Bounding the delays of the MPPA network-on-chip with network calculus: Models and benchmarks

Marc Boyer, Amaury Graillat, Benoît Dupont de Dinechin, Jörn Migge

Bounding the delays of the MPPA Network-on-Chip with network calculus: models and benchmarks

Marc Boyer [1], Amaury Graillat [2,3],
Benoît Dupont de Dinechin [2], Jörn Migge [4]

[1] ONERA/DTIS, Université de Toulouse
F-31055 Toulouse – France
[2] Kalray S.A
F-38330 Montbonnot Saint Martin – France
[3] Université Grenoble Alpes, Verimag
38000 Grenoble – France
[4]RealTime-at-Work
F-54600 Villers-lès-Nancy – France

March 15, 2021

Abstract

The Kalray MPPA2-256 processor integrates 256 processing cores and 32 management cores on a chip. Theses cores are grouped into clusters and clusters are connected by a high-performance network on chip (NoC). This NoC provides hardware mechanisms (ingress traffic limiters) that can be configured to offer service guarantees.

This paper introduces a network calculus formulation, designed to configure the NoC traffic limiters, that also computes guarantee upper bounds on the NoC traversal latencies. This network calculus formulation accounts for the traffic shaping performed by the NoC links, and can be solved using linear programming. This paper then shows how existing network calculus approaches (the Separated Flow Analysis – SFA ; the Total Flow Analysis – TFA ; the Linear Programming approach – LP) can be adapted to analyze this NoC. The delay bounds obtained by the four approaches are then compared on two case studies: a small configuration coming from a previous study, and a realistic configuration with 128 or 256 flows.

From theses cases studies, it appears that modeling the shaping introduced by NoC links is of major importance to get accurate bounds. And when all packets have the same size, modeling it reduces the bound by 20%-25% on average.

1 Introduction

As embedded systems require ever increasing computing performance while operating at low power, multicore-based systems appear as a solution. Moreover, in order to host time-critical functions, such platforms must provide some response time guarantees. And as in any distributed platform, bounding the communication delay is a key point of real-time performances.

The Kalray MPPA2 processor has been designed to offer high computing performances and energy efficiency on time-constrained applications. In particular, its network on chip (NoC) provides hardware mechanisms (ingress traffic limiters) that can be configured to offer service guarantees such as flow minimum bandwidth, flow maximum delay, and congestion-free operations. But since the computation of the exact worst latencies can be too complex, as shown in Bouillard et al. (2010), one has to rely on delay bounds.

Getting the best capabilities from such a platform requires efficient methods to compute communication delay bounds. This paper presents and compare several of them, all based on deterministic network calculus. Whereas a large literature on the computation on delay bounds for NoCs exists, not many deal with real implemented architectures (Section 4). The MPPA NoC is an interesting target for analysis, as its architecture is designed to minimize implementation complexity while ensuring service guarantees.

This paper presents the Kalray MPPA NoC architecture in Section 2, whose key elements are the ingress traffic limiters and the router switches. Section 3 provides the necessary background on deterministic network calculus. Section 4 presents the state of the art. Section 5 introduces notations commons to all the methods presented in this article. Section 6 introduces a new "explicit linear" method for computing the delay bounds, which maps the network calculus equations to a Mixed-Integer Linear Problem (MILP) formulation solvable in polynomial time. Then, Section 7 shows how existing network calculus approaches for computing latencies (Total Flow Analysis – TFA, Separated Flow Analysis – SFA) can be adapted to analyze this NoC, and how the common case where all packets have the same size can be modeled. Finally, all these methods are run in Section 8 on two case studies. The first has been already presented in Dupont de Dinechin and Graillat (2017). It allows to compare the new methods to already published results. Moreover, it is small enough to allow an interpretation of the results. The second case study is more realistic: each of the 16 clusters sends 4 or 8 independent data flows. Section 8.6 gives some insight on the mathematical reasons for the observed upper bound differences.

2 Description of the NoC

The MPPA2-256 processor integrates 256 processing clustered with 16 compute clusters and 2 I/O clusters. The clusters communicate through a NoC, with one node per compute cluster and 8 nodes per I/O cluster. The MPPA2 NoC is a direct network based on a 2D-torus topology extended with extra links

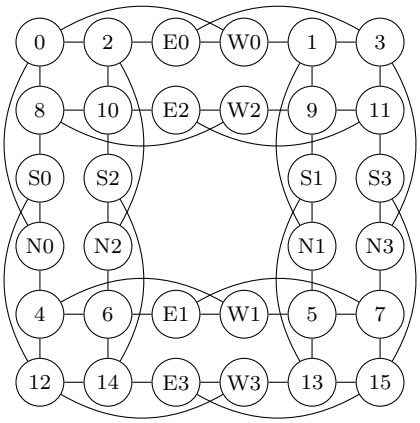

Figure 1: MPPA2 NoC topology unfolded (I/O nodes are labeled N0..N3, E0..E3, S0..S3, W0..W3).

connected to the otherwise unused ports of the NoC nodes on the I/O clusters (see Figure 1). More details can be found in Saidi et al. (2015).

The MPPA2 NoC implements wormhole switching with source routing and without virtual channels. With wormhole switching, a packet is decomposed into flits (of 32-bits on the MPPA2 NoC), which travel in a pipelined fashion across the network elements, with buffering and flow control applied at the flit level. The packet follows a route determined by a bit string in the header. The packet size is between 2 and 71 flits.

Once a buffer is full, the flow control mechanism of wormhole switching requires that the previous router stores flits instead of forwarding them. This *back pressure* mechanism can go back up to the source, a situation called *congestion*. Congestion can also lead to deadlock of a wormhole switching NoC when flows are not routed feed-forward, as presented in Dupont de Dinechin et al. (2014).

Each MPPA2 NoC node is composed of a cluster interface and a router (Fig. 2). They are eight traffic limiters in the cluster interface. Each one implements a token-bucket traffic shaper with configurable burst b and rate r. The burst parameter must be large enough to allow to send one full packet at link speed (one flit per cycle) before being limited by the budget (as illustrated in Figure 3 – the exact relation between r, b and the packet size will be given in eq. (20)). Each router is connected to its four neighbors and to the local cluster (respectively called North, West, South, West and Local). Each output port has four (or five) queues, to store waiting flits. They are arbitrated using a per packet round-robin algorithm.

Whereas the back pressure mechanism of the wormhole switching can lead to complex interactions between flows, and even deadlocks, one may avoid its activation by preventing the complete filling of queues. This can be done by: 1) defining a static set of data flows; 2) allocating to each flow a traffic limiter

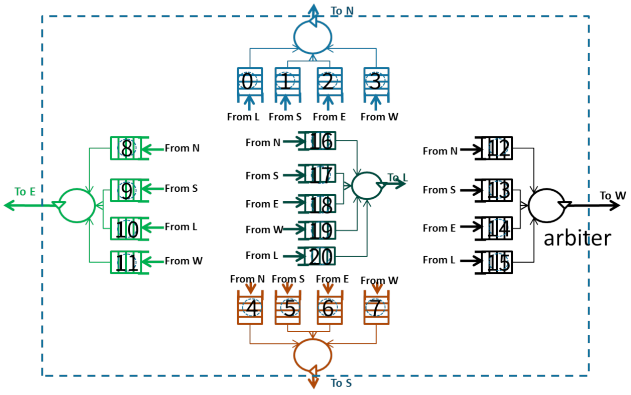

Figure 2: Structure of a MPPA2 NoC router.

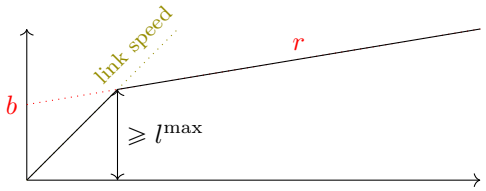

Figure 3: Token-bucket traffic limiter.

and a route, with and adequate configurations of the traffic limiters. Assuming the route of each data flow is determined (for example using the techniques in Dupont de Dinechin et al. (2014)), a network calculus formulation can be used to compute the configuration of each traffic limiter.

3 Deterministic Network Calculus

Deterministic network calculus is a theory designed for the performance analysis of computer networks. Its main purpose is to compute upper bounds on delay and buffer memory usage in networks Cruz (1991).

The following is a short summary of the deterministic network calculus theory, in order to present its main results and set the notations. All results presented in this section can be found in Chang (2000), Le Boudec and Thiran (2001), except when a specific reference is given.

