrivals at server 1.*

Constraints In the constraints given below, server h is the successor of server j ($h = \mathrm{succ}(j)$).

- *Time constraints:*
  - $\forall j \in \mathbb{N}_n,\ \forall k \in \{0, \dots, d(j)-1\},\ \mathbf{t}_{(j,k)} \geq \mathbf{t}_{(j,k+1)}$;
  - $\forall j \in \mathbb{N}_n,\ \forall k \in \{0, \dots, d(j)\},\ \mathbf{t}_{(j,k)} \leq \mathbf{t}_{(h,k)}$;

- *FIFO constraints:*
  - $\forall j \in \mathbb{N}_n,\ \forall k \in \{0, \dots, d(h)\},\ \forall i \in \mathrm{Fl}(j),\ \mathbf{F}_i^{(j)}\mathbf{t}_{(j,k)} = \mathbf{F}_i^{(h)}\mathbf{t}_{(h,k)}$;

- *Service constraints:*
  - $\forall j \in \mathbb{N}_n,\ \sum_{i \in \mathrm{Fl}(j)} \mathbf{F}_i^{(h)}\mathbf{t}_{(h,d(h))} \geq \sum_{i \in \mathrm{Fl}(j)} \mathbf{F}_i^{(j)}\mathbf{t}_{(j,d(j))}$;
  - $\forall j \in \mathbb{N}_n,\ \sum_{i \in \mathrm{Fl}(j)} \mathbf{F}_i^{(h)}\mathbf{t}_{(h,d(h))} \geq \sum_{i \in \mathrm{Fl}(j)} \mathbf{F}_i^{(j)}\mathbf{t}_{(j,d(j))} + R_j(\mathbf{t}_{(h,d(h))} - \mathbf{t}_{(j,d(j))}) - R_j T_j$;

- *TFA++ constraints:* We denote by d_j^{TFA} the delay of server j computed in Algorithm 2, line 4.

- $\forall j \in \mathbb{N}_n,\ \forall k \in \{0, \ldots, d(h)\},\ \mathbf{t}_{(h,k)} - \mathbf{t}_{(j,k)} \leq d_j^{TFA}$;

- *SFA constraints:* We denote by d_i^{SFA} the delay computed with Algorithm 3, with $j = \pi_i(1)$ and $h' = \text{succ}(\pi_i(\ell_i))$,
  - $\forall i \in \mathbb{N}_m,\ \forall k \in \{0, \ldots, d(h')\},\ \mathbf{t}_{(h',k)} - \mathbf{t}_{(j,k)} \leq d_i^{SFA}$;

- *Arrival constraints:* with $j = \pi_i(1)$,
  - $\forall i \in \mathbb{N}_m,\ \forall 0 \leq k < k' \leq d(j),\ \mathbf{F}_i^{(j)}\mathbf{t}_{(j,k)} - \mathbf{F}_i^{(j)}\mathbf{t}_{(j,k')} \leq b_i + r_i(\mathbf{t}_{(j,k)} - \mathbf{t}_{(j,k')})$;

- *Shaping constraints:*
  - $\forall j \in \mathbb{N}_n,\ \forall 0 \leq k < k' \leq d(h), \sum_{i \in \text{Fl}(j,h)}(\mathbf{F}_i^{(h)}\mathbf{t}_{(h,k)} - \mathbf{F}_i^{(h)}\mathbf{t}_{(h,k')}) \leq L_j + C_j(\mathbf{t}_{(h,k)} - \mathbf{t}_{(h,k')})$;

- *Monotonicity constraints:* with $j = \pi_i(1)$,
  - $\forall i \in \mathbb{N}_m,\ \forall k \in \{0, \ldots, d(j) - 1\},\ \mathbf{F}_i^{(j)}\mathbf{t}_{(j,k)} \geq \mathbf{F}_i^{(j)}\mathbf{t}_{(j,k+1)}$.

Objectives: Several objectives can be defined depending on the performance to compute. For example, in the rest of the paper, we will use:

- *delay objective:* To compute the an upper bound of the delay of flow f_i, ending at server n,
  - $\max : \mathbf{t}_{(n+1,0)} - \mathbf{t}_{(\pi_i(1),0)}$.

- *backlog objective (or maximum amount of data of flow f_i in transit at any time):* Alternatively, to obtain an upper bound of the worst-case end-to-end backlog of flow f_i starting at server j and ending at server n, one introduces the following constraints and objective:
  - $\forall k \in \{0, \ldots, d(j)\}, \mathbf{F}_i^{(j)}\mathbf{t}_{(n+1,0)} - \mathbf{F}_i^{(j)}\mathbf{t}_{(j,k)} \leq b_i + r_i(\mathbf{t}_{(n+1,0)} - \mathbf{t}_{(j,k)})$;
  - $\max\ :\ \mathbf{F}_i^{(j)}\mathbf{t}_{(n+1,0)} - \mathbf{F}_i^{(n+1)}\mathbf{t}_{(n+1,0)}$.

Number of variables and constraints The total number of time variables is at most $(n+2)(n+1)/2$ (the worst-case is obtained for trees of maximal depth, that is tandem networks), and for each flow, there is at most one process variable per time variable, so the number of process variables is at most $m(n+2)(n+1)/2$ (this number is reached when all flows cross all servers). In total, the number of variables is $O(mn^2)$.

Similarly, there are at most $n(n+1)$ time constraints, $mn(n-1)/2$ FIFO constraints, $2n$ service constraints, $mn(n+1)/2$ TFA++ constraints, mn SFA constraints, $mn(n+1)/2$ arrival constraints, $n^2(n+1)/2$ shaping constraints and $mn(n+1)/2$ monotonicity constraints. The additional constraints for computing the backlog is at most n. In total, the number of constraints is $O(mn^3)$.

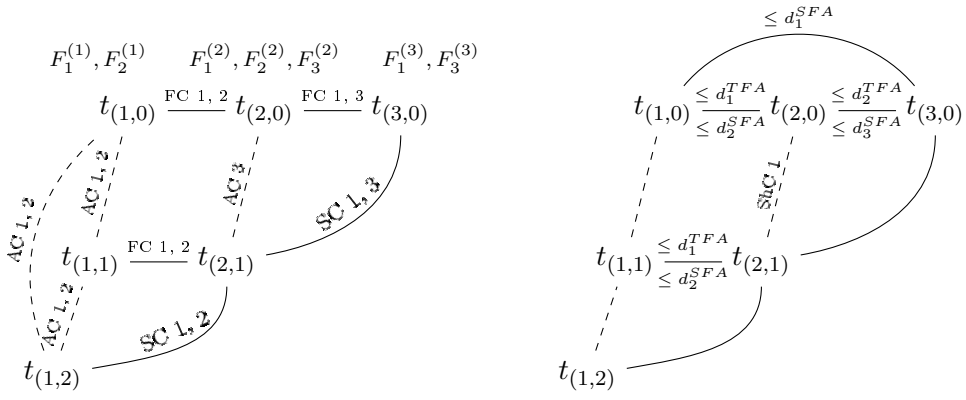
The number of variables and constraints is then polynomial in the size of the network.

Theorem 5. *1. Let D be an optimal solution of the linear program described above with the delay objective and d be the worst-case delay of the flow of interest. Then $D \geq d$.*

2. Let B be an optimal solution of the linear program described with the backlog objective and b be the worst-case backlog of the flow of interest. Then $B \geq b$.

Maximize:	$\mathbf{t}_{(3,0)} - \mathbf{t}_{(1,0)}$
such that (Time constraints)	$\mathbf{t}_{(1,0)} \leq \mathbf{t}_{(2,0)} \leq \mathbf{t}_{(3,0)}$ $\mathbf{t}_{(1,1)} \leq \mathbf{t}_{(2,1)}$ $\mathbf{t}_{(2,1)} \leq \mathbf{t}_{(2,0)}$ $\mathbf{t}_{(1,2)} \leq \mathbf{t}_{(1,1)} \leq \mathbf{t}_{(1,0)}$
(FIFO constraints)	$\mathbf{F}_1^{(1)}\mathbf{t}_{(1,0)} = \mathbf{F}_1^{(2)}\mathbf{t}_{(2,0)} = \mathbf{F}_1^{(3)}\mathbf{t}_{(3,0)}$ $\mathbf{F}_1^{(1)}\mathbf{t}_{(1,1)} = \mathbf{F}_1^{(2)}\mathbf{t}_{(2,1)}$ $\mathbf{F}_2^{(1)}\mathbf{t}_{(1,0)} = \mathbf{F}_2^{(2)}\mathbf{t}_{(2,0)}$ $\mathbf{F}_2^{(1)}\mathbf{t}_{(1,1)} = \mathbf{F}_2^{(2)}\mathbf{t}_{(2,1)}$ $\mathbf{F}_3^{(2)}\mathbf{t}_{(2,0)} = \mathbf{F}_3^{(3)}\mathbf{t}_{(3,0)}$
(Service constraints)	$\mathbf{F}_1^{(2)}\mathbf{t}_{(2,1)} + \mathbf{F}_2^{(2)}\mathbf{t}_{(2,1)} \geq \mathbf{F}_1^{(1)}\mathbf{t}_{(1,2)} + \mathbf{F}_2^{(1)}\mathbf{t}_{(1,2)} + R_1(\mathbf{t}_{(2,1)} - \mathbf{t}_{(1,2)}) - R_1T_1$ $\mathbf{F}_1^{(2)}\mathbf{t}_{(2,1)} + \mathbf{F}_2^{(2)}\mathbf{t}_{(2,1)} \geq \mathbf{F}_1^{(1)}\mathbf{t}_{(1,2)} + \mathbf{F}_2^{(1)}\mathbf{t}_{(1,2)}$ $\mathbf{F}_1^{(3)}\mathbf{t}_{(3,0)} + \mathbf{F}_3^{(3)}\mathbf{t}_{(3,0)} \geq \mathbf{F}_1^{(2)}\mathbf{t}_{(2,1)} + \mathbf{F}_3^{(2)}\mathbf{t}_{(2,1)} + R_2(\mathbf{t}_{(3,0)} - \mathbf{t}_{(2,1)}) - R_2T_2$ $\mathbf{F}_1^{(3)}\mathbf{t}_{(3,0)} + \mathbf{F}_3^{(3)}\mathbf{t}_{(3,0)} \geq \mathbf{F}_1^{(2)}\mathbf{t}_{(2,1)} + \mathbf{F}_3^{(2)}\mathbf{t}_{(2,1)}$
(TFA++ constraints)	$\mathbf{t}_{(2,0)} - \mathbf{t}_{(1,0)} \leq d_1^{TFA}$ $\mathbf{t}_{(2,1)} - \mathbf{t}_{(1,1)} \leq d_1^{TFA}$ $\mathbf{t}_{(3,0)} - \mathbf{t}_{(2,0)} \leq d_2^{TFA}$
(SFA constraints)	$\mathbf{t}_{(3,0)} - \mathbf{t}_{(1,0)} \leq d_0^{SFA}$ $\mathbf{t}_{(2,0)} - \mathbf{t}_{(1,0)} \leq d_1^{SFA}$ $\mathbf{t}_{(2,1)} - \mathbf{t}_{(1,1)} \leq d_1^{SFA}$ $\mathbf{t}_{(3,0)} - \mathbf{t}_{(2,0)} \leq d_2^{SFA}$
(Greedy-shaper constraints)	$\mathbf{F}_1^{(1)}\mathbf{t}_{(2,0)} - \mathbf{F}_1^{(1)}\mathbf{t}_{(2,1)} \leq L_1 + C_1(\mathbf{t}_{(2,0)} - \mathbf{t}_{(2,1)})$
(Arrival constraints)	$\mathbf{F}_1^{(1)}\mathbf{t}_{(1,0)} - \mathbf{F}_1^{(1)}\mathbf{t}_{(1,1)} \leq b_1 + r_1(\mathbf{t}_{(1,0)} - \mathbf{t}_{(1,1)})$ $\mathbf{F}_1^{(1)}\mathbf{t}_{(1,1)} - \mathbf{F}_1^{(1)}\mathbf{t}_{(1,2)} \leq b_1 + r_1(\mathbf{t}_{(1,1)} - \mathbf{t}_{(1,2)})$ $\mathbf{F}_1^{(1)}\mathbf{t}_{(1,0)} - \mathbf{F}_1^{(1)}\mathbf{t}_{(1,2)} \leq b_1 + r_1(\mathbf{t}_{(1,0)} - \mathbf{t}_{(1,2)})$ $\mathbf{F}_2^{(1)}\mathbf{t}_{(1,0)} - \mathbf{F}_2^{(1)}\mathbf{t}_{(1,1)} \leq b_2 + r_2(\mathbf{t}_{(1,0)} - \mathbf{t}_{(1,1)})$ $\mathbf{F}_2^{(1)}\mathbf{t}_{(1,1)} - \mathbf{F}_2^{(1)}\mathbf{t}_{(1,2)} \leq b_2 + r_2(\mathbf{t}_{(1,1)} - \mathbf{t}_{(1,2)})$ $\mathbf{F}_2^{(1)}\mathbf{t}_{(1,0)} - \mathbf{F}_2^{(1)}\mathbf{t}_{(1,2)} \leq b_2 + r_2(\mathbf{t}_{(1,0)} - \mathbf{t}_{(1,2)})$ $\mathbf{F}_3^{(2)}\mathbf{t}_{(2,0)} - \mathbf{F}_3^{(2)}\mathbf{t}_{(2,1)} \leq b_3 + r_3(\mathbf{t}_{(2,0)} - \mathbf{t}_{(2,1)})$
(Monotonicity constraints)	$\mathbf{F}_1^{(1)}\mathbf{t}_{(1,0)} \geq \mathbf{F}_1^{(1)}\mathbf{t}_{(1,1)} \geq \mathbf{F}_1^{(1)}\mathbf{t}_{(1,2)}$ $\mathbf{F}_2^{(1)}\mathbf{t}_{(1,0)} \geq \mathbf{F}_2^{(1)}\mathbf{t}_{(1,1)} \geq \mathbf{F}_2^{(1)}\mathbf{t}_{(1,2)}$ $\mathbf{F}_3^{(2)}\mathbf{t}_{(2,0)} \geq \mathbf{F}_3^{(2)}\mathbf{t}_{(2,1)}$

Table 1: Linear program for the toy example of Figure 6.



(a) FIFO, service and arrival constraints (b) TFA++, SFA and shaping constraints

Figure 7: Schematic view of the constraints for the linear program to analyze network of Figure 6. Times are ordered by solid or dashed lines from bottom-left to top-right. (a) the processes attached to the times are noted above. FIFO (FC) and service (SC) constraints with the involved flows are in solid lines, and arrival constraints (AC) in dashed lines. (b) TFA++ and SFA constraints (solid lines) impose upper bounds between dates involved in the FIFO constraints, and shaping constraints (ShC) in dashed lines are used at the output of server 1.

Proof. The proof is similar to the proof in [12, 13] (upper bound part). We prove only the first statement (delay bound). The proof of the end-to-end backlog is similar.

Let $(F_i^{(j)})_{i,j}$ be an admissible trajectory for the network, and assume without loss of generality that f_1 is the flow of interest. Let $t_{(n+1,0)}$ be the departure time (at server n) of a bit of interest.

The first step of the proof is to assign values to the time and process variables and the second step is to prove that this assignment satisfies all the linear constraints.

To set the variables, we proceed by a backward induction on the servers and build times $(t_{(h,k)})_{k\in\{0,\dots,d(h)\}}$ and assignment of the variables satisfying:

- for all $k \in \{0,\dots,d(h)\}$, $\mathbf{t}_{(h,k)} = t_{(h,k)}$;
- for all $k \in \{0,\dots,d(h)-1\}$, $F_i^{(h)}(t_{(h,k)}) \leq \mathbf{F}_i^{(h)}\mathbf{t}_{(h,k)} \leq F_i^{(h)}(t_{(h,k)}^+)$;
- $\mathbf{F}_i^{(h)}\mathbf{t}_{(h,d(h))} = F_i^{(h)}(t_{(h,d(h))})$,

with the right-limit notation $f(t^+) = \lim_{s\to t, s>t} f(s)$.

For the initialization, $h = n+1$, we set $\mathbf{t}_{(n+1,0)} = t_{(n+1,0)}$ and for all $i \in \mathrm{Fl}(n)$, $\mathbf{F}_i^{(n+1)}\mathbf{t}_{(n+1,0)} = F_i^{(n+1)}(t_{(n+1,0)})$.

Assume the assignment is done for server h, and consider server j such that $\mathrm{succ}(j) = h$.

Let us assign the time variables of server j: for all $k \in \{0,\dots,d(h)\}$, there exists s such that $\forall i \in \mathrm{Fl}(j)$, $F_i^{(j)}(s) \leq \mathbf{F}_i^{(h)}\mathbf{t}_{(h,k)} \leq F_i^{(j)}(s^+)$. Set $t_{(j,k)}$ to the maximal value of such s, assign $\mathbf{t}_{(j,k)} = t_{(j,k)}$ and $\mathbf{F}_i^{(j)}\mathbf{t}_{(j,k)} = \mathbf{F}_i^{(h)}\mathbf{t}_{(h,k)}$.

There exists s such that

$$\sum_{i\in \mathrm{Fl}(j)} F_i^{(h)}(t_{(h,d(h))}) \geq [(\sum_{i\in \mathrm{Fl}(j)} F_i^{(j)}) * \beta_j](t_{(h,d(h))}) \\ = \sum_{i\in \mathrm{Fl}(j)} F_i^{(j)}(s) + \beta_j(t_{(h,d(h))} - s).$$

As $\mathbf{F}_i^{(h)}\mathbf{t}_{(h,d(h))} = F_i^{(h)}(t_{(h,d(h))})$, set $t_{(j,d(j))} = s$ and assign $\mathbf{t}_{(j,d(j))} = t_{(j,d(j))}$ and $\mathbf{F}_i^{(j)}\mathbf{t}_{(j,d(j))} = F_i^{(j)}(t_{(j,d(j))})$.

For flows f_i that do not cross server j but for which $F_i^{(j)}$ is defined (the last server they cross is a predecessor of server j), we assign the variables as $\mathbf{F}_i^{(j)}\mathbf{t}_{(j,k)} = F_i^{(j)}(t_{(j,k)})$.

The assignment of variables corresponding to server j is then done according to the desired properties.

