Half-modeling of shaping in FIFO net with network calculus

Marc Boyer

Half-modeling of shaping in FIFO net with network calculus

Marc Boyer
ONERA – Toulouse, France

E-mail: Marc.Boyer@onera.fr

Abstract

Shaping is a well known solution to increase performances in networks, and especially worst-case performances. In the avionic context, considering the AFDX embedded backbone, it had been shown that modelling the shaping introduced by the links leads to significant gain. This shaping have been modelled to compute local delay with network calculus in previous studies. In this paper, we try to model it for an end-to-end delay where each server is shared by two flows with a FIFO policy. Due to technical difficulties, only the shaping of one of the two flows is modelled (giving the title "half-modeling of shaping").

1 Introduction

Shaping is a well known solution to increase performances in networks, and especially worst-case performances. In the avionic context, considering the AFDX embedded backbone, it had been shown in [1, Table 2] that modelling the shaping introduced by the links[1] on realistic configuration can lead to gain from 10% to 50%, depending on the formal method used, the data flow considered, etc.

Among other methods, network calculus [12, 13, 16, 10] can be used to compute bounds on worst-case traversal time of network (WCTT). Network calculus has a solid mathematical background (the (min,plus) algebra), is scalable (useful classes of modelling have linear algorithms [9]), and have been used to certify the A380 AFDX backbone [15, 14].

This paper aims to model in network calculus two characteristic of AFDX networks: FIFO policy and shaping introduced by links. Both have been studied independently, and the aim is to combine both. In this first work, we consider a simple topology, a sequence of servers, each one shared by two flows with a FIFO policy (cf Figure 1). Moreover, the modelling is incomplete: for technical reasons, when computing the WCTT of a flow, we model only the shaping on the considering flow R, not on the interfering ones R'_i (leading to the title "half-modeling of shaping"). The main results of the paper consists in solving a well-known equation related to FIFO scheduling (presented in eq (4)) on this specific topology and one shaped flow (Theorem 1).

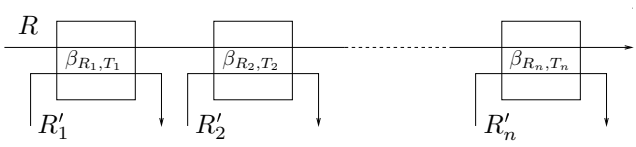

Figure 1. Tandem topology

As presented on small examples, depending on the bursts sizes, this could lead to significant gains (cf Table 1). But on realistic industrial example, the method fully modelling the shaping and neglecting the FIFO is still the better one. This encourages us to continue our works and fully combine shaping and end-to-end FIFO impact.

After a short introduction on network calculus in Section 2, related works will be presented in Section 3, which will also include a detailed presentation on the model considered in this paper (Section 3.3). The main contribution of the paper is in Section 4. Section 5 gives some numerical results on an simple example, to illustrate the gains of the new method. Section 6 concludes.

2 Network calculus

Notations $\mathbb{R}$ denotes the set of real numbers. The minimum (resp. maximum) operator is denoted $\wedge$ (resp. $\vee$). $[x]^+ \stackrel{\text{def}}{=} x \vee 0$

Here is a (very short) introduction to network calculus. The reader should refer to [12, 13] for the first works and [10, 16] for a complete presentation.

[1] Depending on authors, this impact of the links is called *grouping*, to illustrate the fact that the flows sharing a single link can be seen as a "group", or *serialisation*, to highlight the fact that the frames on the link are transmitted in sequence. From network calculus point of view, this effect is taken into account by modelling the link as a shaper.

Network calculus is a theory to get deterministic upper bounds in networks. It is mathematically based on the $(\wedge, +)$ dioid.

Network calculus mainly handles non decreasing functions, null before 0 : $\mathcal{F}$. They are, among others, four common curves (parametrised by real positive values d, R, T, r, b) latency δ_d, rate λ_R, rate-latency $\beta_{R,T}$, token bucket $\gamma_{r,b}$, test $\mathbb{1}_{\{>T\}}$ defined by:

$$\delta_d(t) = \begin{cases} 0 \text{ if } t \leqslant d \\ \infty \text{ otherwise} \end{cases} \qquad \mathbb{1}_{\{>T\}}(t) = \begin{cases} 1 \text{ if } t > 1 \\ 0 \text{ otherwise} \end{cases}$$

$$\gamma_{r,b}(t) = (rt + b)\mathbb{1}_{\{>0\}}(t) \qquad \lambda_R(t) = Rt$$

$$\beta_{R,T} = R[t - T]^+$$

Three basic operators on $\mathcal{F}$ are of interest, convolution $*$, deconvolution $\oslash$ and the sub-additive closure f^*.

$$\mathcal{F} = \left\{ f : \mathbb{R} \to \mathbb{R} \;\middle|\; \begin{array}{lll} x < y & \Longrightarrow & f(x) \leqslant f(y) \\ x < 0 & \Longrightarrow & f(x) = 0 \end{array} \right\}$$

$$(f * g)(t) = \inf_{0 \leqslant u \leqslant t} (f(t-u) + g(u)) \tag{1}$$

$$(f \oslash g)(t) = \sup_{0 \leqslant u} (f(t+u) - g(u)) \tag{2}$$

$$f^* = \delta_0 \wedge f \wedge (f * f) \wedge (f * f * f) \wedge \cdots \tag{3}$$

They are several well-known properties of theses operators and the common curves. The convolution is associative and commutative, $\delta_t * \delta_{t'} = \delta_{t+t'}$, $f * \delta_d(t) = f(t-d)$ if $t \geqslant d$, 0 otherwise, $f \oslash \delta_d(t) = f(t+d)$, $\beta_{R,T} = \delta_T * \lambda_R$.

A flow is represented by its cumulative function $R \in F$, where $R(t)$ is the total number of bits sent by this flow up to time t. A flow R is said to have a function α as *arrival curve* iff $\forall t, s \geqslant 0 : R(t+s) - R(t) \leqslant \alpha(s)$. It means that, from any instant t, the flow R will produce at most $\alpha(s)$ new data in s time units. An equivalent condition, expressed in the $(\wedge, +)$ dioid is $R \leqslant R * \alpha$. If α is an arrival curve for R, also is α^*. A server offers a *service* of curve β iff for all arrival flow, the relation between the input flow R and the output flow R', we have $R' \geqslant R * \beta$. In this case, $\alpha' = \alpha \oslash \beta$ is an arrival curve for R'. The delay experimented by the flow R can be bounded by the maximal horizontal difference between curves α and β, formally defined by $h(\alpha, \beta)$ (a graphical interpretation of h can be found in Figure 2).

$$h(\alpha, \beta) = \sup_{s \geqslant 0} \left(\inf \left\{ \tau \geqslant 0 \,\middle|\, \alpha(s) \leqslant \beta(s + \tau) \right\} \right)$$

These first results allow to handle linear topologies, like the one of Figure 3. Given the arrival curve α of flow R, and the services curves β, β' of network elements S, S', we are able to get a bound[2] on the delay in S: $h(\alpha, \beta)$, and another for the delay in S': $h((\alpha \oslash \beta)^*, \beta')$.