3.1 Mathematical background and notations

Let $\mathcal{F}$ denote the set of functions from $\mathbb{R}^+$ to $\mathbb{R}^+$, and $\mathcal{F}^\uparrow$ the subset of non-decreasing functions: $\mathcal{F}^\uparrow \stackrel{def}{=} \{f \in \mathcal{F} \mid \forall t, d \in \mathbb{R}^+ : f(t+d) \geqslant f(t)\}$.

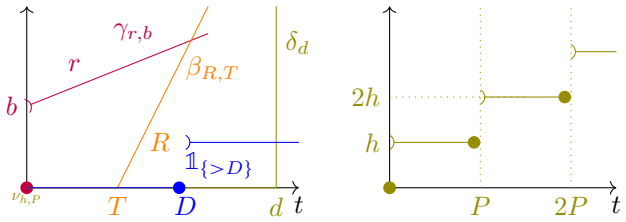

Figure 4: Common curves in network calculus.

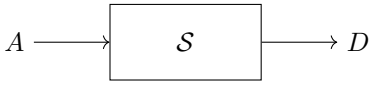

Figure 5: A server crossed by a flow.

Since one may need to project functions in $\mathcal{F}$ or $\mathcal{F}^\uparrow$, let define $[f]^+ \stackrel{def}{=} \max(f, 0)$, $f_\uparrow : \mathcal{F} \to \mathcal{F}^\uparrow$, $f_\uparrow(t) \stackrel{def}{=} \sup_{0 \leqslant s \leqslant t} f(s)$, and $[f]^+_\uparrow \stackrel{def}{=} ([f]^+)_\uparrow$.

The composition operator is denoted $\circ$: $(f \circ g)(x) = f(g(x))$. The ceiling is denoted $\lceil . \rceil$ and the flooring $\lfloor . \rfloor$: $\lceil 1.5 \rceil = 2$, $\lfloor 1.5 \rfloor = 1$.

The network calculus relies on the (min,+) dioid. On this structure, convolution $*$ and deconvolution $\oslash$ operators are defined as:

$$(f * g)(t) \stackrel{def}{=} \inf_{0 \leqslant s \leqslant t} \{f(t-s) + g(s)\}, \tag{1}$$

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

The point-wise minimum operator of functions is denoted $\wedge$.

Some functions, plotted in Figure 4, are commonly used: the delay function $\delta_T(t) = 0$ if $t \leqslant T$, ∞ otherwise; the token-bucket function $\gamma_{r,b}(t) = (rt + b) \wedge \delta_0(t)$; the rate-latency function $\beta_{R,T}(t) = R\,[t - T]^+$; the test function $\mathbb{1}_{\{>D\}}(t) = 1$ if $t > D$, 0 otherwise; the pure rate $\lambda_R = \beta_{R,0}$; and the stair-case $\nu_{h,P}(t) = h \left\lceil \frac{t}{P} \right\rceil$.

3.2 Modeling systems within network calculus

In network calculus, a flow is modeled by its *cumulative function*, a function $A \in \mathcal{F}^\uparrow$, left-continuous[1], with $A(0) = 0$. The semantics of such a function is that $A(t)$ represents the total amount of data sent by the flow up to time t.

A *server* is a relation $\mathcal{S}$ between two cumulative functions, such that for any arrival A, it exists a departure D such that $(A, D) \in \mathcal{S}$. Moreover, for any $(A, D) \in \mathcal{S}$, $D \leqslant A$, meaning that the departure of a bit of data always occurs after its arrival. One may also denote by $A \xrightarrow{\mathcal{S}} D$ the relation $(A, D) \in \mathcal{S}$.

[1]For a discussion on continuity in network calculus, see Boyer et al. (2013) or § 1.3 in Bouillard et al. (2018).

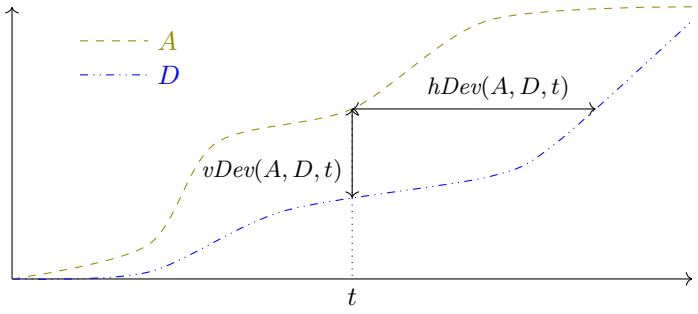

Figure 6: Delay and backlog between arrival and departure flows.

When order of data within a flow is preserved, the *delay* at time t is defined as $hDev(A, D, t)$

$$hDev(A, D, t) \stackrel{def}{=} \inf \left\{ d \in \mathbb{R}^+ \mid A(t) \leqslant D(t+d) \right\}. \tag{3}$$

The *backlog* at time t is $vDev(A, D, t)$,

$$vDev(A, D, t) \stackrel{def}{=} A(t) - D(t). \tag{4}$$

The worst delay (resp. backlog) associated to the pair (A, D) is the supremum of the delay (resp. backlog) for all time t.

$$hDev(A, D) \stackrel{def}{=} \sup_{t \in \mathbb{R}^+} hDev(A, D, t), \tag{5}$$

$$vDev(A, D) \stackrel{def}{=} \sup_{t \in \mathbb{R}^+} vDev(A, D, t). \tag{6}$$

A *n-server* $\mathcal{S}$ is a relation that associates to each vector of arrival cumulative functions $(A_1, \ldots, A_n)$ at least one vector of departure cumulative functions $(D_1, \ldots, D_n)$ such that $\forall i \in [1, n] : D_i \leqslant A_i$.

Given a n-server, its *aggregate server* $\mathcal{S}_\Sigma$ is defined by $(A_1, \ldots, A_n) \xrightarrow{\mathcal{S}} (D_1, \ldots, D_n) \Rightarrow A \xrightarrow{\mathcal{S}_\Sigma} D$ with $A = \sum_{i=1}^{n} A_i$, $D = \sum_{i=1}^{n} D_i$; and for any $i \in [1, n]$, its *residual server* $\mathcal{S}_i$ is defined by $(A_1, \ldots, A_n) \xrightarrow{\mathcal{S}} (D_1, \ldots, D_n) \Rightarrow A_i \xrightarrow{\mathcal{S}_i} D_i$.

3.3 Contracts

A cumulative function A is said to have a function $\alpha \in \mathcal{F}$ as *maximal arrival curve* if

$$\forall t, d \in \mathbb{R}^+ : A(t+d) - A(t) \leqslant \alpha(d). \tag{7}$$

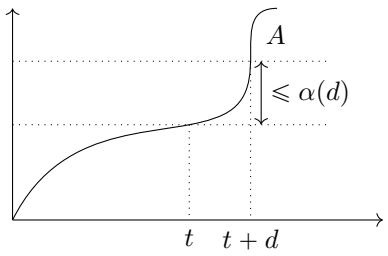

Figure 7: Arrival curve.

This condition is equivalent to $A \leqslant A * \alpha$. The adjective "maximal" is often omitted since even if it exists a notion of minimal arrival curve, it is not commonly used, and in particular it is not used in this article.

Two contracts on the minimal capacity of a server exist: a *simple* minimal service and a *strict* minimal service.

Given a server $\mathcal{S}$, it offers a *simple minimal service* of curve $\beta \in \mathcal{F}$ if

$$\forall A \xrightarrow{\mathcal{S}} D : D \geqslant A * \beta. \tag{8}$$

This server offers a *strict minimal service* of curve $\beta \in \mathcal{F}$ if

$$\forall A \xrightarrow{\mathcal{S}} D, \forall t, d \geqslant 0, \forall x \in [t, t+d), A(x) > D(x) \implies D(t+d) - D(t) \geqslant \beta(d). \tag{9}$$

An interval $[t, t+d)$ such that $\forall x \in [t, t+d) : A(x) > D(x)$ is called a *backlogged interval* or *backlogged period*.

If a server offers a strict minimal service of curve β, it also offers a simple minimal service of curve β. Servers that are work-conserving, that is, do not idle as long as there is data to transmit, offer a strict service.