Note that by transitivity,

$$F_1^{(n+1)}(t_{(n+1,0)}) = \mathbf{F}_1^{(n+1)}\mathbf{t}_{(n+1,0)} = \mathbf{F}_1^{(\pi_1(1))}\mathbf{t}_{(\pi_1(1),0)} \\ \in [F_1^{(\pi_1(1))}(t_{(\pi_1(1),0)}), F_1^{(\pi_1(1))}(t^+_{(\pi_1(1),0)})],$$

and for all $s \leq t_{(\pi_1(1),0)}$, $F_1^{(n+1)}(t_{(n+1,0)}) \geq F_1^{(\pi_1(1))}(s)$, so from the definition of the delay, $\mathbf{t}_{(n+1,0)} - \mathbf{t}_{(\pi_1(1),0)}$ is at least the delay experienced by the bit of interest.

The second step is to check that the variables thus assigned satisfy all the linear constraints of the linear program. By construction, the FIFO and service constraints are satisfied.

As the system is causal, that is, $F_i^{(j)} \geq F_i^{(\mathrm{succ}(j))}$, and the cumulative processes are non-decreasing, the time and monotonicity constraints are satisfied.

For all $i \in \mathbb{N}_m$, let us denote $j = \pi_i(1)$ the first server crossed by flow f_i, for all $0 \leq k < k' \leq d(j)$, $\mathbf{F}_i^{(j)}\mathbf{t}_{(j,k)} - \mathbf{F}_i^{(j)}\mathbf{t}_{(j,k')} \leq F_i^{(j)}(t^+_{(j,k)}) - F_i^{(j)}(t_{(j,k')}) \leq b_i + r_i(t_{(j,k)} - t_{(j,k')})$. The arrival constraints are then satisfied.

Similarly, consider server j and its departure processes $F_i^{(h)}$. For all $0 \leq k < k' \leq d(h)$,

$$\sum_{i\in \mathrm{Fl}(j)} (\mathbf{F}_i^{(h)}\mathbf{t}_{(h,k)} - \mathbf{F}_i^{(h)}\mathbf{t}_{(h,k')}) \leq \sum_{i\in \mathrm{Fl}(j)} (F_i^{(h)}(t^+_{(h,k)}) - F_i^{(h)}(t_{(h,k')})) \\ \leq L_j + C_j(t_{(h,k)} - t_{(h,k')}),$$

and the shaping constraints are satisfied.

Let us focus on the TFA++ constraints. For each FIFO constraint $\mathbf{F}_i^{(j)}\mathbf{t}_{(j,k)} = \mathbf{F}_i^{(h)}\mathbf{t}_{(h,k)}$, we have

$$F_i^{(j)}(t_{(j,k)}) \leq \mathbf{F}_i^{(j)}\mathbf{t}_{(j,k)} = \mathbf{F}_i^{(h)}\mathbf{t}_{(h,k)} \leq F_i^{(h)}(t^+_{(h,k)}),$$

so $t_{(h,k)} - t_{(j,k)} \leq d_j^{TFA}$ and the constraint $\mathbf{t}_{(h,k)} - \mathbf{t}_{(j,k)} \leq d_j^{TFA}$ is satisfied.

Similarly, for each flow f_i, let j be the first server it crosses and h' be the successor of the last server it crosses. For all $t_{(h',k)}$ where $F_i^{(h')}$ is defined, and $t_{(j,k)}$ such that $\mathbf{F}_i^{(j)}\mathbf{t}_{(j,k)} = \mathbf{F}_i^{(h')}\mathbf{t}_{(h',k)}$ (by transitivity) we have

$$F_i^{(j)}(t_{(j,k)}) \leq \mathbf{F}_i^{(j)}\mathbf{t}_{(j,k)} = \mathbf{F}_i^{(h')}\mathbf{t}_{(h',k)} \leq F_i^{(h')}(t^+_{(h',k)}),$$

so $t_{(h',k)} - t_{(j,k)} \leq d_i^{SFA}$ and then the constraint $\mathbf{t}_{(h',k)} - \mathbf{t}_{(j,k)} \leq d_i^{SFA}$ is satisfied.

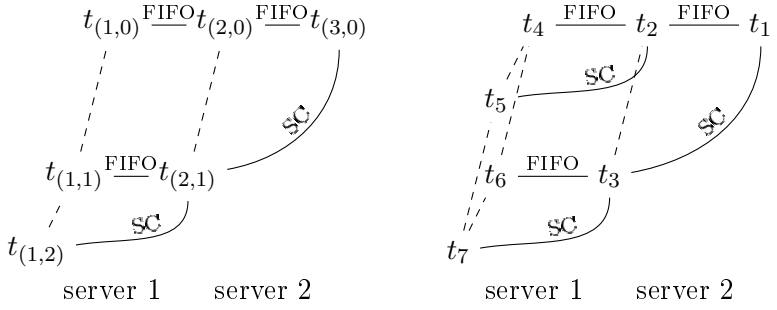
If d is the delay of the bit of interest and D the maximal solution of the linear program, we have $d \leq D$. This is valid for all bits of data of flow f_1, which finished the proof. □ □

4.2 Comments and comparison with [12, 13]

The proposed solution is mostly inspired from [12] and [13]. The time, FIFO, service, arrival and monotonicity constraints are of the same type.

A simplification of [13] The first difference with [13] is that we introduce fewer dates at each server: instead of introducing 2^k time variables for nodes at depth k, we only have $k+1$ times variables. The reason is that for each server, we introduce only one service relation (resulting in two linear constraints to take into account the rate-latency shape of the service curve).

The simplification for the network of Figure 6 is depicted in Figure 8.



(a) New linear program (b) Linear program of [12]

Figure 8: Time variables for the linear program to analyze network of Figure 6. Times are ordered from bottom-left to top-right for the FIFO and service (SC) constraints (solid line). These orders induce an order on the dates corresponding to each server (dashed lines). In (a), only 3 time variables are introduced for server 2, and they are totally ordered. On the contrary, in [12], t_5 and t_6 can not be ordered directly. Either Boolean variables have to be introduced to ensure the monotonicity or the processes relaxed.

TFA++ and SFA constraints As we will see in Example 2, only removing variables does not lead to an acceptable solution, as the performance bounds would be larger than those obtained with SFA or TFA++. To overcome this issue, we introduced SFA and TFA++ constraints, a second difference with [12]. The idea here is to reintroduce the missing service-curve constraints, using instead pure-delay service curves, as per Theorem 3. Indeed, if d_j^{TFA} is a delay bound for server j, then $\delta_{d_j^{TFA}}$ is a service curve for server j. Applying this service curve at time $t_{(\mathrm{succ}(j),k)}$ does not require the introduction of a new time variable: $\mathbf{t}_{(j,k)}$ can be used.

Shaping constraints The third difference is the introduction of shaping constraints. This is a reinforcement of the model, and these type of constraints can also be added to the linear programs of [12, 13] if information about the shaping rate is available, as well as to the linear programs used to compute performance bounds for other scheduling policies (arbitrary multiplexing in [40] and priorities in [45]). We will see in Example 2 and later in Section 7 that these constraints are not very useful for tandem networks, but much more useful for non-tandem trees.

Example 2. *Consider the example of Figure 6, with arrival curves $\alpha : t \mapsto 1 + t$ for all flows and service curves $\beta : t \mapsto 4(t-1)_+$ for both servers. We will consider two cases: in the first case, server 1 is not a greedy-shaper, and in the second case server 1 is also a greedy-shaper with $\sigma_1 : t \mapsto 4t$. In this example, we want to illustrate the usefulness of the SFA and TFA++ constraints in both cases. We call PLP the method described in this section, PLP' this method*

when the TFA++ and SFA constraints are removed, ELP the exponential-size linear program described in [12], without the Boolean variables. The delays are given in Table 2.

Method	TFA++	SFA	ELP	PLP'	PLP
delay without σ_1	3.38	2.83	2.81	3.25	2.81
delay with σ_1	2.95	2.83	2.81	3.25	2.81

Table 2: Comparison of the delay bounds of flow f_0 of the toy example of Figure 6. Note that without σ_1 the TFA++ bounds and constraints are TFA bounds and constraints.

One can first notice that the delay bounds are improved when introducing the shaping constraints only for the TFA++ method (which was designed for that purpose). An explanation is that, in this case, the shaping will not improve the worst-case delay bound: it has been showed in [12, Lemma 4] that the worst-case delay in nested tandems is obtained when the service is exact *(the rightmost inequality in Equation* (1) *is an equality). The improvement of the delay bound from PLP' to PLP is then not due to the shaping constraints, but on the SFA and TFA++ constraints. Counter-intuitively, a closer look at the linear programs shows that the improvement in that case is due to the TFA++ constraints (of the first server).*

Figure 9 shows the trajectories computed by the linear program PLP' without σ_1 and provides an explanation: first, if PLP' is used, the time variable $t_{(2,0)}$, used to describe the flows entering the second server appears only as a FIFO constraints in server 1, and is not involved in a service constraint. In this linear program, $t_{(2,0)}$ is set to $t_{(3,0)}$, inducing a larger delay: all data of flow f_3 have been served before serving flow f_1, as if server 2 gave the priority to flow f_3. When adding the TFA++ constraint between times $t_{(1,0)}$ and $t_{(2,0)}$, we enforce that $t_{(2,0)} \leq 1.5$, and then $t_{(3,0)} = t_{(2,0)}$ does not maximize the delay anymore, and the FIFO policy is enforced in server 2.

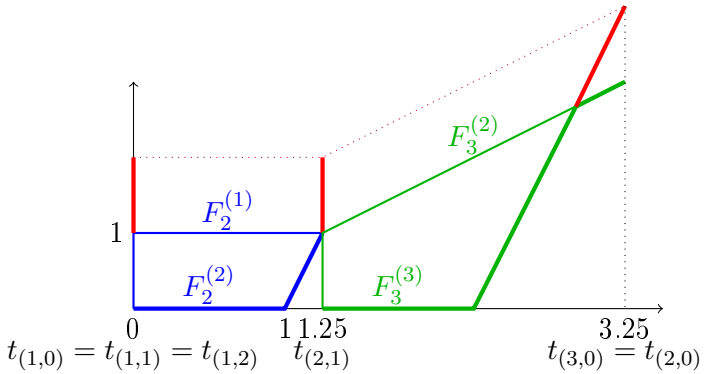


Figure 9: Trajectory reconstructed from the toy example without shaping. (blue) cumulative processes of flow f_2 at server 1; (green) cumulative processes of flow f_3 at server 2;(red) cumulative addition of the processes of flow f_1. At time 0, the burst of size 1 arrives. It is transmitted at time 1.25, and served until time 3.25.

A lower bound of the worst-case performance In [12], a lower bound on the worst-case delay was also computed. It was done by adding more constraints on the linear program (equality of some time variables, $t_5 = t_7$ in Figure 8(b)), and resulted in a polynomial-size linear program. One can notice that there is a correspondence between the time variables of the linear program of our new linear program and the linear program to compute the lower bound.

Note that when adding shaping constraints to the MILP of [12], the tightness of the bounds in this case is not proved, and as a consequence the lower bound might not be valid anymore. An exception is for nested tandems, for which it was proved that a worst-case scenario can be obtained for exact servers. In particular, the departure process of each server is shaped by the service rate.

Tandem vs. tree topologies In this section, we presented the linear program for tree networks, and not for tandems or directly for feed-forward networks. This choice aims at emphasizing that the complexity gap happens between the tree networks and feed-forward networks. Whereas most publications dealing with simple topologies focus on tandem networks or sink-tree networks, the choice here is to present the linear program for tree topologies (that encompass both of them). Indeed, for linear programming approaches, solving the problem for tree network has the same order of complexity than for tandem networks. For example, in [40, Theorem 2], the number of time variables to introduce is the number of paths ending at the root, which is exactly the number of servers for a tree network. Moreover, the subset of times that need to be totally ordered are all naturally ordered by the start of backlogged period relation. Therefore, finding the tight performance bound is a polynomial problem. Another example, in [13], the main step in the generalization of the linear programming approach from tandem networks to feed-forward networks is that at each server, time variables corresponding to that server are deduced from the time variables for each successor, leading to a double exponentiation of the number of time variables. For tree networks, as each server only has one successor, the number of time variables remains exponential. Similar to these examples, the linear problem we propose here has a polynomial size for tree networks. In Section 5, we will see that it can have an exponential size for feed-forward networks.

Generalization to piecewise linear arrival and service constraints Like [12] this linear programming approach remains valid when arrival curves are piecewise linear and concave, and the service curves are piecewise linear and convex. Each arrival curve is then the minimum of token-bucket arrival curves and constraints have now to be written for each token-bucket curve. Similarly, piecewise linear and convex service curves are the maximum of rate-latency curves, and constraints have to be written for each rate-latency curve. The number of variables remains the same, and the number of constraints remains polynomial, provided that the total number of linear pieces is polynomial.

5 Linear programs for feed-forward networks

The linear program proposed in the previous section strongly relies on the tree topology of the network. In this section, we show how to extend the linear programming approach to feed-forward networks. The first method, presented in Paragraph 5.1, is the more accurate and consists in unfolding the network in order to transform it into a tree. This construction is an alternative interpretation of the linear programming for feed-forward network from [13] applied to our simplified linear program of previous section. Unfortunately, the size of the constructed tree might become exponential compared to the size of the original network. We then propose in Paragraph 5.2 an alternative construction to reduce this complexity. For example the decomposition of the network into smaller pieces. These two constructions can of course be combined to optimize the trade-offs between accuracy and tractability. This will be briefly discussed in Paragraph 5.3.

5.1 Unfolding a feed-forward network into a tree network

Intuitively, this is equivalent to introducing FIFO and service time variables independently for each predecessor of servers. Therefore, if a server has two successors, FIFO and service constraints for this server be will be introduced twice and independently. The unfolding of the network of Figure 10 is depicted in Figure 11.

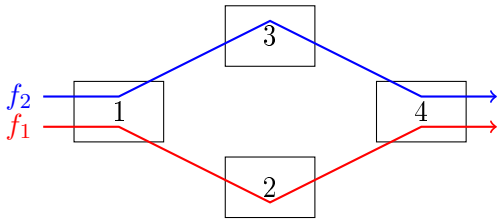


Figure 10: Toy feed-forward network.

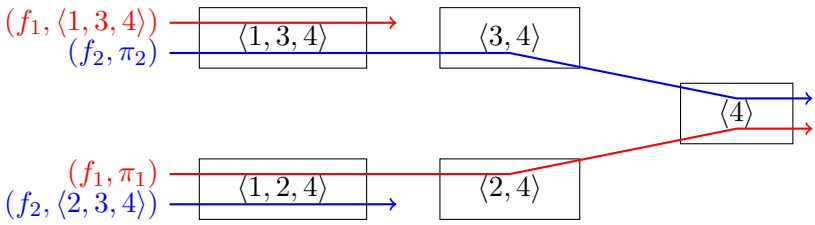


Figure 11: Unfolded network of the network of Figure 10. For example, the original flow following the path $\langle 1,2,4\rangle$, leads to two flows in the unfolding: paths $\langle\langle 1,2,4\rangle, \langle 2,4\rangle, \langle 4\rangle\rangle$ and $\langle\langle 1,2,4\rangle\rangle$.

The unfolding construction Consider a feed-forward network $\mathcal{N}$ and a numbering of the servers such that if (j,h) is an arc, then $j < h$, and that server n is the only server with out-degree 0 of its induced graph (for example, $\mathcal{N}$ is the sub-network induced by a flow).

The unfolded network of $\mathcal{N}$ is denoted $\mathcal{U}$ and defined as follows:

- let Π be the set of paths in $\mathcal{N}$ ending at server n. Then Π is the set of servers of $\mathcal{U}$, server $\langle j_1, j_2, \ldots, n\rangle$ offers the service curve β_{j_1};
- for each node $\pi = \langle j_1, j_2, \ldots, n\rangle$ and each flow f_i, with path π_i, let $\pi' = \langle j_1, j_2, \ldots, j_k\rangle$ be the longest common prefix of π_i and π. If this prefix is not empty, then there is a flow (f_i, π) from node π to $\langle j_k, \ldots, n\rangle$ with arrival curve α_i. If flow f_i is ending at server n, we call the flow from π to $\langle n\rangle$ the copy for flow f_i.

The unfolding procedure can lead to exponential-size linear programs. Consider for example a network with n servers, for which the arcs of the induced graph are $\{(j,h) \mid j < h\}$. There are 2^{n-1} paths to node n (all the increasing sequence of $\{1,\ldots,n\}$ containing n), and the unfolded network has 2^{n-1} nodes.

One can easily check the following lemmas:

Lemma 1. *If $\pi \neq \pi'$, the two flows (f_i, π) and (f_i, π') do not share any common sub-path.*

Lemma 2. *For each server $\pi = \langle j, \ldots, n\rangle$ of $\mathcal{U}$, for all $i \in \mathrm{Fl}(j)$ in $\mathcal{N}$, there exists a flow (f_i, π') such that $(f_i, \pi') \in \mathrm{Fl}(\pi)$ in $\mathcal{U}$.*

Theorem 6. *Let $\mathcal{N}$ be a feed-forward network, $\mathcal{U}$ be its unfolded network and f_i be a flow of $\mathcal{N}$ ending at server n. Let $d_i^{\mathcal{N}}$ be the worst-case delay of flow f_i in $\mathcal{N}$ and $d_i^{\mathcal{U}}$ that of the copy of flow i in $\mathcal{U}$. Then $d_i^{\mathcal{N}} \leq d_i^{\mathcal{U}}$.*

Proof. Let $(F_i^{(j)})$ be an admissible trajectory for network $\mathcal{N}$. Let us build a trajectory for $\mathcal{U}$. For each flow (f_i, π) of $\mathcal{U}$, where $\pi = \langle j_1, j_2, \ldots, n\rangle$, if $\pi' = \langle j_1, j_2, \ldots, j_k\rangle$ is the maximum common prefix of π_i and π, we set $F_{i,\pi}^{(\langle j_x,\ldots,n\rangle)} = F_i^{(j_x)}$ for $x \leq k$, and $F_{i,\pi}^{(n+1)} = F_i^{(j_{k+1})}$. We now need to check that this trajectory is admissible for $\mathcal{U}$.

- First, the arrival processes $F_{i,\pi}^{(\pi)} = F_i^{(\pi_i(1))}$ so it is α_i-constrained.
- Second, consider a server $\langle j, h, \ldots, n\rangle$. The processes arriving to (resp. departing from) this server are $F_{i,\pi}^{(\langle j,h,\ldots,n\rangle)} = F_i^{(j)}$ (resp. $F_{i,\pi}^{(\langle h,\ldots,n\rangle)} = F_i^{(h)}$ or $F_{i,\pi}^{(n+1)} = F_i^{(h)}$) if $i \in \mathrm{Fl}(j)$ and $\langle j, h, \ldots, n\rangle$ is a prefix of π. The arrival and departure processes are then the same as in server j of $\mathcal{N}$, so the service and shaping constraints and FIFO properties are all satisfied.