[2]To effectively compute these bounds, we need algorithms computing operations like $\oslash$, $\cdot^*$, $h(\cdot, \cdot)$. This is the aim of [3] and its COINC [11] implementation.

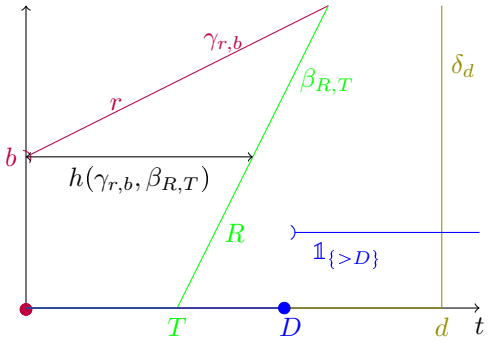

Figure 2. Common curves and delay

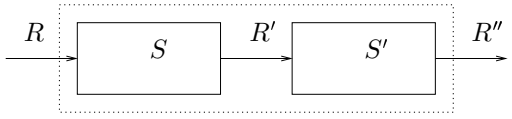

Figure 3. A flow going through two network elements in sequence

But one famous result of the network calculus is known as "pay burst only once" (PBOO). It states that a sequence of two servers, S and S', with respective service of curve β, β', like the one of Figure 3, can be seen as a single server with service of curve $\beta * \beta'$. It's interest comes from the fact that the end-to-end delay is lesser than the sum of local delays (*i.e.* $h(\alpha, \beta * \beta') \leqslant h(\alpha, \beta) + h((\alpha \oslash \beta)^*, \beta')$). And the difference can be, in practise, significant. It's popularity perhaps comes from the simplicity of the proof[3].

In case of more realistic topology, when a network element is shared by different flows (like in Figure 4), with some service policy (FIFO, static priority, etc.), it also exists results to compute bounds on each flow. The idea is to extract, from an *aggregated* flow, the *residual* service offered to each flow (also known as *per-flow* service). This extraction is, at first step, local, *i.e.* on a single server. Depending on the policy, the bounds could be tight (*i.e.* it exists some configuration where the computed bound is reached), or not, or the result being up to now unknown. The second step consists in considering a complete topology, trying to get an end-to-end bound better than the sum of local delays (like the PBOO result). As for local step, depending on the policy *and* the topology, the bounds could be tight or not.

For example, with the non preemptive static priority policy, it is known[4] that the lower priority flow has the residual

[3]Server S' offers a service of curve β means that $R'' \geqslant R' * \beta'$. The same for S: $R' \geqslant R * \beta'$. It implies $R'' \geqslant (R * \beta) * \beta'$ *i.e.* by associativity, $R'' \geqslant R * (\beta * \beta')$, which is the definition of a server with service $\beta * \beta'$.

[4]To be precise, there is a restriction on the flavor of service, which must be *strict*. See [16, Def 1.3.2, Cor. 6.2.1,] for details.

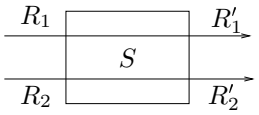

Figure 4. A single shared network element

service $\beta_L = [\beta - \alpha_H]^+$, if the network element has service curve β and the high priority flow has arrival curve α_H.

The common result for FIFO policy (the one focused in this paper) claims that, if a FIFO server (with a service of curve β) is shared by two flows R_1, R_2 (with respective arrival curves α_1, α_2), then, for each $\theta \geqslant 0$, each flow R_i receives the residual service β_i^θ defined by

$$\beta_i^\theta = [\beta - \alpha_j \oslash \delta_\theta]^+ \mathbb{1}_{\{>\theta\}} \tag{4}$$

with $\{i, j\} = \{1, 2\}$ (see [16, Prop 6.4.1] for details). An equivalent definition is $\beta_i^\theta(t) = [\beta(t) - \alpha_j(t-\theta)]^+ \mathbb{1}_{\{>\theta\}}$.

Notice that this result does not define a *single* residual service, but an *infinite set* of services, one for each value of θ. They all compute true bounds, but some are better than others. The issue is the choice of a good θ value.

Another result must be presented, so simple that nobody did ever present it as a result. It is used in [15], and claims that, if d denotes the delay experimented by the aggregated flow, then a FIFO server can be under-approximated by a variable delay d, and β_d is a residual service curve for each flow R_i.

$$\beta_d = \delta_{h(\beta, \alpha_1 + \alpha_2)} \tag{5}$$

3 Related works and considered model

This paper is the merging and continuation of two works. On the one hand, the one of the Computer Networking Group of the Pisa University on FIFO bounds with network calculus [18, 17, 19]. On the other hand, the one of the IRT team from Toulouse University on AFDX networks [15, **?**, 14].

3.1 LUB from Computer Networking Group, Pisa University

As presented in section 2, to handle real case studies, with servers shared by different flows, with network calculus, the first step is to handle locally a service policy, and the second is the handling of a complete topology. From this point of view, the work presented in [18, 17, 19], called LUB (least upper bound), is the generalisation of equation (4), trying to compute an end-to-end delay better than the sum of local delays. Considering more and more complex topologies, they consider the possible ways to chose

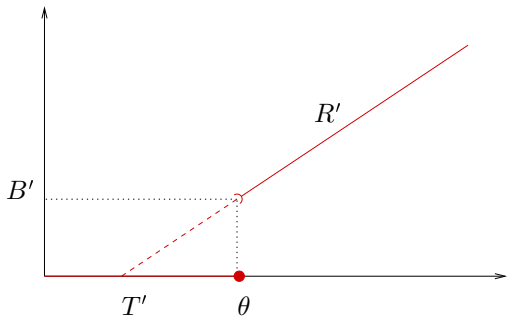

Figure 5. Local residual service

the θ values on each server, the ways to combine local computation, to get better bounds. Some common restriction are done on the class of functions used: arrival curves are token bucket ($\gamma_{r,b}$) and service curves are rate-latency ($\beta_{R,T}$).

In [18], they are considering what they call *tandem* topologies, *i.e.* a sequence of servers, with the flow of interest traversing all servers, and sharing each server with an interfering flow, like in Figure 1. This is also the kind of topology considered in this paper. The idea is, at each node S_i, to compute the residual service with (4), parametrised by a value θ_i, to get a parametrised expression of the end-to-end delay bound (to benefit of the PBOO result), and to minimise this expression. In [17], this approach is generalised by considering sink-tree networks, and in [19] generic topologies (without cyclic dependencies in data flows).