The maximal capacity of a server is also of interest: given an arrival/departure pair $A \xrightarrow{\mathcal{S}} D$, the upper bounds on the delay and backlog of the flow in the server $\mathcal{S}$ are influenced by the minimal performance of the server, but the shape of the departure cumulative functions is influenced by the maximal capacity of the server, as will be shown in Theorem 1.

Let $\sigma \in \mathcal{F}$, a server $\mathcal{S}$ is a σ-shaper if $\forall A \xrightarrow{\mathcal{S}} D$, D has σ as arrival curve.

3.4 Main results

If the contracts on the arrival and the server are known, one can compute upper bounds on the delay, backlog, and also compute the contract on the departure (its allows to propagate the computation).

Theorem 1 (Network calculus bounds)**.** *Let $\mathcal{S}$ be a server, and $A \xrightarrow{\mathcal{S}} D$ two arrival and departure cumulative functions. If $\mathcal{S}$ offers a minimal service of*

curve β, and $\mathcal{S}$ is a σ-shaper, and A has α as arrival curve, then

$$hDev(A, D) \leqslant hDev(\alpha, \beta), \qquad vDev(A, D) \leqslant vDev(\alpha, \beta), \tag{10}$$

and D has α' as arrival curve, with

$$\alpha' = (\alpha \oslash \beta) \wedge \sigma. \tag{11}$$

This theorem computes local bounds, but when considering a sequence of servers, a tighter bound can be computed.

Theorem 2 (Pay burst only once)**.** *Let $\mathcal{S}_1, \mathcal{S}_2$ be two servers offering respectively a minimal simple service of curve β_1, β_2, and consider a flow crossing both in sequence with A, B, C the respective cumulative curves (*i.e. $A \xrightarrow{\mathcal{S}_1} B \xrightarrow{\mathcal{S}_2} C$*). Then, the sequence $\mathcal{S}_1, \mathcal{S}_2$ is a server offering a minimal simple service of curve $\beta_1 * \beta_2$.*

This result is interesting since it gives lower bounds than the sum of local delays[2].

Theorem 3 (Blind multiplexing)**.** *Let $\mathcal{S}$ be a n-server such that $\mathcal{S}_\Sigma$ offers a minimal strict service of curve β. Then, if each arrival A_j has α_j as arrival curve, for any $i \in [1, n]$, the residual server $\mathcal{S}_i$ offers the minimal simple service of curve*

$$\beta_i^{blind} = \left[\beta - \sum_{j \neq i} \alpha_j\right]^+_\uparrow. \tag{12}$$

The result was in (Le Boudec and Thiran, 2001, Thm. 6.2.1) without the non-decreasing closure that has been added in Bouillard (2011). It is also known as "arbitrary multiplexing" since it can be applied on any service policy.

When several flows share a queue with First-In First Out (FIFO) policy, one can derive a per flow residual service.

Theorem 4 (FIFO multiplexing)**.** *Let $\mathcal{S}$ be a n-server such that $\mathcal{S}_\Sigma$ offers a minimal simple service of curve β. Then, if each arrival A_j has α_j as arrival curve, for any $i \in [1, n]$, the residual server $\mathcal{S}_i$ offers the minimal simple service of curves*

$$\beta_i^{g\text{-}FIFO} = \delta_d \text{ with } d = hDev\left(\sum_{j=1}^{n} \alpha_j, \beta\right), \tag{13}$$

$$\beta_i^{\theta-FIFO} = \left[\beta - \sum_{j \neq i} \alpha_j * \delta_\theta\right]^+ \wedge \delta_\theta, \forall \theta \in \mathbb{R}^+. \tag{14}$$

In fact, they are two results for the FIFO policy. One may either compute the delay of the aggregate server, d, or choose one θ for each flow and use

[2] *i.e.* $hDev(\alpha, \beta_1 * \beta_2) \leqslant hDev(\alpha, \beta_1) + hDev(\alpha', \beta_2)$ with $\alpha' = \alpha \oslash \beta_1$

$\beta_i^{\theta-\text{FIFO}}$. In this case, the challenge is the choice of the θ value (that will be discussed in Sections 4 and 7.2). Proofs can be found at Theorems 7.4 and 7.5 in Bouillard et al. (2018).

Proposition 1 (Burstiness increase due to FIFO, general case)**.** *Let $\mathcal{S}$ be a n-server such that $\mathcal{S}_\Sigma$ offers a minimal simple service of curve $\beta_{R,T}$. Assume that the flow of interest A_i has arrival curve γ_{r_i,b_i}, and that the aggregate flow $A_{\neq i} = \sum_{j\neq i} A_j$ has a sub-additive arrival curve $\alpha_{\neq i}$, with $r_{\neq i}$ its long term rate. Then, if $r_i + r_{\neq i} < R$, then the departure flow D_i has arrival curve $\gamma_{r_i,b_i'}$ with*

$$b_i' = b_i + r_i\left(T + \frac{B}{R}\right), \qquad B = \sup_{u\geqslant 0}\left\{\alpha_{\neq i}(u) + r_i u - Ru\right\}.$$

The previous proposition is the re-writing of Theorem 6.4.1 from Le Boudec and Thiran (2001).

Corollary 1 (FIFO and token-bucket arrival curves)**.** *Let $\mathcal{S}$ be a n-server such that $\mathcal{S}_\Sigma$ offers a minimal simple service of curve $\beta_{R,T}$. Assume that each arrival A_j has γ_{r_j,b_j} as arrival curve, with $\sum_{j=1}^n r_j < R$. Then for any $i \in [1,n]$, the residual server $\mathcal{S}_i$ offers the simple minimal service of curve β_{R_i,T_i} with $R_i = R - \sum_{j\neq i} r_j$, $T_i = T + \frac{\sum_{j\neq i} b_j}{R}$, and the departure D_i has arrival curve $\gamma_{r_i,b_i'}$ with $b_i' = b_i + r_i T_i$.*

The previous corollary is the re-writing of Cor. 6.2.3 from Le Boudec and Thiran (2001).

Theorem 5 (Residual service of RR)**.** *Let $\mathcal{S}$ be a n-server shared by n flows, denoted by $(A_1,\ldots,A_n) \xrightarrow{S} (D_1,\ldots,D_n)$, applying a round robin policy. For any $i \in [1,n]$, let $l_i^{\max}$ and $l_i^{\min}$, some upper and lower packet sizes for the flow i.*

If $\mathcal{S}_\Sigma$ offers a strict service of curves β, then the residual server $\mathcal{S}_i$ offers the residual strict service of curves

$$\beta_i^{RR} = \left(\lambda_1 * \nu_{l_i^{\min}, l_i^{\min}+L_{\neq i}^{\max}}\right) \circ \left(\beta - L_{\neq i}^{\max}\right), \tag{15}$$

$$\beta_i^{RR\text{-}lin} = \frac{l_i^{\min}}{l_i^{\min} + L_{\neq i}^{\max}}\left[\beta - L_{\neq i}^{\max}\right]^+ \tag{16}$$

with $L_{\neq i}^{\max} = \sum_{j\neq i} l_j^{\max}$. If $\beta(t) = Rt$, then $\beta_i^{RR\text{-}lin} = \beta_{R_i^{RR},T_i^{RR}}$ with

$$R_i^{RR} = R\frac{l_i^{\min}}{l_i^{\min} + L_{\neq i}^{\max}}, \qquad T_i^{RR} = \frac{L_{\neq i}^{\max}}{R}. \tag{17}$$

This theorem gives three expressions of residual services, but in fact there is only one, since $\beta_i^{\text{RR-lin}}$ is just a linear lower bound of β_i^{RR}, and $\beta_{R_i^{\text{RR}},T_i^{\text{RR}}}$ is the expression of $\beta_i^{\text{RR-lin}}$ when the aggregate service is a constant rate. The curve β_i^{RR} captures the fact that a Round-Robin offers to a flow of interest an

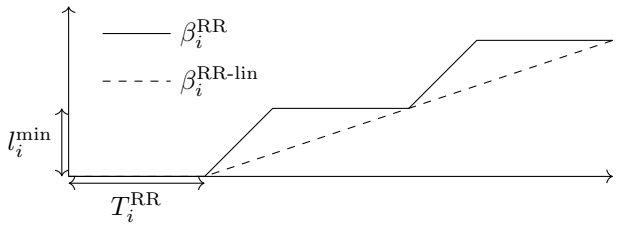

Figure 8: Illustration of WRR residual service, with $\beta(t) = Rt$.