As a consequence, the trajectory is admissible in $\mathcal{U}$. In addition, if we consider the copy (f_i, π) of flow f_i, we have $F_{i,\pi}^{(\pi)} = F_i^{(\pi(1))}$ and $F_{(i,\pi)}^{(n+1)} = F_i^{(n+1)}$, so the delay for the copy of f_i in $\mathcal{U}$ is the same as the delay of flow f_i in $\mathcal{N}$.

For each admissible trajectory of $\mathcal{N}$, we have built an admissible trajectory in $\mathcal{U}$ with the same worst-case delay for flow f_i, which means that $d_i^{\mathcal{N}} \leq d_i^{\mathcal{U}}$. □ □

A similar result holds for the backlog bounds. This unfolding procedure is very similar to the generalization from tandem network to general feed-forward networks in [13]. First, in [13], the times variables defined for each server are inherited from the time variables of all its successors. For server 1 of the network of Figure 10, server 1 inherit from time variables of servers 2 and 3. The unfolding procedure duplicates server 1, which is equivalent when considering only the dates that are introduced for each server. The difference is that the unfolding procedure forgets the dependence between duplicated servers, which can be interpreted as removing Boolean variables in the framework of [13]. Similarly, applying the unfolding procedure in the blind-multiplexing case described in [40] would lead to the same performance bounds as the linear program named ULP (taking the partial order between the time variables). It has been shown in these references that the pessimism introduced by this unfolding procedure is limited. A similar unfolding procedure has already been used in [45] for flows with static priorities. In this latter case, the unfolding could be *pruned* because of the total priority order on the flows and applied to non-feed-forward topologies. This is not the case here.

Like these approaches, the unfolding procedure remains valid when arrival curves are piecewise linear and concave, and the service curves are piecewise linear and convex.

5.2 Decomposition into a tree network by cutting flows

Another solution to cut flows into smaller pieces in order to obtain a tree, or a forest (collection of trees), and compute the arrival curves at places where flows have been cut. Cutting flows is a very classical operation in Network Calculus: one of its main strength is its modularity. The extreme case of cutting are the TFA and SFA methods where servers are analyzed one after the other (hence cut is done at each arc of the induced graph). Cutting is also key to the LUDB analysis of non-nested tandems [32], where the network is cut is nested sub-tandems. Examples of cutting strategies can also be found for the analysis of networks with arbitrary multiplexing scheduling. For instance, in [42], part of the analysis consists in cutting a network into sub-tandem to apply a PMOO analysis. In all these references, cutting is only the first step

of the analysis. The second step is combining the analyses of the smaller networks to compute end-to-end performance. The cutting procedure has been described in [17], and we briefly recall it here.

Consider $G_{\mathcal{N}} = (\mathbb{N}_n, \mathbb{A})$ the graph induced by $\mathcal{N}$, and define $\mathbb{A}_r \subseteq \mathbb{A}$ such that $(\mathbb{N}_n, \mathbb{A} - \mathbb{A}_r)$ is a tree or a forest. Note that $\mathbb{A}_r$ is not uniquely defined, and we consider any possible choice for it. A flow f_i is then transformed into K_i flows (f_i, k) with paths $\langle \pi_i(h_k^i), \ldots, \pi_i(h_{k+1}^i - 1)\rangle$ in $(\mathbb{N}_n, \mathbb{A} - \mathbb{A}_r)$, where $h_1^i = 1$, $h_{K_i+1}^i = \ell_i + 1$ and $(\pi_i(h_k^i - 1), \pi_i(h_k^i)) \in \mathbb{A}_r$ for $1 < k \leq K_i$. The transformation is illustrated in Figure 12.

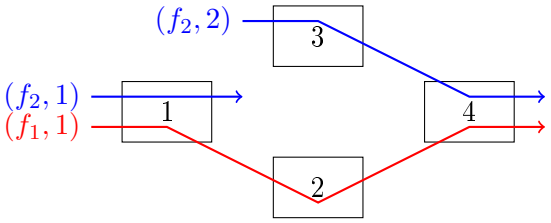


Figure 12: Decomposition of a network into a tree. Example of the network in Figure 10 .

Our aim is to compute a new network $\mathcal{N}^{\text{cut}}$ such that:

- its induced graph is the forest $(\mathbb{N}_n, \mathbb{A} - \mathbb{A}_r)$;
- its servers offer the same guarantees as those of $\mathcal{N}$;
- its flows are $F_{\text{cut}} = \{(f_i, k),\ i \in \mathbb{N}_m,\ k \in \mathbb{N}_{K_i}\}$. The arrival curve of (f_i, k) is an arrival curve for f_i at server $\pi_i(h_k^i)$;
- the arrival processes are shaped: for all $(j, h) \in \mathbb{A}^r$, flows $\{(f_i, k) \mid (\pi_i(h_k^i - 1), \pi_i(h_k^i)) = (j, h)\}$ are shaped by the curve σ_j for all $k \in \{2, \ldots, K_i\}$.

The arrival curves of the flows remain to be computed. As the network is feed-forward, the removed arcs can be sorted in the topological order, and the computations be done according to this order, as described in Algorithm 4.

Algorithm 4: Network analysis of a feed-forward network by flow cutting

```
1 begin
2 |  Sort 𝔸_r in the topological order in 𝒩 according to the first coordinate;
3 |  foreach arc (j,h) in the topological order do
4 |  |  foreach flow (f_i, k+1) with k > 0 starting at server h do
5 |  |  |  Compute an arrival curve for flow (f_i, k+1)
```

To compute of the arrival curve for flow $(f_i, k+1)$, we will use Theorem 4. The service curves and greedy shapers are the only constraints for the servers, so hypothesis (H_2) is satisfied. In short, one just has to compute the maximum backlog of flow (f_i, k). Let j be the first server crossed by flow (f_i, k). There are two possibilities:

- either we do not take into account the greedy shaper of server j, so (H_1) is satisfied and Theorem 4 can be applied. However, not taking into account the shaping effect could lead to pessimistic bounds;
- or we take into account the greedy shaper of server j. If done directly, (H_1) is not satisfied and Theorem 4 cannot be applied.

We propose to slightly transform the network so that the shaping effect can be taken into account for all flows except flow (f_i, k). The transformation is illustrated in Figure 13.

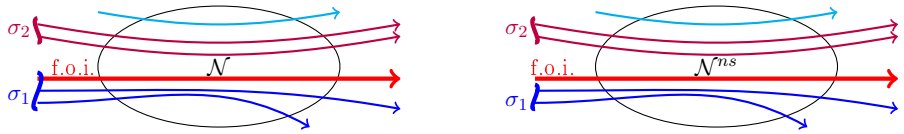


Figure 13: System transformation: (left) the flow of interest (in red and bold) is shaped with two other flows (in blue) by the greedy-shaper σ_1; (right) the flow of interest is not shaped anymore. The rest of the network is not modified.

More precisely, let us consider $\mathcal{N}_{(f_i,k)} = \mathcal{N}^{\text{cut}}((f_i,k))$ the sub-network of $\mathcal{N}^{\text{cut}}$ induced by flow (f_i,k) and $\mathcal{N}^{ns}_{(f_i,k)}$ the same network, except the shaping of the arrival processes becomes: for all h such that $(j,h) \in \mathbb{A}^r$, flows $\{(f_{i'},k') \mid (\pi_{i'}(h^{i'}_{k'}-1), \pi_{i'}(h^{i'}_{k'})) = (j,h)\} \setminus \{(f_i,k)\}$ are shaped by the curve σ_j.

Lemma 3. *If α is an arrival curve for the departure process of flow (f_i,k) in $\mathcal{N}^{ns}_{(f_i,k)}$, then it is also an arrival curve for the departure process of that flow in $\mathcal{N}_{(f_i,k)}$.*

Proof. If flow (f_i,k) is not shaped then the two networks are the same and there is nothing to prove. Otherwise, let us denote I the set of flows shaped together with flow (f_i,k) by σ_j in $\mathcal{N}_{(f_i,k)}$. Let $(F^{(j)}_{\tilde{f}})$ be an admissible trajectory in $\mathcal{N}_{(f_i,k)}$. It is then also an admissible trajectory in $\mathcal{N}^{ns}_{(f_i,k)}$. To prove this, it is enough to check that $\sum_{\tilde{f}\in I\setminus\{(f_i,k)\}} F^{(j)}_j$ is σ_j-constrained: $\forall s \le t$,

$$\sum_{\tilde{f}\in I\setminus\{(f_i,k)\}} \left(F^{(j)}_{\tilde{f}}(t) - F^{(j)}_{\tilde{f}}(s)\right) \le \sigma_j(t-s) - \left(F^{(j)}_{(f_i,k)}(t) - F^{(j)}_{(f_i,k)}(s)\right) \le \sigma_j(t-s),$$

since $F^{(j)}_{(f_i,k)}$ is non-decreasing.

The set of possible departure processes of (f_i,k) in $\mathcal{N}_{(f_i,k)}$ is then included in those of (f_i,k) in $\mathcal{N}^{ns}_{(f_i,k)}$, meaning that if α is an arrival curve for the departure process in $\mathcal{N}^{ns}_{(f_i,k)}$, it is also one for those in $\mathcal{N}_{(f_i,k)}$. □ □

Using the transformation, (H_1) is now satisfied and according to Theorem 4, the arrival curve of flow $(f_i,k+1)$ can then be computed according to Algorithm 5, where the backlog bound can be computed with the linear program given in Section 4.

Algorithm 5: Arrival curve for flow $(f_i,k+1)$

```
begin
    if k = 0 then return α_i;
    Compute N_(f_i,k) = N^cut((f_i,k)) the network induced by flow (f_i,k) in N^F (see page 7);
    Compute N^ns_(f_i,k) = (N_(f_i,k))^ns the network where shaping of flow (f_i,k) has been removed;
    Compute B the backlog bound for flow (f_i,k) in N^ns_(f_i,k) computed with the linear program presented in Section 4;
    return γ_{B,r_i}
```

The end-to-end delay of a flow is then obtained by summing the end-to-end delays of its sub-flows.

5.3 Trade-off between accuracy and tractabilty

The two constructions presented in this section can be applied to any linear programming method, which includes the linear program presented in [12, 13]. Also, the unfolding and cutting procedures can be combined: a network can be cut in sub-networks that are not necessarily trees, and unfolding be performed on each sub-network.

The cutting procedure can also be used to improve the tractability of the linear programming methods: if a network is cut in (more) smaller sub-networks, then the time needed to analyze each sub-network will be reduced. A trade-off has then to be found between the size of sub-networks, and the number of intermediate arrival curves to compute.

Cutting in smaller pieces also deteriorates the accuracy and natural questions arise:

- What is the maximal size of a sub-network to guarantee the tractability of the computation?
- Is it better to use PLP with cutting into medium-size sub-networks (e.g., 10-20 nodes) or ELP with small size sub-networks (e.g., 5 nodes)?
- Where to cut the network?

These questions strongly depend on the analyzed network. An example is given in Section 7 for the two first questions, where a tandem is decomposed into pieces of different lengths, and both delay bounds and computation time are compared for PLP and ELP.

The third question is more involving and is beyond obtaining sub-networks of reasonable size. Intuitively, cutting arcs leading to under-loaded servers and arcs crossed by single flows seem good starting points to build a first heuristic.

Deciding where to cut a network is an optimization problem itself, and has been studied in other contexts. In blind multiplexing, Bondorf et al. [42] compute the performance bounds for each possible (min, plus) decomposition of the network. In [46], Geyer and Bondorf use deep neural networks to find a quasi-optimal decomposition of the network. In the FIFO context, in [10], Thomas et al. cut the network and adds regulator at cuts. The optimization problem is to minimize the number of regulators while maintaining good performance bounds.

6 Network with a general topology

In this section, we study networks with a general topology, including those with cyclic dependencies. For this, we will apply the fixed-point analysis, that has already been described several times in [17, 18, 19], to the analysis of Section 5.2: arcs are removed, so that the induced graph becomes a forest and flows are cut accordingly. Because of the cyclic dependencies, removed arcs cannot be sorted in the topological order, and the fixed point on the arrival curves of all cut flows has to be computed.

In Paragraph 6.1, we first prove a general result about linear programs and fixed points, and first apply it to compute performance bounds in networks with a general topology. Then in Paragraph 6.2, we apply the same result to provide an alternative presentation of the TFA++ analysis for networks with cyclic dependencies developed in [10].

6.1 A linear-program formulation for the analysis of networks with general topology

We formulate the fixed-point equation with a linear program. More precisely, we prove that the fixed point is obtained by extracting some variables from the optimal solution of a linear program.

A first generic result about fixed points Let us first state a generic result about linear programs: We call a LP-function a function $\mathcal{L} : \mathbb{R}_+^N \to \mathbb{R}_+^N$ such that for all $p \in \mathbb{N}_N$, there exists matrices A_p, B_p, C_p such that for all $x \in \mathbb{R}_+^N$,

$$\mathcal{L}_p(x) = \max\{A_p(x,y)^t \mid B_p(x,y)^t \leq C_p,\ y \geq 0\},$$

where vectors x and y represent variables, and $(x,y)^t$ is the transposition of the line vector (x,y).

We will focus on LP-functions that satisfy the properties (P_1) and (P_2) stated below. Note that they only concern coefficients that relate to the variables x and not to the variables y: in the next description, $(B_p)_{c,q}$ is the coefficient of the variable x_q for constraint c.

- (P_1) For all constraints c, for all $\tilde{f}' \in F_{\text{cut}}$, a) $[(B_p)_{c,q} < 0 \wedge (B_p)_{c,r} < 0] \implies q = r$ and b) $(B_p)_{c,q} \leq 0$. In other words, a) there is at most one variable of type x_q in each constraint, and b) this variable appears as an upper bounds.
- (P_2) For all q, $(A_p)_q \geq 0$: the objective is increasing with the variables x_q.

We will use the following result, proved in C.

Theorem 7. *Let $\mathcal{L} : \mathbb{R}_+^N \to \mathbb{R}_+^N$ be a LP-function such that $\mathcal{L}_p(0) > 0$ for all $p \in \mathbb{N}_N$ and satisfying properties (P_1) and (P_2). Then $x^* = \sup\{x \in \mathbb{R}_+^N \mid x \leq \mathcal{L}(x)\} \in (\mathbb{R}_+ \cup \{+\infty\})^N$ is well-defined and if $x^* \in \mathbb{R}_+^N$, it is the unique fixed-point of $\mathcal{L}$.*

Moreover, x^* is also the optimal solution of the linear program

$$\max\{\sum_{p=1}^{N} x_p \mid \forall p \in \{1,\dots,N\}, x_p \leq A_p(x,y_p)^t, B_p(x,y_p)^t \leq C_p,\ x, y_p \geq 0\}, \quad (3)$$

where we call optimal solution *a vector $x \in \mathbb{R}_+^N$ such that there exists $y_p \geq 0$ satisfying the constraints of this linear program and maximizing the sum of its coefficients.*

Application to the linear-programming approach We use the same notations as in Paragraph 5.2: the induced graph of the cut network $\mathcal{N}^{\text{cut}}$ is a forest and $F_{\text{cut}} = \{(f_i,k),\ i \in \mathbb{N}_m, k \in \mathbb{N}_{K_i}\}$ is the set of flows in the network with cuts. Let us set $N = |F_{\text{cut}}|$. As we will use Theorem 4, all the arrival curves computed for a flow (f_i,k) will have arrival rate r_i, we only focus of the burst of the flows, that we denote $x_{(f_i,k)}$ to enforce that it is a variable.

For all $x = (x_{\tilde{f}})_{\tilde{f} \in F_{\text{cut}}} \in \mathbb{R}_+^N$ and all $\tilde{f} \in F_{\text{cut}}$, let us define $\mathcal{L}_{\tilde{f}}(x)$ as the backlog bound computed at line 5 of Algorithm 5, when the burst parameter of flow $\tilde{f}'$ is $x_{\tilde{f}'}$ for all $\tilde{f}' \in F_{\text{cut}}$ and $\mathcal{L}(x) = (\mathcal{L}_{\tilde{f}}(x))_{\tilde{f} \in F_{\text{cut}}}$. Since the cut network is a forest (hence feed-forward) and since we assume the local stability of the network, $\mathcal{L}(x) \in \mathbb{R}_+^{|F_{\text{cut}}|}$, and it is possible to define $\mathcal{C} = \{x \in \mathbb{R}_+^N \mid x \leq \mathcal{L}(x)\}$.

Theorem 8 reminds a sufficient condition for the stability of the network and the arrival curves of the cut flows.

Theorem 8 ([17, Theorem 12.1])**.** *If $x^* = \sup(\mathcal{C})$ is finite, then $\mathcal{N}$ is globally stable and the burst of the arrival curve of flow $\tilde{f}$ is $x^*_{\tilde{f}}$.*

Proof. To prove the result, we use the classical *stopped-time* method. For all $\tau \in \mathbb{R}_+$, we call and denote $A^\tau = A : t \mapsto A(t \wedge \tau)$ the process A stopped at time τ. Consider network $\mathcal{N}^\tau$ when the arrival processes are stopped at time τ. As the amount of data entering the network is finite, network $\mathcal{N}^\tau$ is stable. Let $\alpha^\tau = (\alpha^\tau_{\tilde{f}})_{\tilde{f} \in F_{\text{cut}}}$ be the family of the best arrival curves in

$\mathcal{N}^\tau$ at each cut of $\mathcal{N}^{\text{cut}}$ and F^τ be a method to compute these arrival curves as a function of α^τ in $\mathcal{N}^{\text{cut}}$. As α^τ is the family of *best* arrival curves, we have $\alpha^\tau \leq F(\alpha^\tau)$.

Suppose that we restrict to the arrival curves of the form $\gamma_{b,r_{\tilde{f}}}$ for flow $\tilde{f}$ (where $r_{(f_i,k)} = r_i$) and $x^\tau = (x^\tau_{\tilde{f}})_{\tilde{f}\in F_{\text{cut}}}$ is the best burst parameters. For all functions G computing bursts parameters, we also have $x^\tau \leq G(x^\tau)$. Indeed, for all $r \in \mathbb{R}_+$, for all arrival curve α, the smallest function $\gamma_{b,r}$ such that $\alpha \leq \gamma_{b,r}$ is with $b = \sup_{t\in\mathbb{R}_+}\{\alpha(t) - rt\}$. Then, for each choice of F, the best burst parameter for each flow $\tilde{f}$ is $x^\tau_{\tilde{f}} = \sup_{t\in\mathbb{R}_+}\{\alpha^\tau_{\tilde{f}}(t) - r_{\tilde{f}}t\} \leq \sup_{t\in\mathbb{R}_+}\{F(\alpha^\tau)_{\tilde{f}}(t) - r_{\tilde{f}}t\} := G(x^\tau)_{\tilde{f}}$.