Let's go to mathematical details of [18].

The service at each node S_i has a curve β_{R_i,T_i}, and each interfering flow has arrival curve $\gamma_{r'_i,b'_i}$. For each $\theta_i \geqslant 0$, the residual service can be computed using (4).

$$\beta_i^{\theta_i} = [\beta_{R_i,T_i} - \gamma_{r'_i,b'_i} \oslash \delta_\theta]^+ \mathbb{1}_{\{>\theta_i\}} = \beta_{R'_i,T'_i} \mathbb{1}_{\{>\theta_i\}} \tag{6}$$

with $R'_i = R_i - r'_i$ and $T'_i = \frac{R_i T_i + b_i - r_i \theta_i}{R_i - r'_i}$. But this expression can be seen as the convolution of a delay and a token bucket

$$\beta_i^{\theta_i} = \beta_{R'_i,T'_i} \mathbb{1}_{\{>\theta_i\}} = \delta_{\theta_i} * \gamma_{R'_i,B'_i} \tag{7}$$

with $B'_i = R_i(\theta_i + T_i) + b'_i$, as illustrated in Figure 5.

The end-to-end residual service is the convolution of the local residual services $\beta_{R'_1,T'_1} \mathbb{1}_{\{>\theta_1\}} * \beta_{R'_2,T'_2} \mathbb{1}_{\{>\theta_2\}} * \ldots * \beta_{R'_n,T'_n} \mathbb{1}_{\{>\theta_n\}} = \delta_{\sum_i \theta_i} * \left(\gamma_{R'_1,B'_1} \wedge \ldots \wedge \gamma_{R'_n,B'_n}\right)$. It can be seen as the convolution of a delay and the minimum of affine functions (cf. Figure 6). Such a curve will be called "pseudoaffine curve" in [17]. Computing the end-to-end delay bound consists in computing the horizontal deviation $h(\gamma_{r,b}, \delta_{\sum_i \theta_i} * \left(\gamma_{R'_1,B'_1} \wedge \ldots \wedge \gamma_{R'_n,B'_n}\right))$.

Last step consists in minimising the previous expression, which implies to compute the following infimum, with $b_i \geqslant$

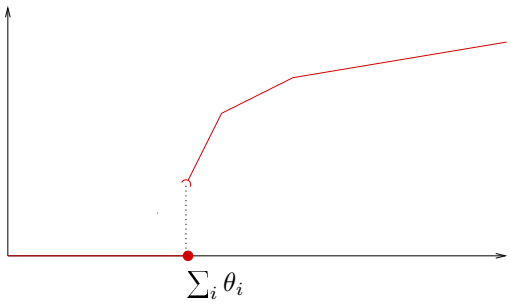

Figure 6. End-to-end residual service

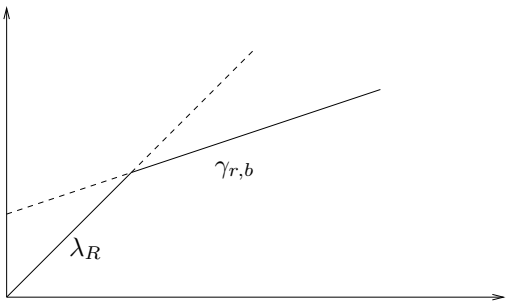

Figure 7. Shaped token bucket

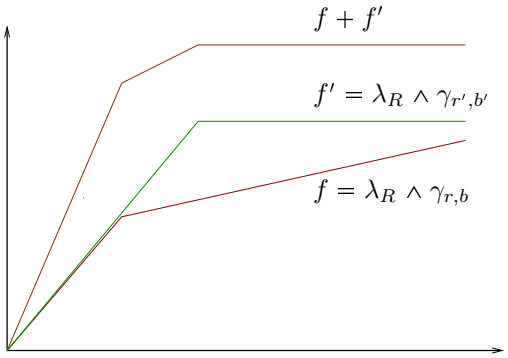

Figure 8. Sum of shaped token bucket gives general CPL curve

0 and $a_i \geqslant 1$.

$$\inf_{x_i \geqslant 0} \left\{ \sum_i x_i \wedge \bigwedge_i [b - a_i x_i]^+ \right\} \tag{8}$$

3.2 IRT, Toulouse University

In [15], the approach is different: only local delays are computed, there is no attempt of modelling the PBOO principle. The global delay of each server is computed, *i.e.* all flows incoming into a server are aggregated (the aggregated arrival curve is simply the sum of individual arrival curve), and the delay experimented by this aggregated flow is computed. Then, the delay computed for a flow is simply the sum of the aggregated delay of each server crossed by the flow.

The contribution of [15], in addition to modelling the AFDX network in network calculus, is to model the shaping introduced by the links (called "grouping" in [15, 14, 1]). On a sender, whatever the applicative burst is, the throughput outgoing of the server is limited by the throughput of the link (100Mb/s in common AFDX). Another point of view, looking at trajectories of frames, is that links *serialise* packets. In network calculus modelling, it means that an applicative token bucket $\gamma_{r,b}$ is shaped by the bit-rate of the link λ_R, leading to a two-slopes affine arrival curve, as presented in Figure 7. But in switches, we have to consider the sum of flow incoming from different links, which is no more two-slope affine curve, as presented in Figure 8. It leads to general concave piecewise linear function (CPL).

Let's go to mathematical details.

Each incoming flow f_i of a server S has arrival curve α_i, in the CPL class. The server as a service of curve β_S. The aggregated arrival curve $\alpha_S = \sum_i \alpha_i$ is still a CPL. The delay bound computed is $d_S = h(\beta_S, \sum_i \alpha_i)$. Because of FIFO policy, δ_{d_S} is a service curve for each flow (cf eq (5))), and each f'_i, output corresponding to the f_i input has $(\alpha_i \oslash \delta_{d_S})^*$ as an arrival curve, which is still a CPL).

The delay experimented by a flow is then the sum of the d_S delays of all the S crossed servers.

This approach does not consider the PBOO principle, but models correctly the shaping introduced by the links. This shaping can increase the performances of the analysis up to 40%. And even when considering another theory, based on trajectories, modelling of shaping leads to 5%–50% gain [1].

3.3 System model

This work is a direct continuation and join of the works presented in previous sections 3.1 and 3.2. We are considering exactly the same kind of topology than the one of [18], illustrated in Figure 1, but with a more general class of arrival curve: the considered flow has a CPL arrival curve, but the interfering flows still have token-bucket $\gamma_{r'_i,b'_i}$ arrival curves.