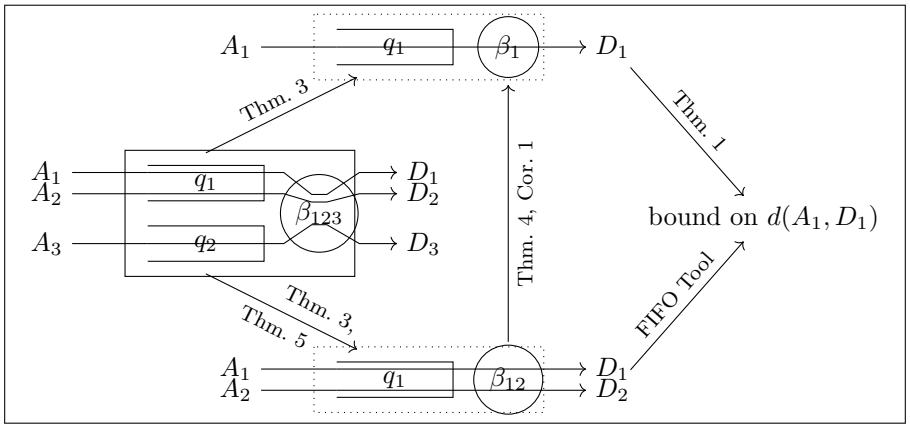

Figure 9: Decomposition of output port in residual servers.

alternation of waiting time (while frames of other flows are transmitted) and service of one single frame. When the aggregate service if a constant rate R, this is an alternation of plateaus (waiting time) and increase with slope R. Since the service is a guaranteed one, it considers that all competing flows send frames of maximal size, whereas the flow of interest sends frames of minimal size. The function $\beta_i^{\mathrm{RR\text{-}lin}}$ only considers a maximal latency, corresponding to the plateau β_i^{RR} and a long term ratio $\frac{l_i^{\min}}{l_i^{\min}+L_{\neq i}^{\max}}$. Their relation is illustrated on Figure 8. The proof can be found in (Bouillard et al., 2018, Thm. 8.6) or Boyer et al. (2019).

3.5 Analysis principles

3.5.1 Single node analysis

When an output port implements a round robin policy between queues, and each input queue is shared by several flows, there exists several ways to compute the delay associated to each flow. Consider Figure 9, where an output port is shared by three flows, A_1, A_2, A_3, with A_1, A_2 buffered in queue q_1 and flow A_3 buffered in queue q_2. Assume we are interested by the flow A_1. From the initial

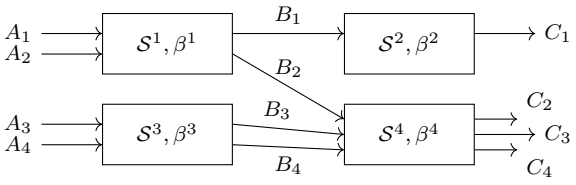

Figure 10: Simple topology.

configuration (on the middle left), with strict service of curve β_{123}, one may compute a residual server, $\mathcal{S}_1$, with service β_1, considering blind multiplexing (cf. Theorem 3). But one also may first reduce the system to a FIFO one, $\mathcal{S}_{12}$, with simple service β_{12}, using either Theorem 3 or Theorem 5. Then, one may either use a tool dedicated to FIFO network, or use Theorem 4 or Corollary 1.

Since each theorem provides a different residual curve, the choice of one or the other may give a different bound on the delay. They all are correct, but some are smaller.

For example, when going from $\mathcal{S}$ to $\mathcal{S}_{12}$, if A_3 uses less than half of the bandwidth, it may be better to use Theorem 3 on blind multiplexing. Indeed, the round robin policy, with equal packets size, will guaranty to queue q_1 half of the bandwidth. If A_3 uses only one third of the bandwidth, blind multiplexing will guaranty queue q_1 two thirds of this bandwidth

3.5.2 Feed-forward network analysis

There exist several ways to bound the end-to-end delay of a given flow, and they may depend on the network topology and the routing of flows. Let F^j denotes the set of flows crossing a server $\mathcal{S}^j$.

The simplest one, the Total Flow Analysis (TFA), initially defined in Schmitt and Zdarsky (2006), computes one bound d^j for each server (a "local" bound), and for a given flow, does the sum of all servers its crosses $d_i^{\text{TFA}} = \sum_{f_i \in F^j} d^j$. It will be presented in details in Section 7.1. In the topology of Figure 10, TFA will compute one delay d^i for each server $\mathcal{S}^i$, and the delay for the flow f_4 (of cumulative functions A_4, B_4, C_4) will be bounded by $d^3 + d^4$.

The most famous one, the Separated Flow Analysis (SFA) computes, for a given flow f_i, for each crossed server $\mathcal{S}^j$, a residual service β_i^j. Then, using the Pay Burst Only Once principle (Theorem 2), one gets an end-to-end service $\beta_i^{\text{SFA}} = \ast_{f_i \in F^j} \beta_i^j$ that allows to compute $d^{\text{SFA}} = hDev(\alpha_i, \beta_i^{\text{SFA}})$ a bound on the end-to-end delay. In the topology of Figure 10, to bound the delay of f_4, SFA will compute β_4^3 (resp. β_4^4), a residual service for the flow f_4 in the server $\mathcal{S}^3$ (resp. $\mathcal{S}^4$), and the delay will be bounded by $hDev(\alpha_4, \beta_3^3 * \beta_3^4)$.

In both SFA and TFA, the computation of the residual service depends on the scheduling policy. None of the algorithm specifies how to compute the arrival curves of the interfering flows (for the flow B_4, the interfering flows are B_2 and B_3). And both handle only feed-forward networks, *i.e.* where there is no cyclic

dependency between flows.

SFA is often considered as better than TFA[3]. But most of the studies have considered only blind multiplexing. As will be shown in this study, when considering FIFO policy, the results can be different. The reason may be that there is no well known strategy to get a "good" residual service for FIFO.

Several approaches has been developed to extend the TFA and SFA ideas (cf. Section 4), most of them assuming that all arrival (resp. service) curves are piece-wise linear concave (resp. convex). One of them, called "LP" in this paper, encodes the network behavior as one mixed-integer linear program and is then able to compute the exact worst case, at the expense of high computation times.

4 State of the art

They have been several studies designed to compute upper bounds on the worst case traversal times (WCTT) of a NoC by a set of data flows. Nevertheless, very few address the Kalray MPPA NoC architecture.

An overview of the state of the art of NoC performance evaluation (up to 2013) can be found in Kiasari et al. (2013).

Most NoCs use a wormhole switching mechanisms: a packet is decomposed as a sequence of flits (typically of 64 or 128 bits), and the flits are forwarded in a cut-through way once the routing decision has been made, based on the header of the packet. This mechanism allows a router to forward the head of a packet before the reception of the full packet. A credit-based mechanism ensures that no buffer overflows: if a destination buffer is full, the switch stops forwarding flits. This can lead to a local buffer filling and then the previous switch must also stop to send flits, and so on, up to the source. This mechanism is called *back-pressure*.

In a real-time environment, the back-pressure mechanism may create large latencies and is quite difficult to analyze. Then, in case of real-time constraints, one often tries to avoid back-pressure activation.

4.1 Approaches without network calculus

TDMA access upon wormhole switching One solution to avoid the back-pressure activation is to build a global time-based schedule (Time Division Multiple Access, TDMA), where times slots are reserved to data flows, in a way such that no contention occurs in the buffers, as in Carle et al. (2014), Perret et al. (2016a), Perret et al. (2016b).

Wormhole switching, virtual channels and static priorities The use of *virtual channels* allows to reduce the number of conflicts in buffer use and so the number of activations of the back-pressure mechanism.

[3] "In network calculus, the Total Flow Analysis (TFA) had been abandoned since it is inferior to other methods." (Bondorf and Schmitt, 2016, §7)

For example, an active community considers NoC with wormhole switching, in each router, preemption at the flit level and static priorities scheduling between virtual channels. Moreover, it is often assumed that the number of virtual channel is not less than the maximum number of contentions in each port, as in Shi and Burns (2008), Nikolić et al. (2016), Burns et al. (2014), Xiong et al. (2017) or Nikolić et al. (2018). Note that with such assumptions, the back-pressure mechanisms of the wormhole switching is only due to higher priority flows.

Wormhole with back-pressure A few papers address the problem of wormhole switching with back-pressure activation within the same priority level.