This is true for all functions G computing burst parameters at cuts, so this is in particular true for $\mathcal{L}$, and we have $x^\tau \leq \mathcal{L}(x^\tau)$.

If x^* is finite, then for all τ, $x^\tau \leq x^*$, so x^* is a family of arrival curves at the cuts for the (non-stopped) processes. The amount of data in transit in $\mathcal{N}$ can be bounded by the amount of data in transit in the locally stable feed-forward network $\mathcal{N}^{\text{cut}}$, where the arrival curve of flow $\tilde{f}$ is $\gamma_{x^*_{\tilde{f}},r_{\tilde{f}}}$, hence it is stable. □ □

Our goal is now to apply Theorem 7 in order to compute $\sup(\mathcal{C})$ as the linear program of Equation (3). Let us first focus on some properties of $\mathcal{L}_{\tilde{f}}(x)$. The variables of this linear program are the time variables $\mathbf{t}_{(j,k)}$ and process variables $\mathbf{F}^{(j)}_i\mathbf{t}_{(j,k)}$ described in Section 4. These variables are represented by the vector y in the generic formulation. The burst parameters only appear in the arrival curve constraints such as $\mathbf{F}^{(j)}_i\mathbf{t}_{(j,k')} - \mathbf{F}^{(j)}_i\mathbf{t}_{(j,k)} \leq x_i + r_i(\mathbf{t}_{(j,k')} - \mathbf{t}_{(j,k)})$. So if x_i becomes a variable, the program remains linear: function $\mathcal{L}$ is an LP-function.

We now show that $\mathcal{L}$ satisfied properties (P_1) and (P_2).

- The objective for flow $\tilde{f}$ is $\max : \mathbf{F}^{(j)}_{\tilde{f}}\mathbf{t}_{(n+1,0)} - \mathbf{F}^{(n+1)}_{\tilde{f}}\mathbf{t}_{(n+1,0)}$ for some j and n. Vector $A_{\tilde{f}}$ has only null coefficients, except at coordinates corresponding to variables $\mathbf{F}^{(j)}_{\tilde{f}}\mathbf{t}_{(n+1,0)}$ (coefficient 1) and $\mathbf{F}^{(n+1)}_{\tilde{f}}\mathbf{t}_{(n+1,0)}$ (coefficient -1). The objective has no variable of type $x_{\tilde{f}'}$, so (P_2) is satisfied.

- If constraint c is $\mathbf{F}^{(j)}_{\tilde{f}'}\mathbf{t}_{(j,k')} - \mathbf{F}^{(j)}_{\tilde{f}'}\mathbf{t}_{(j,k)} \leq x_{\tilde{f}'} + r_{\tilde{f}'}(\mathbf{t}_{(j,k')} - \mathbf{t}_{(j,k)})$, row c of matrix $B_{\tilde{f}}$ has only null coefficients, except those corresponding to variables $\mathbf{F}^{(j)}_{\tilde{f}'}\mathbf{t}_{(j,k')}$ (coefficient 1), $\mathbf{F}^{(j)}_{\tilde{f}'}\mathbf{t}_{(j,k)}$ (coefficient -1), $\mathbf{t}_{(j,k')}$ (coefficient $-r_i$), $\mathbf{t}_{(j,h)}$ (coefficient r_i) and $x_{\tilde{f}'}$ (coefficient -1).

  No variable $x_{\tilde{f}}$ appears in the other types of constraints.

  In conclusion, at most variable of type $x_{\tilde{f}'}$ appears in each constraint, with coefficient -1, so (P_1) is satisfied.

We can now apply the formulation of Equation (3), to the linear programs of Section 4. We proceed as follows:

- we first compute a delay bound for each server using TFA++ (either using [10] or the next paragraph) in order to be able to include TFA++ constraints. As flows are cut, we do not compute the SFA delays for each flow, hence will not include the SFA constraints in the procedure.

- we write the linear constraints for computing the backlog of each flow (f_i, k), $k < K_i$, replacing all parameters $b_{\tilde{f}}$ (the burst parameter of the arrival curve of flow $\tilde{f}$) by a variable $x_{\tilde{f}}$. Except the variables $x_{\tilde{f}}$, all variables are used in only one *sub-linear program* (one sub-linear program per backlog computed);

- each objectives $\max : \mathbf{F}_{\tilde{f}}^{(j)}\mathbf{t}_{(j,k)} - \mathbf{F}_{\tilde{f}}^{(n+1)}\mathbf{t}_{(n+1,0)}$ is replaced by the linear constraint $x_{\tilde{f}} \leq \mathbf{F}_{\tilde{f}}^{(j)}\mathbf{t}_{(j,k)} - \mathbf{F}_{\tilde{f}}^{(n+1)}\mathbf{t}_{(n+1,0)}$;
- for flow $\tilde{f} = (f_i, 1)$, the constraint $x_{\tilde{f}} \leq b_i$ is added;
- the objective is $\max : \sum_{\tilde{f} \in F_{\text{cut}}} x_{\tilde{f}}$.

The end-to-end delay of a flow is computed by summing the end-to-end delays of its sub-flows, setting the burst parameters of the cut flows as the fixed-point just computed.

The assumption $\mathcal{L}_{\tilde{f}}(0) > 0$ for all $\tilde{f} \in F_{\text{cut}}$ has not been discussed yet. It might not always hold. For example, when the initial bursts and the latencies of the servers are all null. However, this assumption holds in general. As we do not shape the flows of interest to compute the burst, we have the following properties: consider a flow with arrival curve $\gamma_{b,r}$ crossing a server with service curve $\beta_{R,T}$. The maximal backlog in the server is $b + rT$, so if either b or T is non-null, the backlog bound transmitted for the bound of the next flow is non-null. Then there exists k such that $\mathcal{L}_{\tilde{f}}^{k}(0) > 0$ for all $\tilde{f}$, where $\mathcal{L}^k$ is the k-th iteration of the composition of $\mathcal{L}$. Here, for example, k can be chosen as $\max_i K_i$. The positivity assumption is only required for the uniqueness of the fixed point of $\mathcal{L}$. Because $\mathcal{L}$ is non-decreasing and concave (see Lemma 5), so is $\mathcal{L}^k$. Then Lemma 7 can be applied to $\mathcal{L}^k$. As a fixed point of $\mathcal{L}$ is necessary a fixed point of $\mathcal{L}^k$, this proves the uniqueness of the fixed point of $\mathcal{L}$.

Finally, note that these results also apply to the other methods to compute performance bounds that are based on linear programming. In particular, this applies to the exponential size linear program for FIFO networks of [12], and the linear program for blind multiplexing of [40].

6.2 Application to TFA++

Let us now give an alternative formulation of the solution with TFA++ given in [10]. In that paper, the authors compute performances networks with cyclic dependencies using the TFA++ method for each server.

Two limitations can be overcome by using the linear-programming approach.

1. The authors assume that they take into account the shaping only when the shaping rates exceed some given value. The reason for this seems to be the simplification of the computation of maximum delay, and more precisely the place where the horizontal distance between the aggregate arrival curve and the service curve is maximized. With the linear programming approach, this place is not directly computed but given as the solution of a linear program. It also allows more complex service curves than rate-latency ones.

2. The method chosen to compute the fixed point is iteration from the origin. The authors show that the least fixed point is indeed a valid solution for computing the performance bounds, which is an improvement compared to [17] that states this result for the greater fixed point. As we showed the uniqueness of the fixed point, the two previous approaches are in fact similar. Here, the linear program avoids the iteration whose raw output would be a lower approximation of the fixed point (the iterated are lower bounds of the fixed point) and thus require additional (yet simple) computations to obtain upper bounds. The authors choose to compute to apply ceiling functions at each iteration, so that they can ensure, in case if convergence, convergence in finite time. Still, the relation between the fixed point of the function and the fixed point of the ceiling of this function is unclear if the uniqueness of the fixed point is not assumed.

Linear program for the TFA++ method: single server The key-step of the TFA++ method is to compute an upper bound of the delay for each server.

First consider a single-server network crossed by m flows $f_1, \ldots, f_m$ with respective arrival curves $\alpha_i = \gamma_{x_i,r_i}$. Assume that some groups of flows are shaped: there exist H disjoint sets $G_h \subseteq \mathbb{N}_m$ for $h \in \mathbb{N}_H$, such that the aggregation of flows $(f_i)_{i \in G_h}$ is constrained by the greedy-shaper $\sigma_h = \gamma_{L_h,C_h}$. The server offers a service curve $\beta_{R,T}$.

Computing a delay bound for the server can be done using a linear program presented in Section 4. In the case of one server, the SFA and TFA++ constraints are useless. We use the notation A_i (resp. D_i) for the arrival (resp. departure) process of flow f_i. This linear program is presented in Table 3.

Maximize	$\mathbf{t} - \mathbf{s}$	
	$\mathbf{u} \le \mathbf{s} \le \mathbf{t}$	time constraints
$\forall i \in \mathbb{N}_m$	$\mathbf{A}_i\mathbf{s} - \mathbf{A}_i\mathbf{u} \le x_i + r_i\mathbf{s} - r_i\mathbf{u}$	arrival constraints
$\forall h \in \mathbb{N}_H$	$\sum_{i \in G_h}(\mathbf{A}_i\mathbf{s} - \mathbf{A}_i\mathbf{u}) \le L_h + C_h\mathbf{s} - C_h\mathbf{u}$	shaping constraints
	$\sum_{i=1}^m \mathbf{D}_i\mathbf{t} \ge \sum_{i=1}^m \mathbf{A}_i\mathbf{u} + R\mathbf{t} - R\mathbf{u} - RT$	service constraints
	$\sum_{i=1}^m \mathbf{D}_i\mathbf{t} \ge \sum_{i=1}^m \mathbf{A}_i\mathbf{u}$	
$\forall i \in \mathbb{N}_m$	$\mathbf{A}_i\mathbf{s} = \mathbf{D}_i\mathbf{t}$	FIFO constraints

Table 3: Linear program for the delay bound of one server.

Note that this linear program can be simplified: $\mathbf{s} - \mathbf{u}$ can be replaced by $\mathbf{s}$, $\mathbf{t} - \mathbf{u}$ by $\mathbf{t}$, $\mathbf{A}_i\mathbf{s} - \mathbf{A}_i\mathbf{u}$ by $\mathbf{A}_i\mathbf{s}$, and $\mathbf{D}_i\mathbf{t} - \mathbf{A}_i\mathbf{u}$ by $\mathbf{D}_i\mathbf{t}$. Doing so, one gets rid of variables $\mathbf{u}$ and $\mathbf{A}_i\mathbf{u}$.

This linear program corresponds to the formulation of PLP and ELP, and all time variables can be ordered in the single-server case, so the linear program corresponds to the exact worst-case delay computed in [12], hence to the maximum horizontal distance between the aggregate arrival curve and the service curve. For the sake of being self-contained, we give a direct proof in D.

Lemma 4. *The optimal solution of Table 3 is the horizontal distance between* $\alpha = \sum_{h=1}^{H}(\sigma_{L_h,C_h} \wedge \sum_{i \in G_h} \gamma_{x_i,r_i}) + \sum_{i \notin \bigcup_{h \in \mathbb{N}_H} G_h} \gamma_{x_i,r_i}$ *and* $\beta_{R,T}$.

The linear program to compute the burst parameter of the departure process of any flow crossing the server is then given in Table 4 (exemplified for flow f_1).

Maximize	$x_1 + r_1\mathbf{d}$	
	$\mathbf{s} \le \mathbf{t}$	time constraints
$\forall i \in \mathbb{N}_m$	$\mathbf{A}_i\mathbf{s} \le x_i + r_i\mathbf{s}$	arrival constraints
$\forall h \in \mathbb{N}_H$	$\sum_{i \in G_h} \mathbf{A}_i\mathbf{s} \le L_h + C_h\mathbf{s}$	shaping constraints
	$\sum_{i=1}^m \mathbf{D}_i\mathbf{t} \ge +R\mathbf{t} - RT$	service constraints
	$\sum_{i=1}^m \mathbf{D}_i\mathbf{t} \ge 0$	
$\forall i \in \mathbb{N}_m$	$\mathbf{A}_i\mathbf{s} = \mathbf{D}_i\mathbf{t}$	FIFO constraints
	$\mathbf{d} = \mathbf{t} - \mathbf{s}$	delay at server j

Table 4: Linear program for the burst parameter of the departure process of flow f_1.

Linear program for the TFA++ method: general topology Now consider a network with a general topology. In the TFA++ analysis, the network is cut at every arc: $\mathbb{A}_r = \mathbb{A}$ and one can denote the cut flows by $f_{i,j}$, $j \in \pi(i)$, where $f_{i,j}$ corresponds to flow i crossing server

j. We use the notation $A_i^{(j)}$ (resp. $D_i^{(j)}$) for the arrival (resp. departure) process of flow f_i at server j.

One can define the function $\mathcal{L}$ as follows:

- if $j = \pi_i(1)$, $\mathcal{L}_{f_{i,\pi_i(1)}}(x) = b_i$: the burst parameter at the first server crossed by a flow is fixed. Properties (P_1) and (P_2) are satisfied;
- otherwise, $\mathcal{L}_{f_{i,\pi_i(k)}}(x)$ is given by the linear program of Table 4, where x_1 is $x_{f_{i,\pi_i(k-1)}}$ and the other variables x_i are $x_{f_{i',\pi_i(k-1)}}$, the burst of the flows entering server $\pi_i(k-1)$. Note that in the arrival constraints, whenever a variable x_i is present, it appears as an upper bound, so (P_1) is satisfied. Moreover, the objective is increasing with x_1, so (P_2) is satisfied.

From Theorem 7, following the formulation of Equation (3), the linear program used to compute the fixed point is written in Table 5.

Strictly applying formulation of Equation (3) would require solving $|\mathrm{Fl}(j)|$ times the linear program corresponding to server j. This is unnecessary as the same delay bound is computed each flow crossing server j, and similarly, as the bursts are increasing proportionally to the delay bounds, the objective $\sum_{i,j} \mathbf{x}_i^{(j)}$ can be replaced by $\sum_j \mathbf{d}_j$.

Maximize	$\sum_{i,j} \mathbf{x}_i^{(j)}$ or $\sum_j \mathbf{d}_j$	
such that	for all server j	
	$0 \leq \mathbf{s}_j \leq \mathbf{t}_j$	time constraints
$\forall i \in \mathrm{Fl}(j)$	$\mathbf{A}_i^{(j)} \mathbf{s}_j \leq \mathbf{x}_i^{(j)} + r_i \mathbf{s}_j$	arrival constraints
$\forall h$	$\sum_{i \in \mathrm{Fl}(h,j)} \mathbf{A}_i^{(j)} \mathbf{s}_j \leq L_h + C_h \mathbf{s}_j$	shaping constraints
	$\sum_{i \in \mathrm{Fl}(j)} \mathbf{D}_i^{(j)} \mathbf{t}_j \geq R_j \mathbf{t}_j - R_j T_j$	service constraints
	$\sum_{i \in \mathrm{Fl}(j)} \mathbf{D}_i^{(j)} \mathbf{t}_j \geq 0$	
	$\mathbf{A}_i^{(j)} \mathbf{s}_j = \mathbf{D}_i^{(j)} \mathbf{t}_j$	FIFO constraint
	$\mathbf{d}_j \leq \mathbf{t}_j - \mathbf{s}_j$	delay at server j
$\forall i \in \mathrm{Fl}(j)$	$\mathbf{x}_i^{(\mathrm{succ}(j))} \leq \mathbf{x}_i^{(j)} + r_i \mathbf{d}_j$	burst propagation
$\forall i$	$\mathbf{x}_i^{(\pi_i(0))} \leq b_i$	initial bursts

Table 5: Linear program for TFA++ with general topology.

Equivalence with the bound of [10] In this paragraph, we make explicit the relation between out TFA++ delay bound and that computed in Theorem 2 of [10]. In that paper, the shaping curves are $\sigma_h = \gamma_{\ell_{\max}, c_h}$, where $\ell_{\max}$ is the maximum length of a packet and c_h the shaping rate for server h. The procedure of Theorem 2 of [10] is done in three steps for each server:

1. Compute the aggregate arrival curve α at server j, using the arrival curves of each flow crossing it and the greedy shapers from the preceeding servers. The arrival curve is given

by Equation (2) in [10] is (with our notations[2] and neglecting the packetization effect)

$$\alpha = \sum_{h=1}^{H} (\sigma_h \wedge \sum_{i \in G_h} \alpha_i) + \sum_{i \notin \bigcup_{h \in \mathbb{N}_H} G_h} \alpha_i.$$

To take into account the packetization effect, $\alpha_i = \gamma_{b_i,r_i}$ must be replaced by $\gamma_{b_i+\ell\max\frac{r_i}{C_h},r_i}$.

2. Compute the delay bound D as the maximal horizontal distance between α and the service curve β of this server (that is rate latency). This is where the additional hypothesis on a lower bounds of the shaping rates is used, to simplify the computation of the maximal horizontal distance.

3. Compute the arrival curves of the departure processes for each flow after this server as[3]

$$\alpha_i' = \alpha_i \oslash \delta_D = \gamma_{b_i+r_iD,r_i}.$$

The packetization effect enables to improve this formula by replacing D by $D' = D - \ell_{\min}(\frac{1}{R_j} - \frac{1}{C_j})$, where $\ell_{\min}$ is the minimum length of a packet and C_j the shaping rate of the server.

Table 4 directly implements the three steps, and Table 5 computes the fixed point without resorting to iteration. Incidentally, ceiling is not required anymore.

In Tables 4 and 5, the packetization effect and improvement are neglected. They can be taken into account by replacing the propagation constraint by $\mathbf{x}_i^{(\mathrm{succ}(j))} \leq \mathbf{x}_i^{(j)} + r_i\mathbf{d}_i - \ell_{\min}(\frac{1}{R_j} - \frac{1}{C_j}) + \ell_{\max}\frac{r_i}{C_j}$. The term with $\ell_{\min}$ corresponds to the improvement of step 3, and the term with $\ell_{\max}$ to the packetization effect in step 1.

7 Experimental results

In this section, we compare our solution to the state-of-the-art ones. In the first part of the experimental results, we use very simple topologies for the comparisons: tandem networks, mesh network and the ring. In the second part, we use networks topologies met in realistic applications.