4 Computing the minimal bound

To begin, let us introduce some notations on the kind of manipulated curves. A concave piecewise linear (CPL) function is said to be under normal form if the terms are sorted by decreasing slopes, and there is no useless term (cf the *non-redondant* property of pseudoaffine curves in [19]).

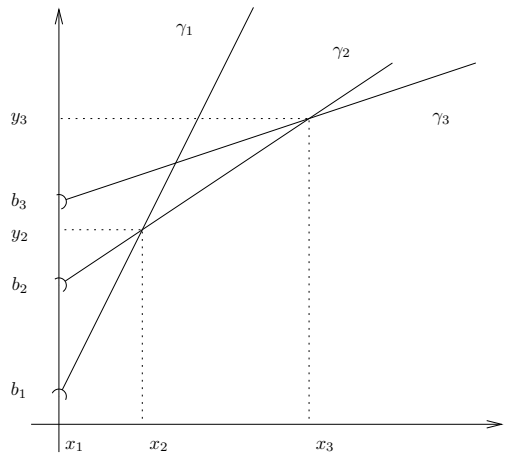

Figure 9. Normal form of CPL

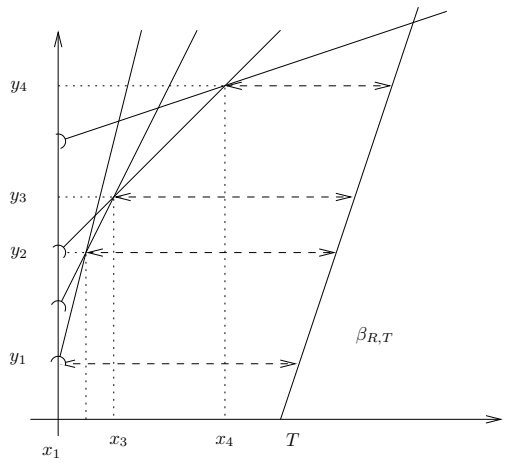

Figure 10. Horizontal deviation between CPL and rate-latency

Definition 1 (Normal form of CPL) Let $\gamma_{r_1,b_1}, \ldots, \gamma_{r_n,b_n}$ be a set of token-bucket functions. Let γ_i denotes γ_{r_i,b_i}. The term $\bigwedge_{i=1}^{n} \gamma_i$ is said to be *under normal form of minimum of γ functions*, iff there is no useless constraint (9) and the γ_i are sorted by decreasing rate (10).

$$\forall i, \exists t_i > 0, \forall j \neq i : \gamma_i(t_i) < \gamma_j(t_i) \tag{9}$$

$$i < j \Rightarrow r_i > r_j \tag{10}$$

If $\bigwedge_{i\in[1,n]} \gamma_{r_i,b_i}$ is under normal form, the sequence $x_1, \cdots, x_{n+1}$ of intersection points, and $y_1, \cdots, y_{n+1}$ the intersection values are formally defined by:

$$\begin{cases} x_1 = 0, y_1 = b_1 \\ \gamma_i(x_i) = \gamma_{i+1}(x_i) = y_k \quad \text{for } 1 \leqslant i \leqslant n \\ x_{n+1} = y_{n+1} = \infty \end{cases} \tag{11}$$

□

It should be obvious that each CPL has a normal form. To effectively compute it, an algorithm is given in [8].

An example of such a set and related definitions is shown in Figure 9.

Lemma 1 (Horizontal dev. between CPL and rate-latency) *Let $\bigwedge_i \gamma_{r_i,b_i}$ be a CPL under normal form, and R, T, two non negative reals, with $r_n \leqslant R$. Then, the horizontal deviation can be computed*

$$h(\bigwedge_i \gamma_{r_i,b_i}, \beta_{R,T}) = T + \frac{y_k}{R} - x_k \tag{12}$$

with $k = \min\{i \,|\, r_i \leqslant R\}$. □

PROOF Graphically (see Figure 10), the maximal distance must be computed at one y_i level. Going from the origin, while the rates r_i are greater than R, the horizontal deviation grows, up to x_k, which is, by definition, the first slope lesser than R. ■

To make the different proofs in this paper, we manipulate a lot of $\beta_{R,T}\mathbb{1}_{\{>T'\}}$ related expressions (going from eq. (4), cf Figure 6). Here are expressed different points of view on such expressions. They are of interest because of the different properties on the involved operators.

Lemma 2 (Some point of view on $\beta_{R,T}\mathbb{1}_{\{>T'\}}$ expressions) *Let R, T, D, r, b be some positive reals and $f \in \mathcal{F}$ an increasing function. Then*

$$f\mathbb{1}_{\{>T'\}} = \delta_{T'} \wedge f \tag{13}$$

$$\beta_{R,T}\mathbb{1}_{\{>D\}} = \begin{cases} \beta_{R,T} & \text{if } T \geqslant D \\ \delta_D * \gamma_{R,R(D-T)} & \text{if } T < D \end{cases} \tag{14}$$

$$= \delta_{D\vee T} * \gamma_{R,R[D-T]^+} \tag{15}$$

$$\delta_D * \gamma_{r,b} = \begin{cases} \delta_D \wedge \gamma_{r,b-rD} & \text{if } b \geqslant rD \\ \delta_D \wedge \beta_{r,D-\frac{b}{r}} & \text{if } b \leqslant rD \end{cases} \tag{16}$$

□

Interest of eq (13) comes from the fact that, in $(\wedge, +)$ algebra, we have more results on minimum than on product. In particular, $\beta_{R,T}\mathbb{1}_{\{>T'\}} = \delta_{T'} \wedge \beta_{R,T}$.

The idea behind expression (14) was somehow already present in [18, Eq. (12)], and generalised with the notion of *pseudoaffine* curve in [17, Eq. (6)]. Equation (15) is a new point of view that avoid some "per case" manipulations.

PROOF Expression (13) is obvious.

To prove the expression (14), just have a look on Figure 5, illustrated with $D = \theta > T$. If $D \leqslant T$: the $\mathbb{1}_{\{>D\}}$ term has no effect.