The *recursive calculus* was designed in Ferrandiz et al. (2009), Ferrandiz et al. (2011) to compute bounds on the SpaceWire technology, a wormhole-based technology. The recursive calculus is one of the rare methods that fully take into account the back-pressure mechanism of the wormhole switching. It has been adapted to the Kalray MPPA NoC in Ayed et al. (2016) and compared with a network-calculus based approach Dupont de Dinechin et al. (2014) on an example (a more detailed comparison can be found in Boyer et al. (2019)).This recursive calculus approach has been enhanced in Abdallah et al. (2015) to take into account the pipeline effect of the cut-through forwarding in the wormhole switching, considering a NoC with input-queuing and round-robin arbitration.

The Compositional Performance Analysis (CPA, Henia et al. (2005)) is a theory that, like network calculus, uses functions to bounds the flow shape, but, unlike network calculus, uses a busy-period based analysis to compute the per node delay. In Tobuschat and Ernst (2017), the authors develop a CPA-based method to compute the delay bounds on a wormhole NoC, with back-pressure activation and taking into account the input flow shapes.

The trajectory approach, originally developed for Ethernet networks in Martin and Minet (2004) and corrected in Li et al. (2014), has been adapted to NoC, considering a system with input queuing, FIFO arbitration and back-pressure activation in Papastefanakis et al. (2015).

Last, two studies take into account the back-pressure within network calculus framework, and they are presented in the next section.

4.2 Network calculus based approaches

Since the back-pressure is activated once a buffer is full, one way to avoid its activation consists in statically ensuring that it will never occur, by adequate configuration of the traffic limiters. To do so, one may use the network calculus theory that is devoted to the computation on upper bounds on buffer occupancy and delay.

From the network calculus point of view, when the back-pressure mechanism is not activated, the MPPA NoC simply appears as a network using a round robin arbiter and cut-through forwarding. The SFA and TFA principles presented in Section 3.5 are some basic principles to analyze a full network, independently of the scheduling policy, but they have been refined and some of

these enhancements will also be presented. Thereafter, we are going to present first pure network-calculus studies on Weighted Round Robin (WRR), on FIFO policy and thereafter their application to NoCs.

Feed-forward network analysis The SFA and TFA principles, presented in Section 4.2, have been defined in Schmitt and Zdarsky (2006). The Pay Burst Only Once principle, presented in Theorem 2, models the fact that the burst of the flow of interest influences the worst delay in only one server on the flow path. A similar phenomenon exists with the cross-traffic: the multiplexing between the flow of interest and each interfering flow occurs only once, and this phenomenon is called Pay Multiplexing Only Once (PMOO) Schmitt et al. (2008b). The analytical expression of this phenomenon is not based on the min-plus operators Bouillard et al. (2008), and computable expressions have been given only for systems with token-bucket arrival curves and rate-latency service curves and blind multiplexing Schmitt et al. (2008b,a). The performances of the different strategies may depend on the network characteristics Bondorf and Schmitt (2016), and it may even be the case that adding pessimism to the network of interest can lead to better (*i.e.* smaller) upper bounds Bisti et al. (2008); Bondorf (2017). A detailed list of the causes of the over-approximations of these approaches can be found in Bondorf et al. (2017). This article also shows that applying all possible strategies to a given network will lead to an intractable problem (almost 10^{20} equations for a network with 32 routers and 100 flows). Nevertheless, a set of generic strategies can be found in Bondorf and Schmitt (2015); Bondorf et al. (2017). More recently, machine learning has been used to guide the network analysis Geyer and Bondorf (2019).

All previously presented approaches are called "algebraic approaches", since they mainly stay in the algebraic framework of network calculus presented in Section 3: computing residual service curves, arrival curves and using horizontal deviation to get delay bounds.

A different approach has been developed in Bouillard et al. (2010): considering blind multiplexing, instead of locally computing a residual service, the main equations of network calculus are instantiated on a set of well chosen instants. If arrival curves (resp. service curves) are piece-wise linear concave (resp. convex), this set of equations can be encoded as a Mixed-Integer Linear Problem (MILP). The maximal solution of this problem is then the exact worst-case delay. But solving the problem can be very costly,

Weighted round robin A network-calculus model of the WRR policy has been presented in Georges et al. (2011), Georges et al. (2005), without any proof and implicitly considering that all packets have the same size. It gives, for each class, a residual service. The same assumptions are done in Long et al. (2014), where a residual service is also given. Theses works have been generalized in (Bouillard et al., 2018, Thm. 8.6) considering an upper and lower bound on packet size for each flow. This last result is the one presented as Theorem 5 in Section 3.

One may also analyze a WRR arbiter using the "blind multiplexing" (cf. Theorem 3), since a WRR arbiter is also a work-conserving arbiter. One difference between both is that the WRR residual service offered to one queue depends only on the weights and the packet sizes, but is independent from the traffic of the flows using the others queues, whereas the blind multiplexing result does not consider the weights, only the maximal packet size and the flow traffics.

FIFO Both theorems on WRR transform the network into another one using only FIFO policy. They have been several works done on FIFO policy in the network calculus domain. The simplest approach, used for example in Frances et al. (2006),Boyer et al. (2011a), computes the end-to-end delay of a flow by doing the sum of the local delays. But, as recalled in Theorem 2, network calculus allows to compute smaller end-to-end bounds, using the *Pay burst only once* principle. Nevertheless, in the case of the FIFO policy, the application of this principle requires the choice of some real parameter $\theta \geqslant 0$ (cf. Theorem 4) per crossed server. In case of token-bucket arrival curves and rate-latency service curves, an explicit value of this parameter is given in Fidler (2003), based on the latency of the server, the burst of the interfering flows and the rates of the servers shared by the flow of interest and the interfering flows. The same problem has been addressed in Lenzini et al. (2004) and refined in Lenzini et al. (2005), Lenzini et al. (2007), leading to the DEBORAH tool, handling tandem networks, presented in Bisti et al. (2010), and downloadable at Bisti et al. (2011). Since these works only considers token-bucket flows and latency-rate servers, some others works have been done on more general classes of curves in Cholvi et al. (2002), Boyer and Fraboul (2008). Surprisingly, all these works but Fidler (2003) compute either optimal delay or arrival curve, without any explicit expression of the θ parameters.

The solution based on MILP developed for blind multiplexing has been adapted to FIFO multiplexing, in Bouillard and Stea (2014) and since the computation complexity was high, is has been enhanced to compute a simplified problem, that computes only an upper bound on delay. A free implementation for tandem networks, NetCalBound, is provided at Bouillard (2017), and experimental comparisons with DEBORAH are can be found in Bouillard and Stea (2012), Bouillard and Stea (2014). Following Bondorf et al. (2017), computing worst bounds with this method will be called "LP" and computing upper bounds will be called "ULP".

Network on Chip Considering the studies on NoC using network calculus, one may first cite Qian et al. (2009a), where the authors assume a NoC with FIFO policy and infinite buffers. The paper is mainly an adaption of Lenzini et al. (2005) to the NoC context.

The same authors address a realistic configuration in Qian et al. (2009b): each router has only one queue per input port (input queuing), the switching fabric uses a per-flit weighted round-robin to serve this input queues, and

wormhole switching is used to avoid buffer overflow. The network-calculus model takes into account the limited sizes of the queues and the use of the back-pressure mechanism. The back-pressure mechanism is also modeled in Zhan et al. (2013), but the authors seem not aware of the previous work of Qian et al. (2009b) and the equation (5) in Zhan et al. (2013) are different than the equations (4.1) and (4.2) in Qian et al. (2009b).

The back-pressure mechanism is also modeled in Giroudot and Mifdaoui (2020). It first computes the latency introduced by wormhole back-pressure (called "blocking latency"), and then build a service curve $\beta_{R,T}$ where the T is the back-pressure latency.

Weighted round-robin policy is also assumed in Jafari et al. (2010). It considers a NoC where in each port, the number of virtual channels is not less than the number of flows, and that VCs are served with a per-packet round-robin policy. It also assumes that the flows are regulated at the source by a token-bucket shaper. Then, it optimizes the token-bucket parameters in order to minimize the buffer use while "satisfying acceptable communication performances".

This model (round-robin arbitration and token-bucket shaping at the source) is quite close to the MPPA NoC architecture, but the MPPA NoC applies round-robin arbitration per queue, not per flow.