The numerical evaluations are performed on a laptop equipped with Intel Core i7-8565 1.8GHz CPU, 16 Gb of RAM, Windows 10 64bit. The algorithms presented in this paper have been implemented in `Python 3.7`. The linear programming methods are solved in two steps. First a linear program is generated (with a `Python` script), and then solved with the open-source linear programming solver `lp_solve 5.5.2.11`. This implementation is not optimal for at least two reasons. First commercial like `Cplex` and `Gurobi` and other open source can solve linear programs more than 100 times faster [47, 42]. Second, there exists solutions to use `lp_solve` directly inside Python code. This choice is made to allow easy interpretation of the solutions.

This being said, the comparison of the computation times between non-LP solutions and LP solutions can be unfair, but the comparison between LP solutions is still valid. We do not

[2] In [10], $\lambda_{c_h} = \gamma_{0,c_h}$ and we assume that by "$\lambda_{c_h} + \ell_{\max}$", the authors mean $\sigma_h = \gamma_{\ell_{\max},c_h}$, and similarly for the right-hand term when packetization is taken into account. In that case, the convolution is between two sub-additive functions null at 0, and the (min, plus) convolution in the paper is also the minimum of the functions (see [17, Proposition 2.6]).

[3] In [10], Equation (4) is $\alpha_i' = \alpha_i \wedge \gamma_{0,C_j} \oslash \delta_D$. But it is also stated that the obtained function is leaky bucket. This is not true if $D < \frac{b_i}{C_j - r_i}$. Indeed, $\alpha_i \wedge \gamma_{0,C_j} \oslash \delta_D = \alpha_i(\cdot + D) \wedge \gamma_{0,C_j}(\cdot + D) = \gamma_{b_i+r_iD,r_i} \wedge \gamma_{C_jD,C_j}$ and $\gamma_{C_jD,C_j} \geq \gamma_{b_i+r_iD,r_i}$ if and only if $C_jD \geq b_i + r_iD$. We then assume that the authors meant our formulation.

compare the execution times of the scalable methods: whatever the network, they remain below 200 ms. Our interest is about the ability of our PLP to analyze medium-size networks (up to 25 servers) in reasonable time, not being faster than TFA++ or SFA.

7.1 Toy examples

We will compare the five methods we introduced in this paper: LUDB (the upper bound computed by the Deborah software [30]), TFA++, SFA, ELP (linear program upper bound from [12], without Boolean variables, but including shaping constraints), PLP (polynomial-size linear program from Section 4), and the performances obtained when the network has regulators after each server [9]. Regulators regulate the flows according to their arrival curve. It is shown that the delays induced by them is not modified, but the end-to-end delays are computed in a way similar to the TFA algorithm recalled in Algorithm 1, except that in line 5, $b_i^{(succ_i(j))}$ remains $b_i^{(j)}$. When we mention the delay bounds with regulators, this refers to the bounds computed this way, and not to the worst-case delay bounds when inserting regulators.

7.1.1 Tandem networks

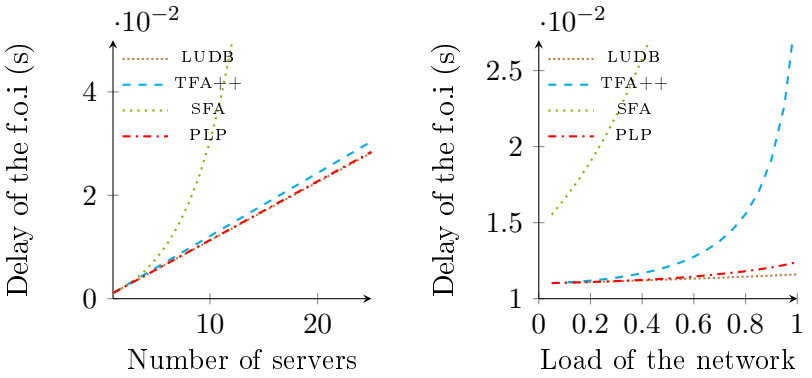
A tandem network with n servers is a network whose underlying graph is a line. By convention, we number the servers from 1 to n in the topological order. We assume uniform networks: each server offers a service curve $\beta : t \mapsto R(t-T)_+$ and a greedy-shaper $\sigma : t \mapsto \eta Rt$, with $\eta \geq 1$; each flow is constrained by the arrival curve $\alpha : t \mapsto b + rt$.

We fix $T = 0.001\,\text{s}$, $b = 1\,\text{Kb}$ and $R = 10\,\text{Mb/s}$. The arrival rate r will vary to study the network at different loads, and η vary in order to study the sensitivity of TFA++ and PLP to the shaping rate. We say that delay d_1 is $X\%$ higher than delay d_2 if $(d_1 - d_2)/d_2 = 0.01X$.

The flow of interest (f.o.i.) crosses all servers, and we will study three different configurations: sink-tree tandems, interleaved tandems, and source-sink networks.

Sink-tree tandems First, in order to assess the accuracy of our approach compared to exact method, we compare the bounds of the different methods for sink-tree tandems, where LUDB (Deborah) computes exact worst-case delays: for a tandem of n servers, there are n flows with respective paths $\langle i, \ldots, n \rangle$, $i \in \mathbb{N}_n$.

Sink-tree tandems are a particular case of nested tandems, so using shaping with $\eta = 1$ does not decrease the worst-case delay.



(a) Delay in function of the length of the tandem.

(b) Delay in function of the load.

Figure 14: Comparison of different methods for sink-tree tandems.

Figure 14a depicts the delays when the length of the tandem grows for $U = 0.5$. The difference between LUDB and PLP is not visible, and less than 1%. One can also notice that the TFA++ bound is quite accurate (TFA++ is 8% higher than LUDB for length 25), whereas SFA is outperformed. Figure 14b depicts the delay bounds for a sink-tree tandem of length 10 and for varying loads. PLP slightly looses accuracy (at load $U = 1$, PLP is 7% higher than LUDB). The two other bounds are outperformed at high loads.

Interleaved tandems There are $n-1$ interfering flows, with path $\langle i, i+1 \rangle$ for all $1 \leq i < n$ as depicted on Figure 15 for 4 servers.

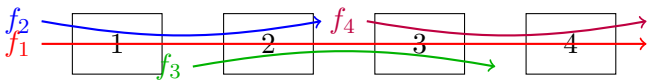


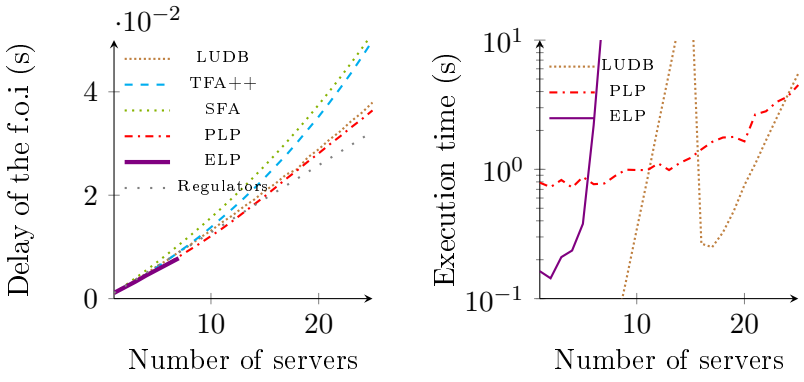
Figure 15: Interleaved tandem network of length 4.

If the arrival rate is r, then the load of the network is $U = 3r/R$.

Figure 16a shows the delay bound obtained when the number of servers varies from 1 to 25, when $\eta = 1$ and $U = 0.5$. ELP method can only be computed up to 7 servers. One can check that ELP gives the tightest bound, followed by PLP, which confirms to our results.

The regulators also allow obtaining slightly better bounds for tandems longer than 15, which seems also intuitive as the bursts cannot propagate in the network, and regulating the flows then has more effect on longer networks. More surprisingly, one can see that the SFA method, that does not take into account the shaping effect of the maximal service curve behaves almost like TFA++, specially for long network. For the tandem of length 25, TFA++ is 37% higher than PLP and SFA 41% higher than PLP. LUDB is very close to PLP. To obtain this we run the option `--ludb-nnested-sta` (without this option, SFA outperforms LUBD).

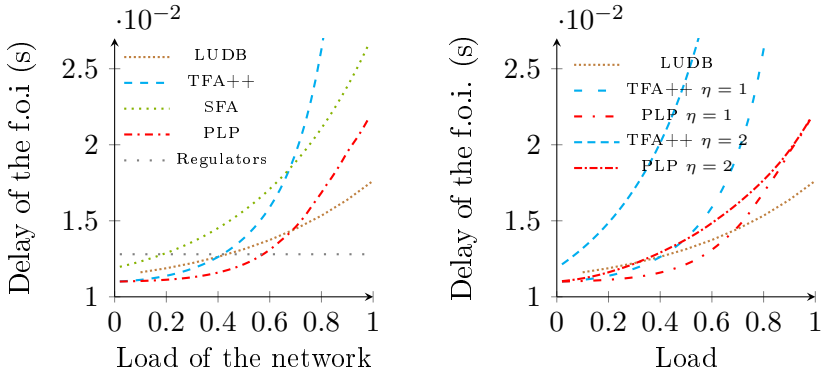
Figure 16b compares the execution time of the two linear programming methods. One can check that PLP, whose execution time is below 5 seconds for 25 servers, scales much better than ELP, whose execution time is already 10 seconds for 7 servers. One can also notice that LUDB is not tractable, and the computation could not be done is less than minute for networks with size at least 15. To obtain bounds for networks larger than 15, we use the option `ludb-heuristic 1`, whose computation time is much lower, but still exponential, and PLP would be more efficient for network with sizes at least 25.



(a) Delay. (b) Execution time.

Figure 16: Comparison of different methods for interleaved tandems when varying the number of servers.

Figure 17a shows the delay bounds obtained for a network of 10 servers in function of the load. We discard the ELP method. The first observation is that SFA is always outperformed by one method from the state of the art (LUDB or TFA++). TFA++ is better that LUDB for very small loads (less than 0.38), but behaves very poorly for high loads. PLP is better that LUDB for loads less than 0.7, and then becomes worse than LUDB: the TFA++ constraints do not enable to improve the delay bounds anymore. This effect is even more visible on Figure 17b that shows the sensitivity of the delay to the shaping rate. We here compare the servers for shaping rates ηR, for $\eta \in \{1, 2\}$. When $\eta = 2$. TFA++ is completely outperformed by LUDB and PLP. While TFA++ seems very sensitive to the shaping rate, the delays computed with PLP do not vary much, but LUDB is almost always better. Nevertheless, PLP is less sensitive to the shaping rate than TFA++.



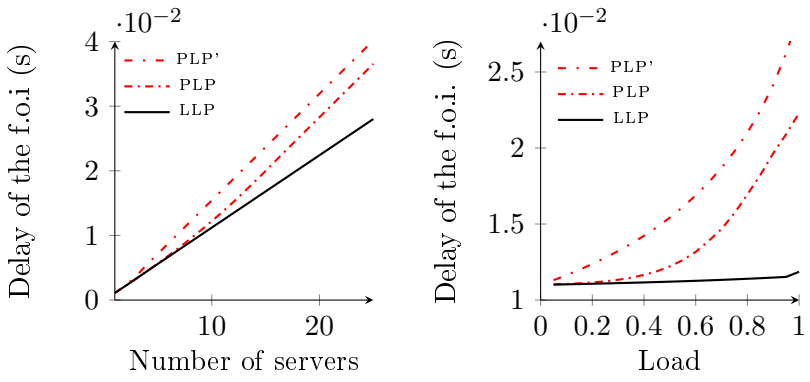
(a) Delay for $\eta = 1$.

(b) Shaping effect.

Figure 17: Comparison of different methods for interleaved tandems in function of the load.

Let us now focus more precisely on the different type of constraints of PLP. As mentioned in Section 4, compared to ELP, some constraints have been removed (service constraints), and some added (SFA and TFA++ and shaping constraints). Let us denote by PLP' the linear program obtained from PLP by removing the SFA and TFA++ constraints. Figure 18a compares different versions of linear programs to see the impacts of the two types of added constraints: the SFA and TFA++ ones on the one hand and the shaping ones on the other hand. To quantify the pessimism of the bounds, we also depict the lower bound derived in [12] (named LLP in the figure). Note that this lower bound of the worst-case delay is valid only when there is no shaper, but still this might give some insight on the accuracy of the bounds. We compare the bounds when the number of servers varies (with $U = 0.5$) and when the load varies (with $n = 10$). We depicted only PLP and PLP'. The effect of shaping in this example is negligible. PLP' is 10% higher than PLP for 25 servers in Figure 18a and 29% for a load of 1 in Figure 18b. PLP is 30 % higher than the lower bound LLP for 25 servers and 90% higher for 10 servers and a load $U = 1$.

Finally, in order to investigate the trade-off between tractability and accuracy of the bounds, let us consider a network with 100 servers. In this case, the tractability limit is reached when applying directly the linear program of Section 4. Instead, we use the results of Section 5, and cut the tandem in smaller pieces. This procedure can be done for both PLP and ELP. We take $\eta = 1$ and $U = 0.5$. The delay obtained with SFA is 0.96 s, and with TFA++, 1.83 s. In that case, SFA is better than TFA++. Deborah is limited to networks with size at most 63, so LUDB is not computed. Figure 19a, we compute the delays when the tandem is decomposed into sub-tandems of a given length (with a potential shorter first sub-tandem), and we vary that length. Figure 19b depicts the execution time. We stop when the execution time is more than

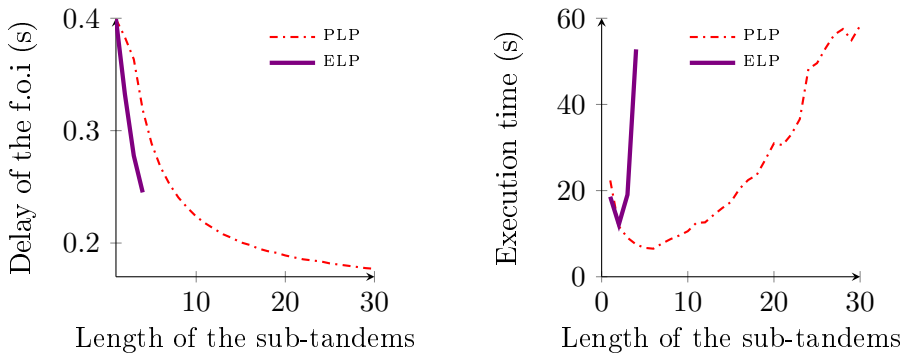


(a) Delay in function of the number of servers

(b) Delay in function of the load

Figure 18: Improvement obtained when adding the SFA and TFA++ constraints in interleaved tandems.

1 minute. First, one can notice that in both cases, the delay bounds decrease when the length of the sub-tandem decrease. Next, in PLP, the execution time first decreases (less very short sub-tandem to analyze), and then increases. The execution time for decomposition of length 1 is more efficient with ELP than PLP because SFA and TFA++ bounds are computed then added to PLP. With ELP, we can decompose the tandem in sub-tandems of length 4 at most, and obtain a delay of 0.24s. With PLP, we can decompose tandems in sub-tandems of length up to 30, and obtain a delay of 0.18 s (ELP is then 33% higher than PLP), it was possible to obtain a delay of 0.24 s by decomposing into sub-tandems of length 8, and the bound was computed in 8.7 s.



(a) Delay bound.

(b) Execution time.

Figure 19: Delay and execution time for ELP and PLP for an interleaved tandem of length 100 when decomposing into sub-tandems.

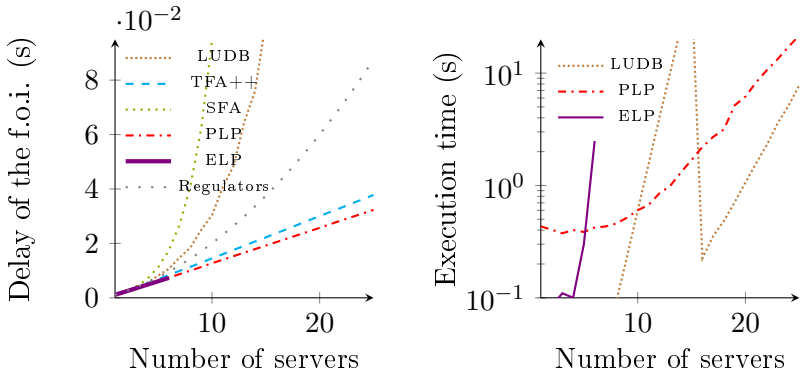
Source-sink network The shaping effect is limited with interleaved networks. We turn to another example where the shaping might have more effect, and change the comparison between the bounds (in particular TFA++ and LUDB versus SFA). We call source-sink tandem a tandem with n servers and $2n-1$ flows. Each flow either starts at server 1 or ends at server n. In the uniform case, there is one flow per possible path, as depicted in Figure 20 with $n = 4$. There are n flows crossing each server, so the load of the network is nr/R.

Figure 21a depicts the delay bounds computed by each method when the length of the

Figure 20: Source-sink tandem of length 4.

tandem grows from 1 to 25 when $\eta = 1$ and $U = 0.5$. One can still check that the linear programming methods give the best delay bounds. Here, the gap between the ELP and PLP methods is very small. The TFA++ also performs very well: for 25 severs it is only 17% higher than PLP. These three bounds are below the bound obtained with regulators. SFA and LUDB are completely outperformed. Indeed, at each server, $n-1$ flows continue to the next server. Then the shaping has a very strong effect on the performances.

Figure 21b compares the execution times of ELP and PLP. Again we see the tractability improvement of this new approach compared to ELP.



(a) Delay.

(b) Execution time.

Figure 21: Comparison of different methods for source-sink networks in function of the number of servers.

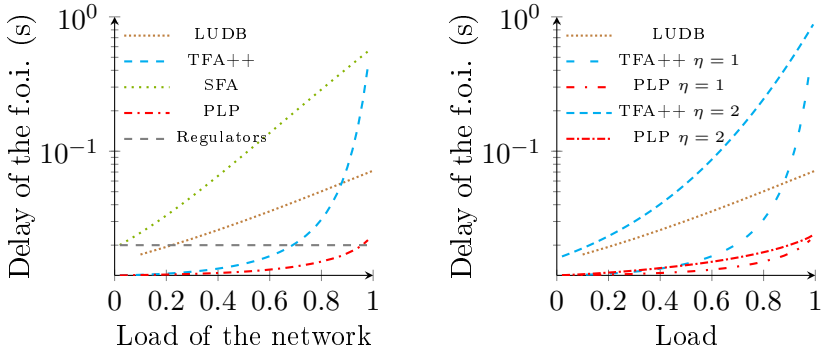
Figure 22a shows how the delay bounds grow with the load of a tandem of length 10. When the load is small, TFA++ and PLP are similar, but the gap between the two grows exponentially fast. For example, TFA++ is 13% higher than PLP for a load of 0.5 and 105% for a load of 0.8. PLP is smaller than the delay bound with regulators until a load of 0.95.