To go on from (14) up to (15), just remark that $\beta_{R,T}$ can

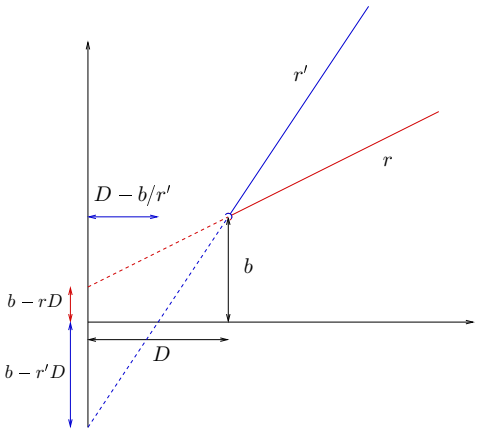

Figure 11. Shifted γ can be expressed as a minimum

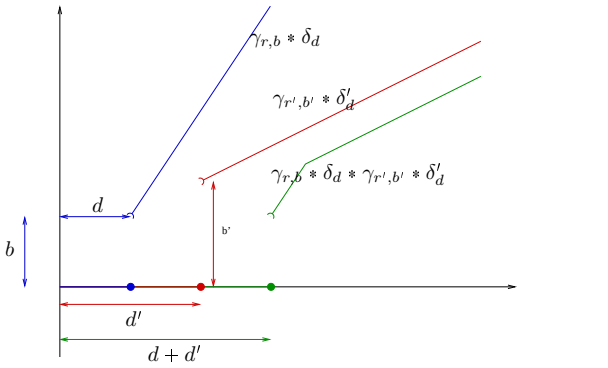

Figure 12. Convolution of two truncatd beta functions

also be written $\beta_{R,T} = \delta_T * \gamma_{R,0}$. Then, we have

$$\beta_{R,T}\mathbb{1}_{\{>D\}} = \begin{cases} \delta_T * \gamma_{R,0} & = \delta_{D\vee T} * \gamma_{R,R[D-T]^+} \\ & \text{if } T \geqslant D \\ \delta_D * \gamma_{R,R(D-T)} & = \delta_{D\vee T} * \gamma_{R,R[D-T]^+} \\ & \text{if } T < D \end{cases}$$

Looking at Figure 11 gives eq (16). Shifting a $\gamma_{r,b}$ function of D to the right is like, either truncating a γ function, or a β function, depending on the value $b - rD$. ∎

Lemma 3 (Convolution of truncated beta [18]) *Let R, T, D, R', T', D' be six positive reals with $T < D$ and $T' < D'$. Then*

$$\beta_{R,T}\mathbb{1}_{\{>D\}} * \beta_{R',T'}\mathbb{1}_{\{>D'\}} = \delta_{D+D'} * \gamma_{R,R(D-T)} * \gamma_{R',R'(D'-T')} \quad (17)$$

It can be generalised to any number of function and any values of T_i, D_i.

$$\overset{n}{\underset{i=1}{*}} \beta_{R_i,T_i}\mathbb{1}_{\{>D_i\}} = \delta_{\sum_{j=1}^n (D_j \vee T_j)} * \bigwedge_{i=1}^n \gamma_{R_i,R_i[D_i-T_i]^+} \quad (18)$$

$$= \delta_{\sum_{j=1}^n (D_j \vee T_j)} \wedge \bigwedge_{i=1}^n \beta_{R_i,(\sum_{j=1}^n D_j \vee T_j)-[D_i-T_i]^+} \quad (19)$$

□

Equation (18) comes from [18, Proof of Theorem 2], and expression (19) is another point of view on the same equation.

Both expressions (18) and (19) are of interest.

Expression (18) clearly expresses the fact that we have a concave function "shifted" by a delay $\sum_{i=1}^n D_i$. Expression (19) is interesting when trying to compute the delay, with help of Lemma 4 (which states that for wide-sense increasing functions $h(f, g \wedge h) = h(f, g) \vee h(f, h)$).

PROOF

- *Proof of* (17)*:* The proof of (17) is obvious, using equation (14) (transformation into convolution of delay and γ function) and basic properties of convolution ($\delta_d * \delta'_d = \delta_{d+d'}$; cf Figure 12).

$$\begin{aligned} & \beta_{R,T}\mathbb{1}_{\{>D\}} * \beta_{R',T'}\mathbb{1}_{\{>D'\}} \\ = \ & \delta_D * \gamma_{R,R(D-T)} * \delta'_D * \gamma_{R',R'(D'-T')} \\ = \ & \delta_{D+D'} * \gamma_{R,R(D-T)} * \gamma_{R',R'(D'-T')} \end{aligned}$$

- *Proof of* (18)*:* To prove (18), we can generalise (17) to the equation (15), and the fact that convolution and minimum are equals on star-shaped functions [16, Th 3.1.6].

$$\begin{aligned} \overset{n}{\underset{i=1}{*}} \beta_{R_i,T_i} 1_{\{t>D_i\}} &= \delta_{\sum_{i=1}^n (D_i \vee T_i)} * \overset{n}{\underset{i=1}{*}} \gamma_{R_i,R_i[D_i-T_i]^+} \\ &= \delta_{\sum_{i=1}^n (D_i \vee T_i)} * \bigwedge_{i=1}^n \gamma_{R_i,R_i[D_i-T_i]^+} \end{aligned}$$

- *Proof of* (19)*:* Going from (18) to (19) is done using distributivity of min over deconvolution, and (16).

  The first step uses the distributivity: $f * (g \wedge h) = (f * g) \wedge (f * h)$:

$$\delta_{\sum_{i=1}^n (D_i \vee T_i)} * \bigwedge_{i=1}^n \gamma_{R_i,R_i[D_i-T_i]^+} = \bigwedge_{i=1}^n \left(\delta_{\sum_{j=1}^n (D_j \vee T_j)} * \gamma_{R_i,R_i[D_i-T_i]^+} \right) \quad (20)$$

The second step uses (16). We have to compare $b = R_i[D_i - T_i]^+$ and $rD = R_i \sum_{i=j}^{n}(D_j \vee T_j)$.

$$\sum_{j=1}^{n}(D_j \vee T_j) \geqslant \sum_{j=1}^{n} D_j \geqslant D_i \geqslant [D_i - T_i]^+$$
$$\Rightarrow R_i \sum_{j=1}^{n}(D_j \vee T_j) \geqslant R_i[D_i - T_i]^+$$

We are in the case $rD \geqslant b$ that is to say $\delta_D * \gamma_{r,b} = \delta_D \wedge \beta_{r,D-\frac{b}{r}}$. That is to say:

$$\begin{aligned}&\delta_{\sum_{j=1}^{n}(D_j \vee T_j)} * \gamma_{R_i,R_i[D_i-T_i]^+} \\ &= \delta_{\sum_{j=1}^{n}(D_j \vee T_j)} \wedge \beta_{R_i,\sum_{j=1}^{n}(D_j \vee T_j)-[D_i-T_i]^+} \end{aligned} \quad (21)$$

Reporting (21) into (20) leads to:

$$\begin{aligned}&\delta_{\sum_{j=1}^{n}(D_j \vee T_j)} * \bigwedge_{i=1}^{n} \gamma_{R_i,R_i[D_i-T_i]^+} \\ &= \bigwedge_{i=1}^{n} \left(\delta_{\sum_{j=1}^{n}(D_j \vee T_j)} \wedge \beta_{R_i,\sum_{j=1}^{n}(D_j \vee T_j)-[D_i-T_i]^+}\right) \\ &= \delta_{\sum_{i=j}^{n}(D_j \vee T_j)} \wedge \bigwedge_{i=1}^{n} \left(\beta_{R_i,\sum_{j=1}^{n}(D_j \vee T_j)-[D_i-T_i]^+}\right) \blacksquare\end{aligned}$$

Let now present one useful lemma for computing our delay.