The Kalray MPPA is explicitly targeted in Giannopoulou et al. (2016), avoiding back-pressure by adequate traffic limiter configuration, but per flow round-robin is assumed.

In Dupont de Dinechin et al. (2014), a first network calculus model of the Kalray MPPA model was presented, assuming constant packet size.

Last, computing routing and resource allocation under delay constraint have been also studied in Frangioni et al. (2014), Frangioni et al. (2017)

5 Notations on topology

Before presenting the different methods used in this study to compute upper bounds for flows on the MPPA NoC, let us introduce some notations shared by all these methods.

These notations will be illustrated on a small example. In Figure 11, a flow f_1 goes from N1 to N3, crossing routers R1, R2, R3; another flow f_2 goes from N2 to N3, crossing routers R2, R3. In router R1, the flow f_1 is set in the queue "From Local" of the output port "To West". In router R2, it is set into the queue "From East" of the output port "To West". And in router R3, it uses the queue "From East" of the output port "To Local".

A hierarchical model would define routers, with ports and queues as attributes of a router. Our network calculus model considers a flat set of all ports in the NoC, $\{p^1, \ldots, p^{n_P}\}$, and also a flat set of all queues $\{q^1, \ldots, q^{n_Q}\}$. Figure 12 reports a subset of the queues involved in example of Figure 11: only queues "From Local" and "From West" have been drawn, and only the used ports. For example, the output port "To East" of the router R1 is p^1, and its queue "From Local" is q^1.

$\{q^1, \ldots, q^{n_q}\}$	set of queues
$\{p^1, \ldots, p^{n_p}\}$	set of ports
$p(q^i) = p^k$	q^i is an input queue of p^j
$\{f_1, \ldots, f_{n_f}\}$	set of flows
$l_i^{\min}, l_i^{\max}$	minimal and maximal packet size of f_i
$q^j \xrightarrow{f_i} q^k$	f^i goes from q^j to q^k
Q_i	route of flow f_i, as a sequence of queues
F^j	set of flows crossing q^j
A_i^j	cumulative function of f_i entering q^j
D_i^j	cumulative function of f_i leaving $p(q^j)$
α_i^j	arrival curve of A_i^j
$\dot{\alpha}_i^j$	arrival curve of D_i^j

Table 1: Notations related to topology.

The relation between queues and ports is done by a function p such that $p(q^i) = p^k$ if q^i is an input queue of the port p^j. In the example, $p(q^1) = p(q^2) = p^1$, $p(q^3) = p(q^4) = p^2$, etc.

The set of flow is $\{f_1, \ldots, f_{n_f}\}$. A flow has a determined path between one source and one destination[4], $l_i^{\min}$ (resp. $l_i^{\max}$) denotes the minimal (resp. maximal) size of a packet of flow f_i. The route of a flow is denoted queue per queue: $q^j \xrightarrow{f_i} q^k$ if the flow f^i goes from the queue q^j to the queue q^k.

For a flow f_i, Q_i is the (ordered) sequence of queues it crosses, *i.e.* since the flow f_1 follows the path $q^1 \xrightarrow{f_1} q^4 \xrightarrow{f_1} q^6$, then $Q_1 = q^1 q^4 q^6$.

For a queue q^j, F^j denotes the set of flows crossing this queue. Of course, if a queue q^j is in the path of flow f_i, then f_i is in the set of flows crossing this queue, *i.e.* $q^j \in Q_i \iff f_i \in F^j$. In the example, $F^1 = F^4 = \{f_1\}$, $F^2 = F^5 = \varnothing$, $F^3 = \{f_2\}$, and $F^6 = \{f_1, f_2\}$.

The cumulative function of the flow f_i entering the queue q^j is denoted A_i^j. The cumulative function leaving the output port $p(q^j)$ is denoted D_i^j.

For a given method[5], α_i^j (resp. $\dot{\alpha}_i^j$) denotes the arrival curve of the cumulative function A_i^j (resp. D_i^j). Of course, $q^j \xrightarrow{f_i} q^k$ implies $D_i^j = A_i^k$ and $\dot{\alpha}_i^j = \alpha_i^k$.

6 Explicit linear method for the MPPA NoC

The delay experienced by a flow crossing a NoC depends on the capacity of network elements, on the route from the source to the destination, and on the characteristics of the flows sharing some buffer or links along this route. We assume that each flow has been assigned a path and a maximum rate that

[4]The MPPA NoC has multicast capabilities, not considered here to keep notations simple.

[5]Different methods may compute different arrival curve for the same cumulative function.

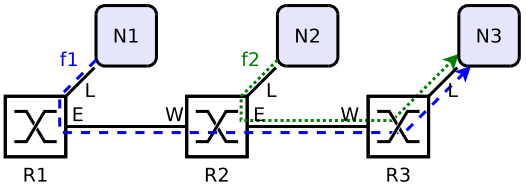

Figure 11: Network elements and flows example to illustrate notations.

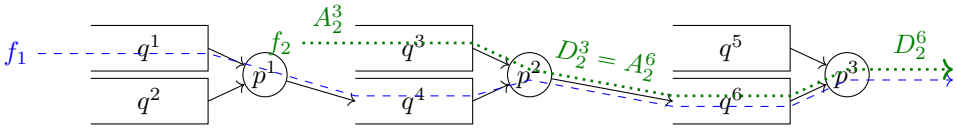

Figure 12: Partial translation of example of Figure 11.

ensures no link capacity is exceeded. This global network optimization problem can be solved using the max-min fairness criterion, for instance by using one of the methods described in Dupont de Dinechin et al. (2014).

Once flow paths and their maximum rate is known, the problem of ensuring that no back-pressure mechanism is active and expressing bounds on the flow end-to-end delay can be formulated as a Mixed-Integer Linear Problem (MILP). Indeed, by assuming that the flow rates are constant (set in previous step) it is possible to express the backlogs as variables of a linear problem. A solver is then used to maximize the flows bursts while keeping backlog in each queue not greater that the queue size.[6]. Since backlog and delays are closely related, solving this problem also computes upper bounds on delays.

This section will present only the part related to delays, and the reader may refer to Dupont de Dinechin and Graillat (2017) for details on backlog, routing and fairness.

This method is called "explicit" since the network calculus results presented in Section 3, involving specific operators (deviations, convolutions, etc.) are particularized in the specific case of affine arrival and service curves, and explicit analytic expressions are derived.

In this linear formulation, the arrival curve associated to each flow f_i at the input of a queue $q^j \in Q_i$ is a token-bucket $\alpha_i^j = \gamma_{r_i,b_i^j}$, where r_i is its rate (constant along the path) and b_i^j its burstiness in front of queue q^j.

[6]The use of MILP in this explicit linear formulation must not be confused with the use of MILP in LP method. In the explicit linear formulation, the variables of the mixed-integer linear problem are the burst sizes of the arrival curves at queue input, and routing-related values, as presented in Dupont de Dinechin and Graillat (2017), whereas the LP approach assumes that the flow parameters are fixed and the MILP variables are specific instants and the values of the cumulative functions at these instants, cf. Bouillard and Stea (2014).

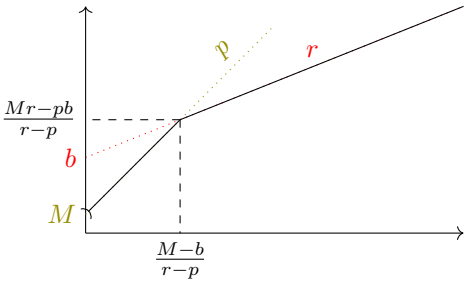

Figure 13: T-SPEC flow characteristic curve.

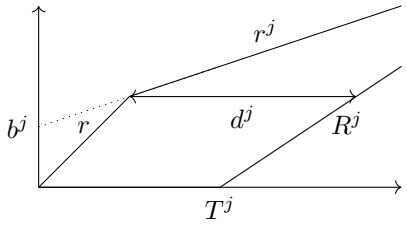

Figure 14: Delay between shaped token-bucket and rate-latency service: $d^j = hDev(\lambda_r \wedge \gamma_{r^j,b^j}, \beta_{R^j,T^j})$.