Figure 22b compares the delay bounds for shaping service rates, with $\eta \in \{1, 2\}$. Again, the PLP method is not very sensitive to this parameter, contrary to the TFA++ method. LUDB outperforms TFA++ when $\eta = 2$.

7.1.2 Mesh network

Consider the mesh network of Figure 23. There is one flow per path from server 0 or 1 to server 8, which represents a total of 16 paths. Servers 0 to 7 have the same characteristics as above, and server 8's service rate is $2R$, as there are twice as many flows crossing it compared to the other servers. We also keep the same characteristics as above for the flows.

Figures 24 compares the delays obtained for TFA++, SFA and the two different methods introduced for analyzing feed-forward networks: network unfolding and flow cutting. Due to its computation time, we do not compare with ELP. Figure 24a depicts the delays when $\eta = 1$ and Figure 24b when $\eta = 5$. Similar to the previous cases, TFA++ is very accurate when $\eta = 1$ and the load is small, but becomes pessimistic for larger values of η or higher loads. It is not surprising that the unfolding the network leads to tighter delay bounds than cutting the



(a) Delay for $\eta = 1$.

(b) Shaping effect.

Figure 22: Comparison of different methods for source-sink networks in function of the load.

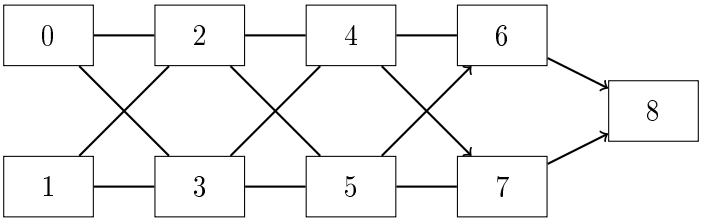
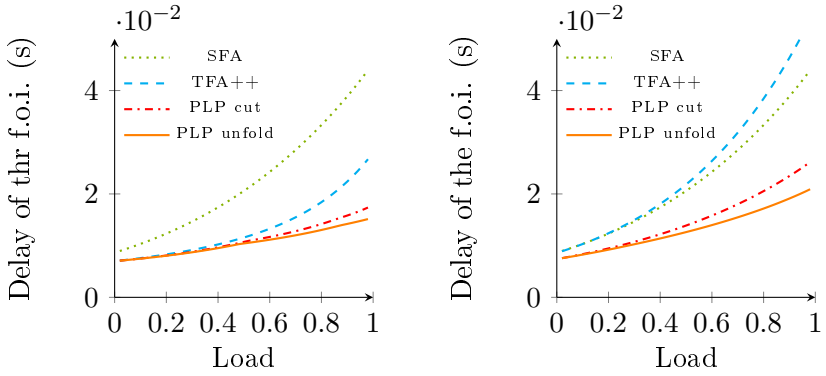


Figure 23: Mesh network.

network. Indeed, cutting a flow induces some over-approximations. But the unfolded network's size being exponential in the size of the original network, this method is not scalable. We notice that the gap between the two methods is not very large, specially when $\eta = 1$.



(a) $\eta = 1$.

(b) $\eta = 5$.

Figure 24: Comparison of different methods for a mesh network in function of the load.

7.1.3 Ring network

Consider a ring network, as depicted on Figure 25 for $n = 4$. There are n flows of length n in a ring of length n. If the arrival rate is r, then the load of the network is $U = nr/R$. Figure 26a shows the worst-case delay bound for the different methods when the number of servers grows from 2 to 10 (and to 4 for ELP). The load of the network is 0.5 and $\eta = 1$. One can see that the bounds found for ELP and PLP are really close to one another and the gap with TFA++ of the same order as in the previous topologies. Here again, SFA gives very inaccurate bounds.

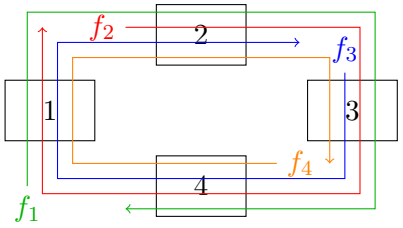
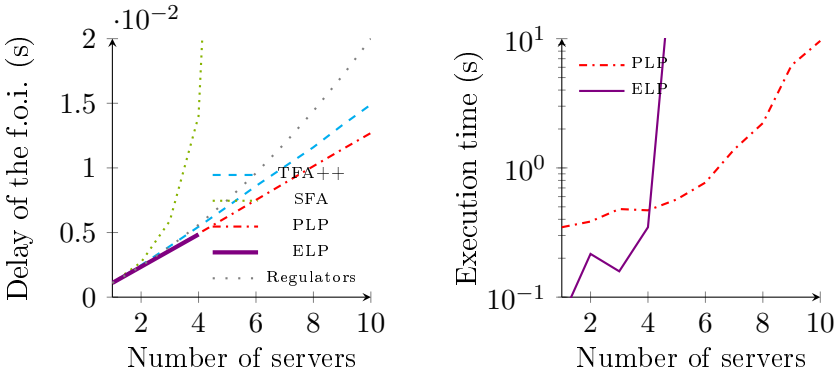


Figure 25: Ring network with $n = 4$.

This is inline with the example of the source-sink tandem: at each server $n-1$ flows are shaped together, which makes TFA++ very efficient. The execution time of ELP and PLP is depicted in Figure 26b. One can see that PLP takes longer to compute, because computing the fixed point requires solving a much larger linear program than for tandem networks. More precisely, three linear programs need to solved: 1) compute the TFA++ delays (using Paragraph 3.1); 2) compute the fixed point as the optimal solution of the linear program of Equation (3); 3) compute the delay bound of the flow of interest. The second linear program to solves takes most of the computation time. For networks of respective size 3, 5 and 10, the computation of TFA++ delays take respectively 0.007 s, 0.008 s, 0.018 s, the computation of Equation (3) takes 0.009 s, 0.081 s and 8.679 s, and the computation of the delay of the flow of interest takes 0.007 s, 0.014 s and 0.103 s.

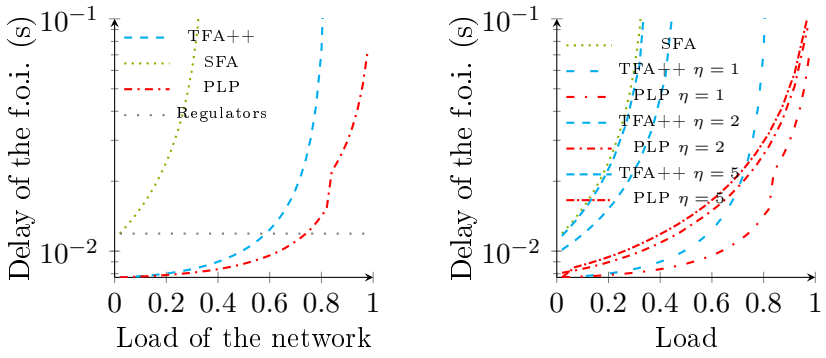
Figure 27a compares the different approaches (except ELP) when the load of the ring of length 7 grows from 0 to 1. Similar to the previous examples, PLP computes much tighter bounds than TFA++ for hight loads, and has a larger stability region (local stability). The influence of TFA++ on PLP is also more visible: when the TFA++ delay bounds become infinite, the delay bounds of PLP increases faster. Again, Figure 27b shows how the performances evolve when the shaping rate of the servers grows, for $\eta \in \{1, 2, 5\}$. When $\eta = 5$, the delays of TFA++ are comparable with SFA. With TFA++, the stability region also decreases with η: the sufficient conditions for the stability computed by TFA++ for $\eta = 1, 2, 5$ are respectively $U < 0.85$, $U < 0.55$ and $U < 0.38$. From PLP, the local stability condition seems sufficient to ensure the global stability in ring networks. In this example, the stability condition is also improved compared to the one of Rizzo and Le Boudec in [23], where for a ring of length 6, the stability condition is $U < 0.9$.



(a) Delay.

(b) Execution time.

Figure 26: Comparison of different methods for the ring network in function of the number of servers.



(a) Delay. (b) Shaping effect.

Figure 27: Comparison of different methods for the ring network in function of the load.

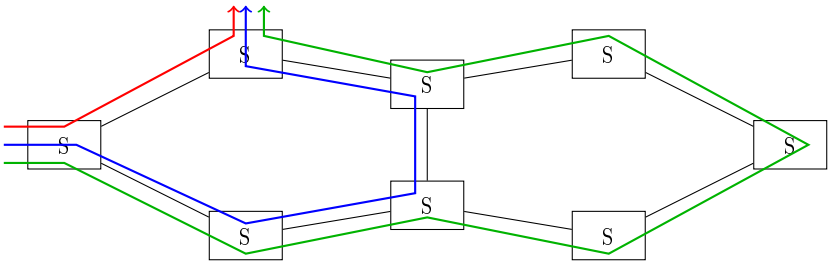


Figure 28: Carrier network with three types of paths: direct paths (red), one-ring path (blue), two-ring path(green).

7.2 Realistic examples

7.2.1 Carrier network

Consider the network of Figure 28. It is made of two bidirectional rings. There are 8 flows departing from each router, 4 of them going to each of the neighbors (except the two central nodes that are not considered as neighbors). Two flows follow a direct path (path of length 1) and one follows a path along one ring and the last path follows the two rings, as depicted on Figure 28. Links are bidirectional, so Figure 28 is not the exact representation of the network, that we do not give here for the sake of readability: there are 18 servers and 64 flows in this network. In the example, all packets have length $L = 1024\,\mathrm{b}$ and are periodically sent for each flow at period $P = 125\,\mu\mathrm{s}$. Then each flow is constrained by the arrival curve $\alpha : t \mapsto B + tT/P$. The service curve of each link is $\beta : t \mapsto R(t - L/R)$. The greedy-shaper defined to take into account the packetization at each router is $\sigma : t \mapsto Ct + L$. The problem is to find the rate R to allocate to these flows so that a maximum delay bound is satisfied for all flows. Let us assume that the target delay is $75\,\mu\mathrm{s}$.

Table 6 shows the rates to be allocated in two scenarios: first when *hard slicing* is at work, that is $C = R$, and the other when *soft slicing* is at work, that is $C = 10\,\mathrm{Gb/s}$ is the total capacity of the links.

	TFA++	PLP	ELP
$R = C$	690 Mb/s (0.166 s)	650 Mb/s (112 s)	510 Mbps (24 min)
R, $C = 10\,\mathrm{Gb/s}$	1.350 Gb/s (0.2 s)	1.0 Gb/s (130 s)	540 Mbps (10 min)

Table 6: Service rate required to guarantee the maximum delay of $75\,\mu\mathrm{s}$, with different methods.

One can observe that in the case of hard-slicing, TFA++ compares well with PLP (it is only 6% higher). This can be explained because to obtain a delay bound as small as $75\mu s$, the load of the network is small (approximately 15%). But the TFA++ is 35% higher than ELP. In this scenario, a 24 minute computation might not be considered too costly given the gain on the bandwidth. In the case of soft slicing, TFA++ is 35% higher than PLP, and 150% higher than ELP.

7.2.2 Smart-Campus network

In this example, the network topology is a sink tree. Four classes of flows are circulating in the network, from the leaves to the root, as depicted in Figure 29, every class of traffic has flows following all paths, and the service policy is FIFO per class. Among the four classes, the DRR scheduling is at work, and each flow is offered the same guarantee, that is 25% of the service rate, and the quantum assigned to each class is Q. If a network element has service curve $\beta = \beta_{R,0}$, the DRR service curve for class i is, according to Boyer et al. [7], $\beta_i^{DRR} = \beta_{R/4,T_i}$, with $T_i = [3Q(Q + \ell_i) + (L - \ell_i)]/R$ where ℓ_i is the maximum length of a packet of class i, and $L = \sum_{i=1}^{4} \ell_i$. The service rate of each server is given (in Gb/s) on Figure 29 and we take $Q = 16\,\text{kb}$. The characteristics of each class of flow is given in Table 7. Moreover, there is a shaping for each class of flow (separately) at the entrance of the network, at rate 1 Gb/s. The shaper is then $t \mapsto 10^9 t + \ell_i$ for class i.

Class	burst	arrival rate	packet size
Electric protection	42.56 kb	8.521 Mb/s	3040 b
Virtual reality game	2.16 Mb	180 Mb/s	12 kb
Video conference	3.24 Mb	162 Mb/s	12 kb
4K video	7.2 Mb	180 Mb/s	12 kb

Table 7: Characteristics of the 4 classes of flows.

Table 8 summarizes the delay found for each class and each method. In order to make all methods more tractable and accurate, we first concatenate servers that are crossed by the same sets of flows. This enables ELP to compute the delays fast, which would not be possible otherwise.

This is a case where LUDB can compute tight delay bounds, when shaping is not taking into account. This bound then gives an idea of the effect of shaping. SFA also does not take into account the shaping effect, and SFA is then larger than LUDB.

The other bounds take the shaping into account, and we can see TFA++ are slightly better than LUDB, but is still twice as large as ELP and PLP. The PLP provides a good approximation of ELP. We also see that ELP and PLP provide good approximations of the actual worst-case: the last column of Table 8 represents the delay obtained when the network is simulated and the maximum traffic arrived from time 0. The gap can also be explained by the fact that the DRR service curve can be pessimistic and that this trajectory may not be the one maximizing the delays.

Last, to assess the usefulness of both algebraic and shaping constraints, in Table 9 shows the different delay bounds obtained by PLP when removing shaping and/or SFA and TFA++ constraints. One can observe that both types of new sets of constraints improve the bounds. Moreover, the improvement obtained by combining of these two types of constraints is drastic.

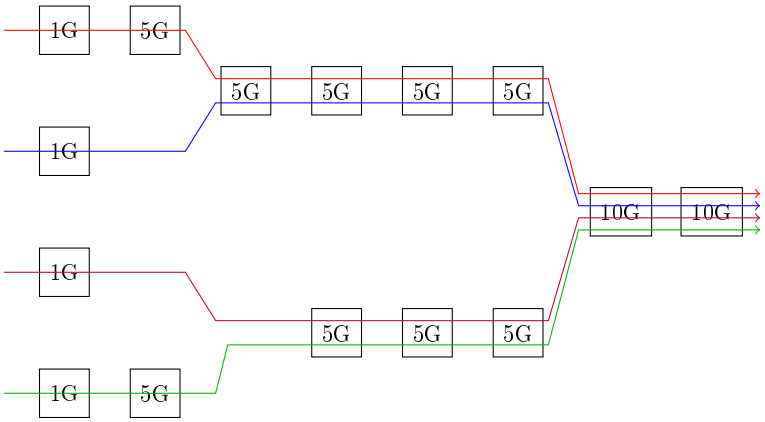


Figure 29: Smart campus network: four classes of traffic arrive according to every path drawn.

class	TFA++	SFA	PLP	ELP	LUDB	simulation
EP	200	240	170	166	223	33
VR	5066	7005	2614	2550	5760	1911
VC	7402	10373	3775	3659	8567	3171
4KV	16482	22995	8289	8082	18864	7147

Table 8: Delays (in μs) with for the four classes of traffic with the different methods, and the simulation of a candidate trajectory for the worst-case delay.

class	PLP	PLP (no shaping)	PLP'	PLP' (no shaping)
EP	170	222	185	224
VR	2614	5801	4114	6551
VC	3775	8615	5869	9600
4KV	8289	18000	13071	21444

Table 9: Delays (in μs) obtained from PLP by removing shaping and/or SFA and TFA++ constraints.

8 Conclusion

In this paper, we have proposed a new linear program to compute performance bounds in FIFO networks. This linear program offers a good trade-off between accuracy of the bounds and tractability. While it does not lead to performance bounds as accurate as the linear programs of [13], it can be performed in polynomial time, which enables to use it in larger networks from realistic cases. This new algorithm also improves the performances bounds compared to the algebraic methods from the literature (SFA, TFA++). Interestingly, this improvement is obtained by introducing these algebraic bounds in the linear program.

We also presented a linear-programming solution to deal with networks having cyclic dependencies. Although presented for FIFO networks, this solution is valid for the other linear-programming methods used in other models of Network Calculus (e.g., blind multiplexing in [40]). This method improves both the delay bounds and the stability region of the literature.

Comparison with other scalable methods (TFA++, SFA) also enables to have a more precise knowledge of when these bounds are accurate. While SFA is never accurate, we can exhibit cases where TFA++ provides accurate performance guarantees: when the load of the network is small or medium and the service and shaping rates coincide.

Through the example of TFA++, it was already known that shaping plays a very important in reducing the performance bounds. We show it is also the case with the linear-programming techniques. One research direction would also be to see if SFA can be adapted to take into account the shaping effect into a SFA++ method. Some work has already been done in this direction [48], and when the flow of interest is shaped at the input of the network. The effect of the shaping of the cross-traffic still requires investigation.

The trade-off between accuracy and tractability presented in this paper is a first step toward obtaining good trade-offs between accuracy and scalability. Many questions remain open for scalability. First, the PLP algorithm is tractable, but it might not yet be usable directly for large network. We showed one example where it can be used for analyzing networks with 100 servers by decomposing it in sub-networks. This approach has to be generalized. In particular, it raises the issue of finding good decompositions of networks. Several directions can be investigated: finding good heuristics, use backtracking algorithms as in [42], or use deep-learning approaches as in [46].

Last, one limitation of the linear-programming methods is that they require concave arrival curves. Taking into account the packetization effect and the length of packets can improve the performance bounds. This was demonstrated in [49, 50] for priorities and GPS, and recently in [51] for WRR. In [10], the length of the packets of the bit of interest is also taken into account. Packetization is also taken into account in [6] to compare different methods in FIFO networks. While some packetization effects can be integrated in TFA++ constraints, future work will also focus on the introduction of packetization constraints.

Acknowledgement

The author would like to thank the anonymous reviewers for their precious commments that helped to greatly improve the manuscript.

References

[1] Time-Sensitive Networking Task Group, http://www.ieee802.org/1/pages/tsn.html (2017).

[2] R. Cruz, Quality of Service Guarantees in Virtual Circuit Switched Networks, IEEE Journal on selected areas in communication 13 (1995) 1048–1056. doi:10.1109/49.400660.

[3] J. M. McManus, K. W. Ross, Video-on-demand over ATM: Constant-rate Transmission and Transport, IEEE J.Sel. A. Commun. 14 (6) (1996) 1087–1098. doi:10.1109/49.508280.