Lemma 4 (Horizontal delay with min of functions) *Let f, g, g' be three wide-sense increasing functions. Then,*

$$h(f, g \wedge g') = h(f, g) \vee h(f, g') \quad (22)$$

□

PROOF We are going to use two results: the first is the expression of $h(\cdot,\cdot)$ in term of deconvolution and the second is the left distributivity of max on deconvolution $(f \vee g) \oslash h = (f \oslash g) \vee (f \oslash h)$, and some trivial relationship on non-decreasing functions (like $(g \wedge h)^{-1} = g^{-1} \vee h^{-1}$ or $(g \vee g') \circ f = (g \circ f) \vee (g' \circ f)$).

$$\begin{aligned}h(f, g \wedge g') &= (((g \wedge g')^{-1} \circ f) \oslash \lambda_1)(0) \\ &= (((g^{-1} \vee g'^{-1}) \circ f) \oslash \lambda_1)(0) \\ &= (((g^{-1} \circ f) \vee (g'^{-1} \circ f)) \oslash \lambda_1)(0) \\ &= (((g^{-1} \circ f) \oslash \lambda_1) \vee ((g'^{-1} \circ f) \oslash \lambda_1))(0) \\ &= h(f, g) \vee h(f, g') \blacksquare\end{aligned}$$

Here comes the main result of the paper, the local application of eq. (4) on each server, parametrised by a set of $\theta_1, \ldots, \theta_n$ values, and the expression of the least solution.

Theorem 1 (End-to-end delay in simple FIFO topology) *Let R be a flow with a CPL function (under normal form of Definition 1) $\bigwedge_i \gamma_{r_i,b_i}$ as arrival curve. This flow goes thought a sequence of servers $S_1, \ldots, S_n$ with respective service curve $\beta_{R_1,T_1}, \ldots, \beta_{R_n,T_n}$. Each server S_i is also crossed by a flow R'_i with arrival curve $\gamma_{r'_i,b'_i}$, as illustrated on Figure 1 The policy of each server is FIFO. Then, the end-to-end delay of flow R is bounded by:*

$$\inf_{\forall i \in [1,n]:\theta_i \geqslant 0} h\left(\bigwedge_i \gamma_{r_i,b_i}, \mathop{\circledast}_{i=1}^{n} \beta_{R_i - r'_i, T^i_{\theta_i}} \mathbb{1}_{\{>\theta_i\}}\right) \quad (23)$$
$$= \begin{cases} \displaystyle\sum_{i=1}^{n}\left(T_i + \frac{b'_i}{R_i}\right) + \sum_{i=1}^{n} \frac{y_{k'_i} - (R_i - r'_i)x_{k'_i}}{R_i} \\ \qquad\qquad \text{if } \sum_{i=1}^{n} \frac{R_i - r'_i}{R_i} \leqslant 1 \\ \displaystyle\sum_{i=1}^{n}\left(T_i + \frac{b'_i}{R_i}\right) + \sum_{p(i)\leqslant k} e_i + e_q\left(1 - \sum_{p(i)\leqslant k} \frac{R_i - r'_i}{R_i}\right) \\ \qquad\qquad \text{otherwise} \end{cases}$$

with $T^i_{\theta_i} = \frac{R_i T_i + b'_i - r'_i\theta_i}{R_i - r'_i}$, $k'_i = \min\{j \,|\, r_j \leqslant R_i - r'_i\}$, $e_i \stackrel{\text{def}}{=} \frac{y_{k'_i} - (R_i - r'_i)x_{k'_i}}{R_i - r'_i}$, $p : [1,n] \mapsto [1,n]$ a permutation such that the sequence $\left(e_{p(i)}\right)$ is wide-sense increasing, and $q \stackrel{\text{def}}{=} \min\left\{p(i) \,\middle|\, 1 \geqslant \sum_{j=1}^{n-p(i)+1} \frac{R_i - r'_i}{R_i}\right\}$. □

PROOF Here is a sketch of proof. By lack of space, the full proof can be downloaded as an appendix on the author WEB page. A previous version can also be found in [5].

Since this work is a generalisation of [18] (it solves $\inf_{\forall i \in [1,n]:\theta_i \geqslant 0} h(\gamma_{r,b}, \circledast_{i=1}^{n} \beta_{R_i - r'_i, T^i_{\theta_i}} \mathbb{1}_{\{>\theta_i\}})$) the organisation of the proof is the same. In order to be self-sufficient, the proof will be completely done, but we try to highlight the common part and our specific contribution.

The first step consist in restricting the value domain of θ.

$$\begin{aligned}&\inf_{\forall i \in [1,n]:\theta_i \geqslant 0} h\left(f, \mathop{\circledast}_{i=1}^{n} \beta_{R_i - r'_i, T^i_{\theta_i}} \mathbb{1}_{\{>\theta_i\}}\right) = \\ &\qquad \inf_{\forall i \in [1,n]:\theta_i \geqslant T_i + \frac{b'_i}{R_i}} h\left(f, \mathop{\circledast}_{i=1}^{n} \beta_{R_i - r'_i, T^i_{\theta_i}} \mathbb{1}_{\{>\theta_i\}}\right)\end{aligned} \quad (24)$$

It comes from the fact that, if $\theta_i < T_i + \frac{b'_i}{R_i}$, then, $\beta_{R_i - r'_i, T^i_{\theta_i}} \mathbb{1}_{\{>\theta_i\}} = \beta_{R_i - r'_i, T^i_{\theta_i}}$, *i.e.* the test term has no effect. And, in this case, increasing the θ_i increases the value of the $\beta_{R_i - r'_i, T^i_{\theta_i}}$ term. This step was done in [18].

This allows to make a variable substitution, $\theta'_i = \theta_i + T_i + \frac{b'_i}{R_i}$.