6.1 Arrival curve at queue input, and shaping of incoming link

Queue q^j receives the aggregates of flows F^j passing through it, so its arrival curve is of leaky-bucket type γ_{r^j,b^j} with

$$r^j = \sum_{f_i \in F^j} r_i, \qquad b^j = \sum_{f_i \in F^j} b_i^j. \tag{18}$$

But this aggregate flow comes from a link of peak rate r. Then, it also has λ_r as arrival curve. Combining both, it yields the arrival curve $\lambda_r \wedge \gamma_{r^j,b^j} : t \mapsto \min(rt, b^j + r^j t)$, which is a special case of the standard T-SPEC arrival curve $\alpha(t) = \min(M + pt, rt + b)\mathbb{1}_{\{t>0\}}$ used in IntServ Firoiu et al. (2002). Note the intersection of the two lines $pt + M$ and $rt + b$ has coordinate $(\frac{M-b}{r-p}, \frac{Mr-pb}{r-p})$ and that $\alpha(t) = M + pt$ if $t \in \left(0, \frac{M-b}{r-p}\right]$ and $\alpha(t) = rt + b$ if $t \geqslant \frac{M-b}{r-p}$ (cf. Figure 13).

Assume that this queue q^j receives from the link arbiter a rate-latency service curve β_{R^j,T^j} (the computation of these parameters R^j, T^j will be done in Section 6.3) with $R^j \leqslant r$ and $R^j \geqslant r^j$. The bound on delay for queue q^j is the maximum horizontal deviation between the arrival curve and the service curve $d^j \overset{def}{=} hDev(\lambda_r \wedge \gamma_{r,b^j}, \beta_{T^j,R^j})$. As illustrated in Figure 14, application of the T-SPEC arrival curve on such service curve yields

$$d^j = T^j + \frac{b^j(r - R^j)}{R^j(r - r^j)}. \tag{19}$$

6.2 Flow arrival curve

At ingress, whole packets are atomically injected at rate r. Call u the date when injection ends. We have $ru = l_i^{\max}$ and $l_i^{\max} \leqslant b_i + r_i u$, so

$$\forall f_i \in F : b_i \geqslant b_i^{\min} \overset{def}{=} l_i^{\max} \frac{r - r_i}{r}. \tag{20}$$

We now express the values r_i^j and b_i^j for all flows $f_i \in F^j$ for a queue q^j. If q^j is the first queue traversed by the flow, then $b_i^j = b_i$. Else, let q^k be predecessor of q^j in the sequence of active queues traversed by flow f_i (*i.e.* $q^k \xrightarrow{f_i} q^j$), with β_{R^k,T^k} its (residual) service curve. When flow f_i traverses queue q^k, its burstiness increases differently whether it is alone or aggregated with other flows in q^k.

If the flow is alone in queue q^k, we apply the classic result of the effects of a rate-latency service curve $\beta_{R,T}$ on a flow constrained by an affine arrival curve $\gamma_{r,b}$. The result is another affine arrival curve $\gamma_{r,b+rT}$ Le Boudec and Thiran (2001), so

$$b_i^j = b_i^k + r_i T^k. \tag{21}$$

Else, we apply Prop. 1. Let us introduce $r_{\neq i}^j = r^j - r_i$, $b_{\neq i}^j = b^j - b_i^j$, *i.e.*

$$r_{\neq i}^j = \sum_{f_l \in F^j, l \neq i} r_l, \qquad\qquad b_{\neq i}^j = \sum_{f_l \in F^j, l \neq i} b_l^j. \tag{22}$$

The competing flows have arrival curve $\alpha_{\neq i}(t) = \min(rt, r_{\neq i}^j t + b_{\neq i}^j)\mathbb{1}_{\{t>0\}}$ (the rt term comes from link shaping at q^k ingress). Since this function is sub-additive and $r_i + r_{\neq i} = \sum_{l \in F^i} r_l < R$, the proposition can be applied.

The $\alpha_{\neq i}$ function is a T-SPEC function, which is equal to the first term if $u \leqslant \frac{b_{\neq i}}{r - r_{\neq i}^j}$ and to the second otherwise. Then

$$\sup_{u \geqslant 0} \alpha_2(u) + r_i u - R^j \tag{23}$$

$$= \left(\sup_{0 \leqslant u \leqslant \frac{b_{\neq i}}{r - r_{\neq i}^j}} (r + r_i - R^j)u \right) \vee \left(\sup_{u \geqslant \frac{b_{\neq i}}{r - r_{\neq i}^j}} (r_{\neq i}^j + r_i - R)u + b_{\neq i} \right) \tag{24}$$

$$= b_{\neq i} \frac{r + r_i - R^j}{r - r_{\neq i}^j}. \tag{25}$$

Application of Prop. 1 leads to

$$b_i^j = b_i^k + r_i \left(T^j + \frac{b_{\neq i}^j (r + r_i - R^j)}{R^j (r - r_{\neq i}^j)} \right). \tag{26}$$

Note that the use of Cor. 1 would lead to $b_i^k + r_i \left(T^j + \frac{b_{\neq i}^j}{R^j} \right)$ that does not capture the benefit of the shaping r at input.

6.3 Link Arbiter Service Curves

On the MPPA2 NoC, the output link arbiters operate in round-robin per input queues at the packet granularity, while each queue contains flows aggregated

in FIFO. As the packets presented to a link arbiter are not processed in FIFO order, previous work (e.g. Bouillard and Stea (2015)) would have to assume blind multiplexing between all flows and fail to exploit FIFO aggregation. This is addressed in Dupont de Dinechin and Graillat (2017) by exposing the service offered to each queue of a link arbiter: either, the rate and latency ensured by round-robin packet scheduling; or, the residual service guaranteed by blind multiplexing across queues when the round-robin service does not apply. Then, each queue can be seen as a server applying a FIFO policy.

The service curve offered by a link arbiter to each of its queues is abstracted as a rate-latency function $\beta^j = \beta_{R^j,T^j}$. The first approach to derive this curve is to consider the behavior of the round-robin arbiter, assuming that each flow f_i has its packet sizes bounded by a minimum $l_i^{\min}$ and a maximum $l_i^{\max}$. Let $l_{F^j}^{\min} \stackrel{def}{=} \min_{f_i \in F^j} l_i^{\min}$ and $l_{F^j}^{\max} \stackrel{def}{=} \max_{f_i \in F^j} l_i^{\max}$ be respectively the minimum and maximum packet sizes for q^j (with convention that $\max \varnothing = 0$ to encode the fact that a queue crossed by no flow has no influence on the round robin arbiter). Let $Q^{\neq j} \stackrel{def}{=} \left\{ q^k \mid p(q^k) = p(q^j), k \neq j \right\}$ be the set of queues sharing the same arbiter than q^j. By application of eq. (17), the general round-robin service curve $\beta^j = \beta_{R^j,T^j}$ for q^j is

$$R^j = \frac{r l_{F^j}^{\min}}{l_{F^j}^{\min} + \sum_{k \in Q^{\neq j}} l_{F^k}^{\max}}, \qquad T^j = \frac{\sum_{k \in Q^{\neq j}} l_{F^k}^{\max}}{r}. \tag{27}$$

The second approach to derive a service curve for queue q^j is to consider that the round-robin arbiter serves packets at peak rate r according to a blind multiplexing strategy across the queues. Application of Theorem 3 yields the blind multiplexing service curve $\beta^j = \beta_{R^j,T^j}$ for q^j

$$R^j = r - \sum_{k \in Q^{\neq j}} r^k, \qquad T^j = \frac{\sum_{k \in Q^{\neq j}} b^k}{r - \sum_{k \in Q^{\neq j}} r^k}. \tag{28}$$

The blind multiplexing service curve must be used whenever the sum of flow rates inside q^j exceeds R^j in Eq. (27). Else, we select the formula that evaluates to the lowest T^j.

6.4 End-to-End Latency Bound

For computing an upper bound on the end-to-end delay of any particular flow f_i, we proceed in three steps. First, compute the residual (or left-over) service curve β_i^j of each active queue q^j traversed by f_i. Second, find the equivalent service curve β_i^* offered by the NoC to flow f_i through the convolution of the left-over service curves β_i^j. Last, find the end-to-end delay bound by computing d_i^* the delay between α_i the arrival curve of flow f_i and β_i^*. Adding d_i^* to the constant delays of flow f_i such as the traversal of non-active queues and other logic and wiring pipeline yields the upper bound.

This approach is an application of the SFA principle (cf. Section 3.5.2).