[4] M. Boyer, N. Navet, X. Olive, É. Thierry, The PEGASE project: precise and scalable temporal analysis for aerospace communication systems with Network Calculus, in: ISOLA'10, 2010, pp. 122–136. doi:10.1007/978-3-642-16558-0_13.

[5] E. Mohammadpour, E. Stai, M. Mohiuddin, J.-Y. Le Boudec, Latency and Backlog Bounds in Time-Sensitive Networking with Credit Based Shapers and Asynchronous Traffic Shaping, in: 30th International Teletraffic Congress, ITC, 2018, pp. 1–6. doi:10.1109/ITC30.2018.10053.

[6] L. Zhao, P. Pop, Z. Zheng, H. Daigmorte, M. Boyer, Latency Analysis of Multiple Classes of AVB Traffic in TSN with Standard Credit Behavior using Network Calculus, CoRR abs/2005.08256 (2020). arXiv:2005.08256.
URL https://arxiv.org/abs/2005.08256

[7] M. Boyer, G. Stea, W. M. Sofack, Deficit Round Robin with network calculus, in: B. Gaujal, A. Jean-Marie, E. A. Jorswieck, A. Seuret (Eds.), 6th International ICST Conference on Performance Evaluation Methodologies and Tools, ICST/IEEE, 2012, pp. 138–147. doi:10.4108/valuetools.2012.250202.
URL https://doi.org/10.4108/valuetools.2012.250202

[8] E. Mohammadpour, E. Stai, J.-Y. Le Boudec, Improved Credit Bounds for the Credit-Based Shaper in Time-Sensitive Networking, CoRR abs/1901.04957 (2019). arXiv:1901.04957.
URL http://arxiv.org/abs/1901.04957

[9] J.-Y. Le Boudec, A Theory of Traffic Regulators for Deterministic Networks With Application to Interleaved Regulators, IEEE/ACM Trans. Netw. 26 (6) (2018) 2721–2733. doi:10.1109/TNET.2018.2875191.
URL https://doi.org/10.1109/TNET.2018.2875191

[10] L. Thomas, J.-Y. Le Boudec, A. Mifdaoui, On Cyclic Dependencies and Regulators in Time-Sensitive Networks, in: IEEE Real-Time Systems Symposium, RTSS 2019, Hong Kong, SAR, China, December 3-6, 2019, IEEE, 2019, pp. 299–311. doi:10.1109/RTSS46320.2019.00035.
URL https://doi.org/10.1109/RTSS46320.2019.00035

[11] A. Mifdaoui, T. Leydier, Beyond the Accuracy-Complexity Tradeoffs of Compositional Analyses using Network Calculus for Complex Networks, in: 10th International Workshop on Compositional Theory and Technology for Real-Time Embedded Systems (co-located with RTSS 2017), 2017, pp. 1–8.

[12] A. Bouillard, G. Stea, Exact worst-case delay for FIFO-multiplexing tandems, in: B. Gaujal, A. Jean-Marie, E. A. Jorswieck, A. Seuret (Eds.), 6th International ICST Conference on Performance Evaluation Methodologies and Tools, Cargese, Corsica, France, October 9-12, 2012, ICST/IEEE, 2012, pp. 158–167. doi:10.4108/valuetools.2012.250090.
URL https://doi.org/10.4108/valuetools.2012.250090

[13] A. Bouillard, G. Stea, Exact Worst-Case Delay in FIFO-Multiplexing Feed-Forward Networks, IEEE/ACM Trans. Netw. 23 (5) (2015) 1387–1400. doi:10.1109/TNET.2014.2332071.
URL https://doi.org/10.1109/TNET.2014.2332071

[14] J. Zhang, L. Chen, T. Wang, X. Wang, Analysis of TSN for Industrial Automation based on Network Calculus, in: 24th IEEE International Conference on Emerging Technologies and Factory Automation, ETFA 2019, Zaragoza, Spain, September 10-13, 2019, IEEE, 2019, pp. 240–247. doi:10.1109/ETFA.2019.8869053.
URL https://doi.org/10.1109/ETFA.2019.8869053

[15] D. Tamas-Selicean, P. Pop, W. Steiner, Design optimization of TTEthernet-based distributed real-time systems, Real-Time Systems 51 (1) (2015) 1–35.

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

[17] A. Bouillard, M. Boyer, E. Le Corronc, Deterministic Network Calculus: From Theory to Practical Implementation, ISTE, 2018.

[18] J.-Y. Le Boudec, P. Thiran, Network Calculus: A Theory of Deterministic Queuing Systems for the Internet, Vol. LNCS 2050, Springer-Verlag, 2001, revised version 4, May 10, 2004. doi:10.1007/3-540-45318-0.

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

[20] A. Bouillard, Stability and Performance Bounds in Cyclic Networks Using Network Calculus, in: É. André, M. Stoelinga (Eds.), Formal Modeling and Analysis of Timed Systems - 17th International Conference, FORMATS 2019, Amsterdam, The Netherlands, August 27-29, 2019, Proceedings, Vol. 11750 of Lecture Notes in Computer Science, Springer, 2019, pp. 96–113. doi:10.1007/978-3-030-29662-9_6.
URL https://doi.org/10.1007/978-3-030-29662-9_6

[21] S. Bondorf, J. Schmitt, Improving Cross-Traffic Bounds in Feed-Forward Networks – There is a Job for Everyone, in: A. Remke, B. R. Haverkort (Eds.), Measurement, Modelling and Evaluation of Dependable Computer and Communication Systems, Springer International Publishing, Cham, 2016, pp. 9–24.

[22] N. Deo, Graph theory : with applications to engineering and computer science, Prentice-Hall of India ; Prentice-Hall International, 1994.
URL http://www.worldcat.org/isbn/9788120301450

[23] G. Rizzo, J.-Y. Le Boudec, Stability and Delay Bounds in Heterogeneous Networks of Aggregate Schedulers, in: INFOCOM 2008. 27th IEEE International Conference on Computer Communications, Joint Conference of the IEEE Computer and Communications Societies, 13-18 April 2008, Phoenix, AZ, USA, IEEE, 2008, pp. 1490–1498. doi:10.1109/INFOCOM.2008.208.
URL https://doi.org/10.1109/INFOCOM.2008.208

[24] M. Andrews, Instability of FIFO in the permanent sessions model at arbitrarily small network loads, ACM Trans. Algorithms 5 (2007) 33:1–33:29.

[25] A. Charny, J.-Y. Le Boudec, Delay Bounds in a Network with Aggregate Scheduling, in: J. Crowcroft, J. Roberts, M. I. Smirnov (Eds.), Quality of Future Internet Services, First COST 263 International Workshop, QofIS 2000, Berlin, Germany, September 25-26, 2000, Proceedings, Vol. 1922 of Lecture Notes in Computer Science, Springer, 2000, pp. 1–13. doi:10.1007/3-540-39939-9_1.
URL https://doi.org/10.1007/3-540-39939-9_1

[26] G. Rizzo, J.-Y. Le Boudec, Generalization of the RIN Result to Heterogeneous Networks of Aggregate Schedulers and Leaky Bucket Constrained Flows, in: Proceedings of the 15th IEEE International Conference on Networks, ICON 2007, 19-21 November 2007, Adelaide, Australia, IEEE, 2007, pp. 388–393. doi:10.1109/ICON.2007.4444118.
URL https://doi.org/10.1109/ICON.2007.4444118

[27] A. Amari, A. Mifdaoui, Worst-case timing analysis of ring networks with cyclic dependencies using network calculus, in: RTCSA, 2017, pp. 1–10. doi:10.1109/RTCSA.2017.8046319.

[28] J. Grieu, Analyse et évaluation de techniques de commutation Ethernet pour l'interconnexion des systèmes avioniques (September 2004).
URL https://oatao.univ-toulouse.fr/7385/

[29] M. Boyer, A. Graillat, B. D. de Dinechin, J. Migge, Bounding the delays of the MPPA network-on-chip with network calculus: Models and benchmarks, Perform. Evaluation 143 (2020) 102124. doi:10.1016/j.peva.2020.102124.
URL https://doi.org/10.1016/j.peva.2020.102124

[30] L. Bisti, L. Lenzini, E. Mingozzi, G. Stea, DEBORAH: A Tool for Worst-Case Analysis of FIFO Tandems, in: T. Margaria, B. Steffen (Eds.), Leveraging Applications of Formal Methods, Verification, and Validation, Vol. 6415 of LNCS, Springer, 2010, pp. 152–168.

[31] L. Bisti, L. Lenzini, E. Mingozzi, G. Stea, Estimating the worst-case delay in FIFO tandems using network calculus, in: ValueTools '08, 2008, pp. 67:1–67:10.

[32] L. Lenzini, E. Mingozzi, G. Stea, End-to-end Delay Bounds in FIFO-multiplexing Tandems, in: Proceedings of the 2nd International Conference on Performance Evaluation Methodologies and Tools (Valuetools'07), 2007, pp. 1–10.

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

[34] L. Bisti, L. Lenzini, E. Mingozzi, G. Stea, Numerical analysis of worst-case end-to-end delay bounds in FIFO tandem networks, Real Time Syst. 48 (5) (2012) 527–569. doi:10.1007/s11241-012-9153-1.
URL https://doi.org/10.1007/s11241-012-9153-1

[35] M. Fidler, Extending the Network Calculus Pay Bursts Only Once Principle to Aggregate Scheduling, in: M. A. Marsan, G. Corazza, M. Listanti, A. Roveri (Eds.), Quality of Service in Multiservice IP Networks, Second International Workshop, QoS-IP 2003, Milano, Italy, February 24-26, 2003, Proceedings, Vol. 2601 of Lecture Notes in Computer Science, Springer, 2003, pp. 19–34. doi:10.1007/3-540-36480-3_2.
URL https://doi.org/10.1007/3-540-36480-3_2

[36] A. Bouillard, B. Gaujal, S. Lagrange, É. Thierry, Optimal routing for end-to-end guarantees: the price of multiplexing, in: P. W. Glynn (Ed.), Proceedings of the 2nd International Conference on Performance Evaluation Methodolgies and Tools, VALUETOOLS

2007, Nantes, France, October 22-27, 2007, ACM International Conference Proceeding Series, ICST/ACM, 2007, p. 62. doi:10.4108/valuetools.2007.1949.
URL https://doi.org/10.4108/valuetools.2007.1949

[37] A. Bouillard, B. Gaujal, S. Lagrange, É. Thierry, Optimal routing for end-to-end guarantees using Network Calculus, Performance Evaluation 65 (11-12) (2008) 883–906. doi:10.1016/j.peva.2008.04.008.

[38] J. B. Schmitt, F. A. Zdarsky, I. Martinovic, Improving Performance Bounds in Feed-Forward Networks by Paying Multiplexing Only Once, in: F. Bause, P. Buchholz (Eds.), Proceedings 14th GI/ITG Conference on Measurement, Modelling and Evaluation of Computer and Communication Systems (MMB 2008), March 31 - April 2, 2008, Dortmund, Germany, VDE Verlag, 2008, pp. 13–28.

[39] J. B. Schmitt, F. A. Zdarsky, M. Fidler, Delay Bounds under Arbitrary Multiplexing: When Network Calculus Leaves You in the Lurch..., in: INFOCOM 2008. 27th IEEE International Conference on Computer Communications, Joint Conference of the IEEE Computer and Communications Societies, 13-18 April 2008, Phoenix, AZ, USA, IEEE, 2008, pp. 1669–1677. doi:10.1109/INFOCOM.2008.228.
URL https://doi.org/10.1109/INFOCOM.2008.228

[40] A. Bouillard, L. Jouhet, É. Thierry, Tight Performance Bounds in the Worst-Case Analysis of Feed-Forward Networks, in: INFOCOM 2010. 29th IEEE International Conference on Computer Communications, Joint Conference of the IEEE Computer and Communications Societies, 15-19 March 2010, San Diego, CA, USA, IEEE, 2010, pp. 1316–1324. doi:10.1109/INFCOM.2010.5461912.
URL https://doi.org/10.1109/INFCOM.2010.5461912

[41] A. Bouillard, T. Nowak, Fast symbolic computation of the worst-case delay in tandem networks and applications, Perform. Eval. 91 (2015) 270–285. doi:10.1016/j.peva.2015.06.016.

[42] S. Bondorf, P. Nikolaus, J. B. Schmitt, Quality and Cost of Deterministic Network Calculus: Design and Evaluation of an Accurate and Fast Analysis, Proc. ACM Meas. Anal. Comput. Syst. 1 (1) (2017) 16:1–16:34. doi:10.1145/3084453.
URL https://doi.org/10.1145/3084453

[43] S. Bondorf, J. B. Schmitt, Calculating Accurate End-to-End Delay Bounds – You Better Know Your Cross-Traffic, in: Proceedings of the 9th International Conference on Performance Evaluation Methodologies and Tools (VALUETOOLS 2015), 2015, pp. 1–1.

[44] S. Bondorf, J. B. Schmitt, Improving cross-traffic bounds in feed-forward networks - there is a job for everyone, in: A. Remke, B. R. Haverkort (Eds.), Measurement, Modelling and Evaluation of Dependable Computer and Communication Systems - 18th International GI/ITG Conference, MMB & DFT 2016, Münster, Germany, April 4-6, 2016, Proceedings, Vol. 9629 of Lecture Notes in Computer Science, Springer, 2016, pp. 9–24. doi:10.1007/978-3-319-31559-1_3.
URL https://doi.org/10.1007/978-3-319-31559-1_3

[45] A. Bouillard, A. Junier, Worst-case delay bounds with fixed priorities using network calculus, in: S. Lasaulce, D. Fiems, P. G. Harrison, L. Vandendorpe (Eds.), 5th International ICST Conference on Performance Evaluation Methodologies and Tools Communications, VALUETOOLS '11, Paris, France, May 16-20, 2011, ICST/ACM, 2011, pp. 381–390.

doi:10.4108/icst.valuetools.2011.245603.
URL https://doi.org/10.4108/icst.valuetools.2011.245603

[46] F. Geyer, S. Bondorf, Deeptma: Predicting effective contention models for network calculus using graph neural networks, in: 2019 IEEE Conference on Computer Communications, INFOCOM 2019, Paris, France, April 29 - May 2, 2019, IEEE, 2019, pp. 1009–1017. doi:10.1109/INFOCOM.2019.8737496.
URL https://doi.org/10.1109/INFOCOM.2019.8737496

[47] J. L. Gearhart, K. L. Adair, J. D. Durfee, K. A. Jones, N. Martin, R. J. Detry, Comparison of open-source linear programming solvers (2013).

[48] M. Boyer, Half-modeling of shaping in FIFO net with network calculus, in: 18th International Conference on Real-Time and Network Systems, Toulouse, France, 2010, pp. 59–68.
URL https://hal.archives-ouvertes.fr/hal-00544502

[49] W. M. Sofack, M. Boyer, Non Preemptive Static Priority with Network Calculus: Enhancement, in: J. B. Schmitt (Ed.), Measurement, Modelling, and Evaluation of Computing Systems and Dependability and Fault Tolerance - 16th International GI/ITG Conference, MMB & DFT 2012, Kaiserslautern, Germany, March 19-21, 2012. Proceedings, Vol. 7201 of Lecture Notes in Computer Science, Springer, 2012, pp. 258–272. doi:10.1007/978-3-642-28540-0_22.
URL https://doi.org/10.1007/978-3-642-28540-0_22

[50] W. M. Sofack, M. Boyer, Generalisation of GPS and P-GPS in network calculus, in: T. Nolte, A. Willig (Eds.), 9th IEEE International Workshop on Factory Communication Systems, WFCS 2012, Lemgo, NRW, Germany, May 21-24, 2012, IEEE, 2012, pp. 169–172. doi:10.1109/WFCS.2012.6242563.
URL https://doi.org/10.1109/WFCS.2012.6242563

[51] S. M. Tabatabaee, J.-Y. Le Boudec, M. Boyer, Interleaved Weighted Round-Robin: A Network Calculus Analysis, CoRR abs/2003.08372 (2020). arXiv:2003.08372.
URL https://arxiv.org/abs/2003.08372

[52] J. Kennan, Uniqueness of Positive Fixed Points for Increasing Concave Functions on Rn: An Elementary Result, Review of Economic Dynamics 4 (2001) 893–899.

A Proof of Theorem 3

Let d be an upper bound of the delay in the server, and assume without loss of generality that the α-constrained flow is flow 1. Let us denote by A the aggregate arrival process and A_1 the arrival process of flow 1, and similarly for the departure processes D and D_1.

Let d be an upper bound of the delay in the server: for all t, $D(t) \geq A([t-d]_+)$. From Equation (2), we also have $D_1(t) \geq A_1([t-d]_+) = A_1 * \delta_d(t)$: δ_d is a service curve for flow 1.

As a consequence, Theorem 2 can be applied with service curve δ_d and greedy-shaper $\varepsilon : t \mapsto \infty$: $\alpha \oslash \delta_d \wedge \varepsilon = \alpha \oslash \delta_d$ is an arrival curve for the departure process D_1.

B Proof of Theorem 4

Let us denote by A the cumulative arrival process of the flow of interest in the system and D its departure process from the system. Fix s and t such that $s \leq t$. Our goal is to show that $D(t) - D(s) \leq B + r(t-s)$.

Let us first transform the arrival process A in A' so that $A'(u) = A(u)$ for all $u \leq s$ and $A'(u)$ is maximized for all $u > s$. As from (H_1), α is the only constraint for the flow, there exists $H \geq 0$ such that for all $u > t$, $A'(u) = A(s) + H + r(u-s)$.

Consider a departure process D' corresponding to the modified arrival process. As the systems is causal, $D'(u) = D(u)$ for all $u \leq s$.

We now distinguish two cases, depending on whether D' is right-continuous at s (note that since the processes are always left-continuous, this means that whether D' is continuous at t).

1) If D' is right-continuous at s, then $D(s) = D'(s) = D'(s^+)$ (where $D'(s^+) = \lim_{u \to s, s>u} D'(u)$), then the backlog at time s^+ with the modified arrival process A' satisfies $A'(s^+) - D'(s^+) \leq B$. Consequently, we have $B \geq A'(s^+) - D'(s^+) = A(s) + H - D(s)$ and

$$D(t) - D(s) \leq A(t) - D(s) \leq A'(t) - D(s) \leq A(s) + H + r(t-s) - D(s) \leq B + r(t-s).$$

2) If D' is not right-continuous at time s, we use hypothesis (H_2) to modify the departure process.