The second step uses the fact that $h(f, g \wedge g') = h(f, g) \vee h(f, g')$, and some rewriting used in Lemma 2

to transform the convolution in minimum. It leads to term

$$\sum_{i=1}^{n}\left(T_i+\frac{b'_i}{R_i}\right)+ \inf_{\substack{\forall i\in[1,n]\\ \theta'_i\geqslant 0}}\left\{\sum_{i=0}^{n}\theta'_i+\bigvee_{i=1}^{n}\left[\frac{y_{k'_i}-R_i\theta'_i}{R_i-r'_i}-x_{k'_i}\right]^+\right\} \quad (25)$$

The proof in [18] gives the expression of the delay $h(\cdot,\cdot)$ by a computation of the inverse of the convolution term. We chose a more algebraic way, to use distributivity and reduce the problem to the computation of delay between a CPL and a rate-latency curve (Lemma 1).

To solve this expression, we have to minimise an expression $\inf_{x_i\geqslant 0}\left\{\sum_{i=1}^{n}x_i+\bigvee_{i=1}^{n}\left[b_i-a_ix_i\right]^+\right\}$ with $b_i\geqslant 0$ and $a_i\geqslant 1$. This work has been done in [18] (with a little restriction, $b_i>0$). We just have to reuse the result. ∎

5 Example and comparison

Since the equations are not very intuitive, we have done some experiment to get a global idea of the weakness and strong points of the method. The computations have been done with the prototype `NC-maude` [7, 4, 6].

Let us consider the configuration presented in Figure 1: a flow crossing three servers, sharing each server with a interfering flow.

To simplify the interpretation, consider that all servers and all interfering flows have the same characteristics. Each server has service of curve $\beta_{R,T}$, the considered flow has arrival curve $\lambda_R\wedge\gamma_{r,b}$ and the interfering flows have arrival curve $\lambda_R\wedge\gamma_{r',b'}$. We have considered 16 possible values for these parameters. For each configurations, we have computed the delay for crossing two and three servers, with three methods: LUBD (presented in section 3.1), local delay with shaping (presented in section 3.2) and our end-to-end computation with half modelling of shaping.

In the LUBD experiment, the shaping introduced by the link (the λ_R term) can not be modelled: the considered flow (resp. interfering flows) has arrival curve $\gamma_{r,b}$ (resp. $\gamma_{r',b'}$).

In the local delay with shaping, the considered flow has arrival curve $\lambda_R\wedge\gamma_{r,b}$ and the interfering flows have arrival curves $\lambda_R\wedge\gamma_{r',b'}$.

In our half modeling of shaping, the considered flow has arrival curve $\lambda_R\wedge\gamma_{r,b}$ but the interfering flows have arrival curves $\gamma_{r',b'}$.

Before all analyse, keep in mind that, in each server, there is an intrinsic delay of $T=1$ unit. It means that, whatever the input traffic and computation method are, the delay for crossing two servers is at least 2, and at least 3 for three servers.

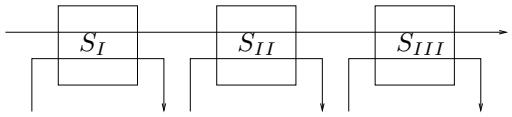

Figure 13. Test configuration

The four firsts topologies, configurations 1 up to 4, have been generated to see the influence of the different parameters, with some "normalised" values.

In the first configuration, all terms have value 1, except for the rates, which are equals to one third, for a global load $\rho=\frac{2}{3}$.

In configuration 2, burst of the interfering flow b' have been increased to 5 compared to configuration 1, in configuration 3, the crossing flow burst is increased to 5. On the opposite, in configuration 4, both bursts have been decreased to $\frac{1}{5}$.

What can be learn from theses examples will be generalised in all the following configuration.

First learning it that our method, which is a direct generalisation of the LUB method, is always better. Second, the relative gain decreases with the length of the path and the absolute gain is constant. Third, the gain of course depends on the burst of the crossing flow, which is not shaped in the LUB method.

Relative performances of the new method w.r.t the local shaping depends of course of the impact of the burst of the interfering flow, whose shaping is not modelled by the new method. The new method can be worst than the other one (21% worst for the configuration 1 for the delay on two servers). But the local shaping can not benefit from the PBOO principle, and the longest the path is, the better the relative performance of the new method is.

The configurations 5 to 8 are the same than configurations 1 to 4, except that the system rate have been multiplied by 5, leading to a global load of 13%, which is more common in embedded systems than the 67% one. In this case, the long term rate of the flows becomes negligible, and the best method is two times the local shaping one, and when the new one is better, the gain is really small, and when it is worst, it really is ($\approx 50\%$).

But a burst value equals to the rate value is not very realistic situation. On a 10Mb/s system, it means that is could exist a burst of 10Mb, *i.e.* 1.25Mo, *i.e.* more than 833 frames of 1500 octets (maximal size of an Ethernet frame) at a single instant. Then, configurations 9 to 11 are copies of the configurations 1 to 4 with a burst divided by 10 (considering burst of 83 maximal size Ethernet frames). In this case, the new method gain is between 12% and 22%.

But configurations 9 to 12 still have a load of 67%. Dividing this load by 2 gives configuration 13 to 16. In this case, the main part of the delay comes from the intrinsic de-

Configurations

Conf	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
R	1	1	1	1	5	5	5	5	1	1	1	1	1	1	1	1
T	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
r	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{6}$	$\frac{1}{6}$	$\frac{1}{6}$	$\frac{1}{6}$
b	1	1	5	$\frac{1}{5}$	1	1	5	$\frac{1}{5}$	$\frac{1}{10}$	$\frac{1}{10}$	$\frac{5}{10}$	$\frac{1}{50}$	$\frac{1}{10}$	$\frac{1}{10}$	$\frac{5}{10}$	$\frac{1}{50}$
r'	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{6}$	$\frac{1}{6}$	$\frac{1}{6}$	$\frac{1}{6}$
b'	1	5	1	$\frac{1}{5}$	1	5	1	$\frac{1}{5}$	$\frac{1}{10}$	$\frac{5}{10}$	$\frac{1}{10}$	$\frac{1}{50}$	$\frac{1}{10}$	$\frac{5}{10}$	$\frac{1}{10}$	$\frac{1}{50}$
$\rho = \frac{r+r'}{R}$	67%	67%	67%	67%	13%	13%	13%	13%	67%	67%	67%	67%	33%	33%	33%	33%

Delai R crossing $S_I; S_{II}$

LUB	5.50	13.50	11.50	2.70	2.61	4.21	3.47	2.12	2.35	3.15	2.95	2.07	2.32	3.12	2.80	2.06
Loc. Shap.	5.41	10.50	9.75	2.81	2.43	2.62	2.54	2.09	2.49	3.12	2.92	2.23	2.27	2.60	2.44	2.08
Half. Shap.	4.75	12.75	7.75	2.55	2.41	4.01	2.47	2.08	2.27	3.07	2.57	2.05	2.22	3.02	2.32	2.04