Let A'_n (resp. D'_n) be the aggregated arrival (resp. departure) process of server n with the modified arrival process A'. There exists D''_n that is modified according to (H_2) that is right-continuous at s and such that $D''_n(u) = D'_n(u)$ for all $u \leq s$. As D''_n is right-continuous at s and is the sum of departure processes, all departure processes are continuous at s, and in particular that of the f.o.i., that we denote D''. In particular, $D(s) = D''(s) = D''(s^+)$. Similar to the previous case, $B \geq A'(s^+) - D''(s^+) = A(s) + H - D(s)$. As a consequence,

$$D(t) - D(s) \leq A'(t) - D(s) \leq A(s) + H + r(t-s) - D(s) \leq B + r(t-s),$$

which finishes the proof. □

Let us now focus on a case where (H_2) holds: for each arrival process A_n, server n admits all departure processes D_n such that $A_n \geq D_n \geq A_n * \beta_n$ and $D_n * \sigma_n = D_n$ where the token-bucket greedy shaper σ_n is greater than β_n.

As β_n is continuous, there exists $v \leq s$ such that $D_n(s) \geq A_n(v) + \beta_n(s-v)$.

So, we can modify D_n from time s so that it is right-continuous at s: define D'_n as $D'_n(u) = D_n(u)$ for all $u \leq s$ and $D'_n(u) = \min(D_n(u), \max(D_n(s), A_n(v) + \beta_n(u-v)))$ for all $u > s$.

$D'_n \in \mathcal{F}$: a) $D'_n(0) = D_n(0) = 0$; b) D'_n is the combination left-continuous functions, so it is left-continuous; c) D'_n is non-decreasing on $[0, s]$ and $(s, +\infty)$. Moreover, for all $u > s$, $D'_n(u) \geq \min(D_n(s), D_n(u)) = D_n(s) = D'_n(s)$, so D'_n is non-decreasing.

Let us now check that $A_n \geq D'_n \geq A_n * \beta_n$:

- first $D'_n \leq D_n \leq A_n$;
- second, for all $u > s$, we have $D_n(u) \geq (A_n * \beta_n)(u)$ and $A_n(v) + \beta_n(s-v) \geq (A_n * \beta_n)(u)$, so $D'_n(u) \geq A_n * \beta_n(u)$. For all $u \leq s$, $D'_n(u) = D_n(u) \geq A_n * \beta_n(u)$. Then $D'_n \geq A_n * \beta_n$.

Let us now check that $D'_n = D'_n * \sigma_n$, or equivalently that D_n is σ_n-constrained. For all $t \geq s$, if $D'_n(t) = D_n(s)$, then for all $u \leq t$, $D'_n(t) - D'_n(u) \leq D_n(s) - \min(D_n(s), D_n(u)) \leq \sigma_n(s-u) \leq \sigma_n(t-s)$. We now only consider cases where $D'_n(t)$ is either $D_n(t)$ or $A_n(v) + \beta_n(t-v)$.

For all $u \geq 0$, either $D'_n(u) = D_n(u)$ and for all $t \geq u$, $D'_n(t) - D'_n(u) \leq D_n(t) - D_n(u) \leq \sigma_n(t-u)$, or $D'_n(u) \geq A_n(v) + \beta_n(u-v)$ and for all $t \geq u$, $D'_n(t) - D'_n(u) \leq \beta_n(t-v) - \beta_n(u-v) \leq \sigma_n(t-u)$ (as $\sigma_n \geq \beta_n$, the rate of σ_n is larger than any chord of the convex function β_n).

Finally, (A_n, D'_n) is an admissible pair of arrival and departure processes for server n satisfying the arrival and departure admissible relation and continuous at s.

C Proof of Theorem 7

In all the paragraph, we assume that $\mathcal{L}$ satisfies properties (P_1) and (P_2).

Preliminary results

Lemma 5 (Shape of $\mathcal{L}$). *For all $p \in \mathbb{N}_N$, $\mathcal{L}_p$ is concave and non-decreasing.*

Proof. To prove the result, let us rewrite function $\mathcal{L}_p(x)$ using the duality of the linear program. First, we separate variables x and y. From Property (P_1), one can transform $B_p(x,y)^t \leq C_p$ into $B'_p y^t \leq C'_p(x)$, where now the coefficients of C'_p depend on x. Property (P_1) tells us that the coefficients of $C'_p(x)$ are linear and non-decreasing in each variable of x. From Property (P_2), one can rewrite $A_p(x,y)^t$ as $A'_p y^t + A''_p x^t$, where all the coefficients of A''_p are non-negative.

We then have $\mathcal{L}_p(x) = \max\{A'_p y^t \mid B'_p y^t \leq C'_p(x),\ y \geq 0\} + A''_p x^t$. As $\mathcal{L}_p(x) \in \mathbb{R}_+$, the linear problem involved in $\mathcal{L}_p(x)$ has a finite optimal solution. The dual problem then has the same optimal solution, and we can express $\mathcal{L}_p(x)$ as

$$\mathcal{L}_p(x) = \min\{C'_p(x)^t w^t \mid B'^t_p w^t \geq A'^t_p,\ w \geq 0\} + A''_p x^t.$$

The polyhedron defined by $\{B'^t_p w^t \geq A'^t_p,\ w \geq 0\}$ does not depend on x. The optimal solution is obtained at a vertex of this polyhedron, that has a finite number of vertices. Then $\mathcal{L}_p(x)$ is the minimum of non-decreasing linear functions (the coefficients of $C'_p(x)$ are non-decreasing in x). Then $\mathcal{L}_p(x)$ is non-decreasing and concave. □ □

Let us denote $\mathcal{C} = \{x \in \mathbb{R}^N_+ \mid x \leq \mathcal{L}(x)\}$.

Lemma 6 (Existence of a fixed point x^*). *The supremum of $\mathcal{C}$ is well-defined in $(\mathbb{R}_+ \cup \{\infty\})^N$, and if $x^* = \sup(\mathcal{C}) \in \mathbb{R}^N_+$, x^* is a fixed-point of $\mathcal{L}$: $x^* = \mathcal{L}(x^*)$.*

Proof. To prove that $\sup(\mathcal{C})$ is well-defined, it suffices to show that $x, x' \in \mathcal{C}$, then $\bar{x} = x \vee x' \in \mathcal{C}$, where the maximum $\vee$ is taken coordinate-wise. As $\mathcal{L}$ is non-decreasing, for all $p \in \mathbb{N}_N$, $x_p \leq \mathcal{L}_p(x) \leq \mathcal{L}_p(x \vee x') = \mathcal{L}_p(\bar{x})$. Similarly, $x'_p \leq \mathcal{L}_p(x') \leq \mathcal{L}_p(x \vee x') = \mathcal{L}_p(\bar{x})$, so $\bar{x}_p = x_p \vee x'_p \leq \mathcal{L}_p(x \vee x')$ and $\bar{x} \leq \mathcal{L}(\bar{x})$. In other words, $\bar{x} \in \mathcal{C}$.

Let us prove that if $x^* \in \mathbb{R}^N_+$, then x^* is a fixed-point of $\mathcal{L}$. First, $x^* \in \mathcal{C}$. Indeed, for all $x \in \mathcal{C}$, $x \leq x^*$, so $x \leq \mathcal{L}(x) \leq \mathcal{L}(x^*)$, and $x^* = \sup\{x \mid x \in \mathcal{C}\} \leq \sup\{\mathcal{L}(x) \mid x \in \mathcal{C}\} \leq \sup\{\mathcal{L}(x^*) \mid x \in \mathcal{C}\} = \mathcal{L}(x^*)$. On the other hand, as $\mathcal{L}$ is non-decreasing, the inequality $\mathcal{L}(x^*) \leq \mathcal{L}(\mathcal{L}(x^*))$ also holds, so $\mathcal{L}(x^*) \in \mathcal{C}$ and $\mathcal{L}(x^*) \leq x^*$. Finally we have proved that $x^* = \mathcal{L}(x^*)$ and x^* is a fixed-point of $\mathcal{L}$. □ □

Uniqueness of the fixed-point Let us now prove that if x^* is finite, $\mathcal{L}$ has a unique fixed-point. To prove this, we will follow the lines of the proof of Kennan [52], where the result is proven for strictly concave functions and strict quasi-increasing functions. These assumptions do not hold as functions $\mathcal{L}_p$ are piecewise linear. We then adapt the proof in the case where $\mathcal{L}_p(0) > 0$ for concave and quasi-increasing functions. The adaptation in straightforward, but for sake of completeness, let use write it.

Definition 2. *A function $g = (g_1, \ldots, g_N) : \mathbb{R}^N_+ \to \mathbb{R}^N_+$ is quasi-increasing if for all $p \in \mathbb{N}_N$, for all x, y, such that $y \geq x$ and $x_p = y_p$, $g_p(y) \geq g_p(x)$.*

In our case, as $\mathcal{L}$ in non-decreasing, $\mathcal{L} - Id$ is quasi-increasing, where Id is the identity function.

Lemma 7. *Let F be a function from $\mathbb{R}^N_+$ to $\mathbb{R}^N_+$, that is concave and such that $F - Id$ is quasi-increasing and $F_p(0) > 0$ for all $p \in \mathbb{N}_N$. If F has a fixed point, then this fixed point is unique.*

Proof. Suppose x and y are two fixed points of F: $F(x) = x$ and $F(y) = y$. Define $\gamma = \min_{q \leq N}(\frac{x_q}{y_q}) = \frac{x_r}{y_r}$. If $\gamma \geq 1$, then $x \geq y$. Suppose now that $\gamma < 1$ and define $w = \gamma y$. On the one hand, $w_p < y_p$ for all $p \leq N$. So by concavity of F_p, we have $F_p(w) \geq (1-\gamma)F_p(0)+\gamma F_p(y) > \gamma F_p(y) = \gamma y_p = w_p$. In particular, $F_r(w) > w_r$.

On the other hand, $w \leq x$ and $w_r = x_r$. Indeed, $w_q = \gamma y_q \leq \frac{x_q}{y_q} y_q = x_q$, and the inequality becomes an equality when $p = r$. As $F - Id$ is quasi-increasing, then $F_r(x) - x_r \geq F_r(w) - w_r$.

But x is a fixed point, so combining the obtained inequalities, we get $0 = F_r(x) - x_r \geq F_r(w) - w_r > 0$, which is a contradiction. So one must have $x \geq y$.

By inverting the roles of x and y, we also obtain $x \leq y$, and finally $x = y$. This concludes the proof regarding the uniqueness of the fixed point. □ □

As $\mathcal{L}$ satisfies all the hypotheses of Lemma 7, *i.e.*, $\mathcal{L}$ is concave, $\mathcal{L}(0) > 0$ and $\mathcal{L} - Id$ is quasi-increasing, if $\mathcal{L}$ has a finite fixed point, then it is unique.

Computation of the fixed-point The second part of the theorem consists in finding a linear program whose optimal solution is that fixed point. Let us consider the linear program

$$\max\{\sum_{p=1}^{N} x_p \mid \forall p \in \mathbb{N}_N, x_p \leq A_p(x, y_p)^t, B_z(x, y_p)^t \leq C_p, x, y_p \geq 0\}. \quad (4)$$

The set of solutions of this linear program is

$$\mathcal{C}' = \{x \in \mathbb{R}_+^N \mid \forall p \in \mathbb{N}_N, \exists y_p \geq 0,\ x_p \leq A_p(x, y_p)^t, B_z(x, y_p)^t \leq C_p\}.$$

Let us first show that the set of solutions is exactly the set $\mathcal{C}$.

Lemma 8. $\mathcal{C} = \mathcal{C}'$

Proof.

$$\begin{aligned} x \in \mathcal{C} &\Leftrightarrow \forall p \in \mathbb{N}_N,\ x_p \leq \mathcal{L}_p(x) \\ &\Leftrightarrow \forall p \in \mathbb{N}_N,\ \exists y_p \geq 0,\ x_p \leq A_p(x, y_p)^t \text{ and } B_p(x, y_p)^t \leq C_p \\ &\Leftrightarrow x \in \mathcal{C}'. \end{aligned}$$

□ □

An optimal solution of Equation (4) is a solution that maximizes the sum of its coefficients.

Lemma 9. *The optimal solution of Equation*(4) *is unique and is* x^*.

Proof. As $x^* = \sup(\mathcal{C})$ is well-defined and $\mathcal{C} = \mathcal{C}'$, $x^* = \sup(\mathcal{C}')$. As a consequence, for each solution x of Equation(4), $x \leq x^*$. Then, $\sum_{p=1}^{N} x_p \leq \sum_{p=1}^{N} x_p^*$. The equality holds if and only if $x = x^*$. □ □

D Proof of Lemma 4

We will use the following classical lemma.

Lemma 10. *For all* $n \in \mathbb{N}$, $X, Y \in \mathbb{R}_+$ *and* $y \in \mathbb{R}_+^n$ *such that* $\sum_{i=1}^{n} y_i \geq Y$, *if* $X \leq Y$ *then there exists* $x \in \mathbb{R}_+^n$ *such that* $x_i \leq y_i$ *and* $\sum_{i=1}^{n} x_i = X$.

Proof. We define recursively for all $i \in \mathbb{N}_n$, $x_i = \min(y_i, X - \sum_{j=1}^{i-1} x_j)$. By construction $x_i \leq y_i$, and it remains to show that $\sum_{i=1}^{n} x_i = X$. We proceed by induction and show that the partial sums $\sum_{j=1}^{i} x_j = \min(\sum_{j=1}^{i} y_j, X)$.

- Initialization: for $i = 1$, $x_1 = \min(y_1, X)$, by definition;
- Induction: Assume that $\sum_{j=1}^{i} x_j = \min(\sum_{j=1}^{i} y_j, X)$. Then

$$\begin{aligned}
\sum_{j=1}^{i+1} x_j &= \sum_{j=1}^{i} x_j + \min(y_{i+1}, X - \sum_{j=1}^{i} x_j) \\
&= \min(\sum_{j=1}^{i} x_j + y_{i+1}, X) \\
&= \min(\min(\sum_{j=1}^{i} y_j, X) + y_{i+1}, X) \\
&= \min(\sum_{j=1}^{i+1} y_j, X + y_{i+1}, X) \\
&= \min(\sum_{j=1}^{i+1} y_j, X).
\end{aligned}$$

One can conclude, as $\sum_{i=1}^{n} x_i = \min(\sum_{i=1}^{n} y_i, X) = X$ as $X \leq Y \leq \sum_{i=1}^{n} y_i$. □

□

Proof of lemma 4. We already showed how to get rid of the variables related to $\mathbf{u}$. Set $\mathbf{As} = \sum_{i=1}^{m} \mathbf{A}_i\mathbf{s}$. For all $\mathbf{s}$, we have the equivalence between

$$\exists \mathbf{A}_i\mathbf{s},\ \forall i \in \mathbb{N}_m,\ \mathbf{A}_i\mathbf{s} \leq x_i + r_i\mathbf{s} \text{ and } \forall h \in \mathbb{N}_H,\ \sum_{i \in G_h} \mathbf{A}_i\mathbf{s} \leq L_h + C_h\mathbf{s}$$

and

$$\exists \mathbf{As},\ \mathbf{As} \leq \sum_{h=1}^{H} (\sum_{i \in G_h} x_i + r_i\mathbf{s} \wedge L_h + C_h\mathbf{s}) + \sum_{i \notin \bigcup_{h \in \mathbb{N}_h} G_h} x_i + r_i\mathbf{s}.$$

The first direction of the equivalence is obtained by setting $\mathbf{As} = \sum_{i=1}^{n} \mathbf{A}_i\mathbf{s}$. To simplify, set $G_{H+1} = \mathbb{N}_m \setminus \cup_{h \in \mathbb{N}_H} G_h$ and $L_{H+1} = C_{H+1} = \infty$. The second direction us obtained by using twice Lemma 10:

- first to the group to find variables $\mathbf{A}^{(h)}\mathbf{s}$ such that $\sum_{h=1}^{H+1} \mathbf{A}^{(h)}\mathbf{s} = \mathbf{As}$, with $Y = \sum_{h=1}^{H+1} (\sum_{i \in G_h} x_i + r_i\mathbf{s} \wedge L_h + C_h\mathbf{s})$ and $y_h = \sum_{i \in G_h} x_i + r_i\mathbf{s} \wedge L_h + C_h\mathbf{s}$ for all $h \in \mathbb{N}_{H+1}$;
- second to each group: find $\mathbf{A}_i\mathbf{s}$ such that $\sum_{i \in G_h} \mathbf{A}_i\mathbf{s} = \mathbf{A}^{(h)}$ and $Y = \sum_{i \in G_h} x_i + r_i\mathbf{s} \wedge L_h + C_h\mathbf{s}$, and $y_i = x_i + r_i\mathbf{s}$.

As $\mathbf{A}_i\mathbf{s} = \mathbf{D}_i\mathbf{t}$, for all $i \in \mathbb{N}_m$, one can get rid of the FIFO constraints and replace $\mathbf{D}_i\mathbf{t}$ by $\mathbf{A}_i\mathbf{s}$ in the service constraints.

$(\mathbf{s}, \mathbf{t}, (\mathbf{A}_i\mathbf{s})_{i=1}^{m}, (\mathbf{D}_i\mathbf{t})_{i=1}^{m})$ is a satisfies the linear constraints if and only is there exists $\mathbf{s}, \mathbf{t}$ and $\mathbf{As}$ such that $\mathbf{s} \leq \mathbf{t}$ and

$$(\beta_{R,T}(\mathbf{t}) =) \max(0, R(\mathbf{t} - T)) \leq \mathbf{As} \leq \sum_{h=1}^{H+1} (\sum_{i \in G_h} x_i + r_i\mathbf{s} \wedge L_h + C_h\mathbf{s}) (= \alpha(\mathbf{s})).$$

It is now possible to remove the variable $\mathbf{As}$, and replace $\mathbf{s}$ by $\mathbf{t}-\mathbf{d}$, and introduce the functions α and $\beta_{R,T}$ and we obtain the equivalent formulation: maximizing $\mathbf{d}$ such that there exists $\mathbf{t} \geq 0$ such that $\mathbf{t} \geq \mathbf{d}$ and

$$\beta_{R,T}(\mathbf{t}) \leq \alpha(\mathbf{t}-\mathbf{d}),$$

which is exactly the formulation of the maximal horizontal distance between α and $\beta_{R,T}$.

Remark that the delay and horizontal distance at $\mathbf{t}$ (page 5) is expressed with a strict inequality ($d_{max} = \sup_t(\sup\{d \geq 0 \mid \beta_{R,T}(\mathbf{t}) < \alpha(\mathbf{t}-\mathbf{d})\})$). As α is strictly increasing, we also have $d_{\max} = \sup_t(\sup\{d \geq 0 \mid \beta_{R,T}(\mathbf{t}) \leq \alpha(\mathbf{t}-\mathbf{d})\})$. □ □