Delai R crossing $S_I; S_{II}; S_{III}$

LUB	7.50	19.50	13.50	3.90	3.81	6.21	4.67	3.16	3.45	4.65	4.05	3.09	3.42	4.62	3.90	3.08
Loc. Shap.	8.81	18.50	15.87	4.58	3.66	4.07	3.83	3.14	4.05	5.19	4.76	3.63	3.47	4.20	3.72	3.17
Half. Shap.	6.75	18.75	9.75	3.75	3.61	6.01	3.67	3.12	3.37	4.57	3.67	3.07	3.32	4.52	3.42	3.06

Gain on the new method for R crossing $S_I; S_{II}$

vs. LUB	13.63%	5.55%	32.60%	5.55%	7.61%	4.72%	28.65%	1.87%	3.19%	2.38%	12.71%	0.72%	4.13%	3.07%	17.14%	0.93%
vs. Loc. Shap.	12.30%	-21.42%	20.51%	9.46%	0.78%	-52.81%	2.83%	0.36%	8.69%	1.60%	11.96%	7.91%	2.34%	-16.30%	4.91%	1.79%

Gain on the new method for R crossing $S_I; S_{II}; S_{III}$

vs. LUB	10.00%	3.84%	27.77%	3.84%	5.21%	3.20%	21.29%	1.25%	2.17%	1.61%	9.25%	0.48%	2.80%	2.07%	12.30%	0.62%
vs. Loc. Shap.	23.46%	-1.35%	38.58%	18.23%	1.22%	-47.64%	4.06%	0.64%	16.80%	11.94%	22.83%	15.37%	4.29%	-7.54%	8.09%	3.48%

Table 1. Comparing methods on topology from Figure 13

lay, and methods difference are small, but the local shaping method is always better when the burst of the interfering flow is big.

6 Conclusion

Looking for worst case performance is an hard task, especially in distributed real-time system where the worst case is hard to determine [5]. Therefor, we are dealing with methods making pessimistic approximations, and trying to reduce this pessimism.

Going from basics works of [12, 13] handling only token-bucket ($\gamma_{r,b}$) traffic, two extensions have been made to reduce this pessimism: [15] handles on each server the shaping introduced by physical links, and [18] models an end-to-end FIFO service to get the benefit of the PBOO result. This works tried to join the two works and have the benefit of both. But for technical reasons, it only handles shaping on the considered flow, not on the interfering one. Then, the new method is always better than the one of [18] (it is a direct generalisation), but performances w.r.t. the local shaping depends on the burst size of the interfering flows.

This of course incites us to study the general case, with shaping on considered and interfering flow. We are currently working on the subject, but it seems really harder.

References

[1] H. Bauer, J.-L. Scharbarg, and C. Fraboul. Applying and optimizing trajectory approach for performance evaluation of AFDX avionics network. In *Proceedings of the 14th IEEE international conference on Emerging technologies & factory automation (ETFA'09)*, pages 690–697, Piscataway, NJ, USA, 2009. IEEE Press.

[2] A. Bouillard, L. Jouhet, and E. Thierry. Tight performance bounds in the worst-case analysis of feed-forward networks. In *Proceedings of the 29th Conference on Computer Communications (INFOCOM 2010)*, pages 1–9, march 2010.

[3] A. Bouillard and E. Thierry. An algorithmic toolbox for network calculus. *Discrete Event Dynamic Systems*, 17(4), october 2007. http://www.springerlink.com/content/876x51r6647r8g68/.

[4] M. Boyer. NC-maude home page. http://www.onera.fr/staff/marc-boyer/tools.php.

[5] M. Boyer. Delay in fifo rate-latency nodes shared by cpl flows. Technical Report RT 2/16417, ONERA, 2010.

[6] M. Boyer. NC-maude: a rewriting tool to play with network calculus. In T. Margaria and B. Steffen, editors, *Proceedings of the 4th International Symposium On Leveraging Applications of Formal Methods, Verification and Validation (ISoLA 2010)*, LNCS. Springer, 2010.

[7] M. Boyer. NC-maude: maude for computation of worst bounds on real-time (embedded) networks. Technical Report 1/16417, ONERA, 2010.

[8] M. Boyer and A. Chagou. Managing aggregation of shaped leaky buckets flows through GPS node in network calculus. Rapport de recherche IRIT/RR–2007-21–FR, IRIT/ENSEEIT – University of Toulouse, Toulouse, France, octobre 2007. http://irt.enseeiht.fr/boyer/Publications.html#IRIT-2007-21.

[9] M. Boyer and C. Fraboul. Tightening end to end delay upper bound for AFDX network with rate latency FCFS servers using network calculus. In *Proc. of the 7th IEEE Int. Workshop on Factory Communication Systems Communication in Automation (WFCS 2008)*, pages 11–20. IEEE industrial Electrony Society, May 21-23 2008.

[10] C.-S. Chang. *Performance Guarantees in communication networks*. Telecommunication Networks and Computer Systems. Springer, 2000.

[11] COINC home page. http://www.istia.univ-angers.fr/~lagrange/software.php.

[12] R. L. Cruz. A calculus for network delay, part I: Network elements in isolation. *IEEE Transactions on information theory*, 37(1):114–131, January 1991.

[13] R. L. Cruz. A calculus for network delay, part II: Network analysis. *IEEE Transactions on information theory*, 37(1):132–141, January 1991.

[14] F. Frances, C. Fraboul, and J. Grieu. Using network calculus to optimize AFDX network. In *Proceeding of the 3thd European congress on Embedded Real Time Software (ERTS06)*, Toulouse, January 2006.

[15] J. Grieu. *Analyse et évaluation de techniques de commutation Ethernet pour l'interconnexion des systèmes avioniques*. PhD thesis, Institut National Polytechnique de Toulouse (INPT), Toulouse, Juin 2004.

[16] J.-Y. Le Boudec and P. Thiran. *Network Calculus*, volume 2050 of *LNCS*. Springer Verlag, 2001. http://lrcwww.epfl.ch/PS_files/NetCal.htm.

[17] L. Lenzini, L. Martorini, E. Mingozzi, and G. Stea. Tight end-to-end per-flow delay bounds in fifo multiplexing sink-tree network. *Performance Evaluations*, 63:956–987, 2005.

[18] L. Lenzini, E. Mingozzi, and G. Stea. Delay bounds for FIFO aggegates: a case study. *Computer Communications*, 28:287–299, 2004.

[19] L. Lenzini, E. Mingozzi, and G. Stea. End-to-end delay bounds in fifo-multiplexing tandems. In P. Glynn, editor, *Proc. of the 2nd International Conference on Performance Evaluation Methodologies and Tools (ValueTool07*, Nantes, France, October 23-25 2007. ICST.

[5] It could be NP-hard [2, § III.D].