For the first step, we have two cases to consider at each queue q^j. Either f_i is the only flow traversing q^j, and $\beta_i^j = \beta_{R^j,T^j}$ from equations (27) or (28). Or, f_i is aggregated in q^j with other flows in F^j. Packets from the flow aggregate F^j are served in FIFO order, so we may apply Corollary 1. This yields the left-over service curve $\beta_i^j = \beta_{R_i^j,T_i^j}$ for an active queue q^j traversed by f_i:

$$R_i^j = R^j - r_{\neq i}^j, \qquad T_i^j = T^j + \frac{b_{\neq i}^j}{R^j}. \tag{29}$$

For the second step, we compute the convolution $\beta_i^* = *_{q^j \in Q_i} \beta_i^j$ of the left-over service curves β_i^j. Thanks to the properties of rate-latency curves Le Boudec and Thiran (2001), β_i^* is a rate-latency curve whose rate R_i^* is the minimum of the rates and the latency T_i^* is the sum of the latencies of the left-over service curves β_i^j:

$$R_i^* = \min_{j \in Q_i} R_i^j, \qquad T_i^* = \sum_{j \in Q_i} T_i^j. \tag{30}$$

For the last step, we compute the delay d_i^* between α_i the arrival curve of flow f_i at ingress and β_i^*. This flow is injected at rate r_i and burstiness b_i, however it is subject to link shaping at rate r as it enters the network. As a result, $\alpha_i = \min(rt, b_i + r_i t)1_{t>0}$ and we may apply Eq. (19):

$$d_i^* = T_i^* + \frac{b_i(r - R_i^*)}{R_i^*(r - r_i)}. \tag{31}$$

7 Adaptation of generic algorithms to the MPPA NoC

Section 6 has presented a modeling of the MPPA NoC using linear constraints for computing deadline while respecting buffer usage constraints (even if in this article, the focus is done only on the delay evaluation).

One may wonder if other algorithms may compute better bounds.

This section presents first how the Total Flow Analysis (TFA) and Separated Flow Analysis (SFA), initially defined for tandem topology with blind multiplexing, can be adapted to the case of the MPPA NoC, and especially to its hierarchical FIFO/RR scheduling (sections 7.1 and 7.2). Thereafter, is discusses how the specific case of constant packet size can help the analysis.

7.1 Total flow analysis

This section presents how the Total Flow Analysis (TFA), introduced in Section 3.5.2, is used and has been adapted to the specific case of the MPPA NoC.

The basic idea is of TFA is, given a queue q^j, to consider $A^j = \sum_{f_i \in F^j} A_i^j$ the *total* input flow, to compute α^j an arrival curve for A^j, and given β^j a service

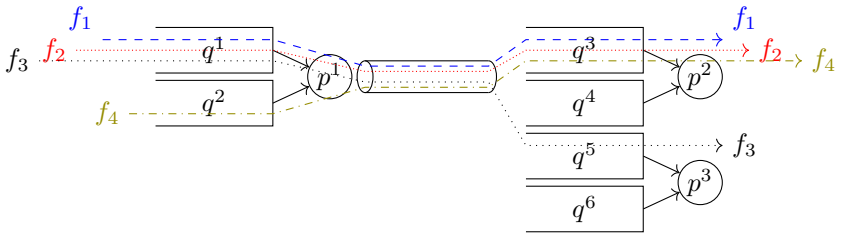

Figure 15: Illustration of TFA analysis.

curve of the queue, to compute $d^j = hDev(\alpha^j, \beta^j)$ a delay bound of the queue. Since the queue applies a FIFO policy between its flows, this delay bound is also a bound for each flow, and the end-to-end delay of a flow f_i can be bounded by the sum of the d^j of the crossed queues q^j: $d_i^{\text{TFA}} = \sum_{q^j \in Q_i} d^j$.

This algorithm requires computing α^j and β^j for each queue q^j.

The computation of α^j relies on the iterative transformation of arrival curve[7]. Let α_i^j be an arrival curve for the flow A_i^j. Then, the corresponding departure flow D_i^j has arrival curve $\dot{\alpha}_i^j = \left(\alpha_i^j \oslash \delta_{d^j}\right) \wedge \delta_0$ (cf. eq. (11) and eq. (13)).

Then, the computation of α^j relies on the identification of all queues q^k sending flits to the queue q^j. Let $I^j \stackrel{def}{=} \left\{ q^k \,\middle|\, \exists f_i : q^k \xrightarrow{f_i} q^j \right\}$ be this set.

Note that if a flow f_i goes from a queue q^k to a queue q^j, then $A_i^j = D_i^k$. Then α^j can be computed as the sum of all individual arrival curves $\dot{\alpha}_i^k$. But all these flows also pass through a link with peak rate r. This shaping implies that λ_r is another arrival curve for A^j, leading to

$$\alpha^j = \lambda_r \wedge \sum_{q^k \in I^j} \sum_{f_i \in F^k \cap F^j} \dot{\alpha}_i^k. \tag{32}$$

The computation of β^j can be done using either the residual service of the round-robin policy (Theorem 5), or the blind multiplexing (Theorem 3). The computation of the blind multiplexing requires computing the arrival curve of the competing flows[8]. It can be of interest when a queue shares the output link with lightly loaded queues. But the TFA algorithm is not forced to choose between both, it can compute both residual services, β^j_{Blind}, β^j_{RR} and then set $d^j = hDev(\alpha^j, \beta^j_{\text{Blind}}) \wedge hDev(\alpha^j, \beta^j_{\text{RR}})$.

For example, consider 4 flows f_1, f_2, f_3, f_4 as in Figure 15. The flows f_1, f_2, f_3 share a queue q^1, are scheduled at output port p^1, cross a link and enter the next switch where f_1 and f_2 are forwarded to q^3, whereas f_3 is forwarded to q^5. Another flow f_4 uses a queue q^2 then is scheduled by the same output port

[7] A discussion on how the original input curves are computed is postponed to Section 7.3.

[8] This can be done using eq. (32). If C^j is the set of queues sharing the same output port than q^j, $\alpha^{-j} = \sum_{k \in C^j} \alpha^k$ is an arrival curve for all the competing flows, and $\beta^j_{\text{Blind}} = \left[\beta - \alpha^{-j}\right]^+_\uparrow$ the blind residual service.

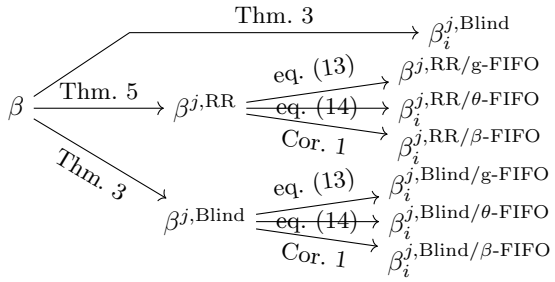

Figure 16: Different ways to compute a residual service.

p^1, crosses the same link and is forwarded to q^3. Assume that all flows f_1, f_2, f_3 comes from the same upstream queue q^0 and have respectively arrival curves $\alpha_1^1, \alpha_2^1, \alpha_3^1$. Then, $I^1 = \{q^0\}$, $I^3 = \{q^1, q^2\}$, $I^5 = \{q^1\}$, $F^1 = \{f_1, f_2, f_3\}$, $F^2 = \{f_4\}$, $F^3 = \{f_1, f_2, f_4\}$, $F^5 = \{f_3\}$. In this configuration, TFA computes $\alpha^1 = \lambda_r \wedge (\alpha_1^1 + \alpha_2^1 + \alpha_3^1)$ and two residual services β_{RR}^1, which is independent of the arrival curve of F_4, α_4^2, and $\beta_{\text{Blind}}^1 = [\lambda_1 - (\lambda_1 \wedge \alpha_4)]_\uparrow^+$. The local delay bound is then $d^1 = hDev(\alpha^1, \beta_{\text{RR}}^1) \wedge hDev(\alpha^1, \beta_{\text{Blind}}^1)$. Then, for each $i \in \{1, 2, 3\}$: $\dot{\alpha}_i^1 = \left(\alpha_i^1 \oslash \delta_{d^1}\right) \wedge \delta_0$. The same, a local delay d^2 is computed for q^2 and $\dot{\alpha}_4^2 = \left(\alpha_4^2 \oslash \delta_{d^2}\right) \wedge \delta_0$. For the queue q^3, its arrival curve is computed as $\alpha^3 = \lambda_r \wedge \left((\dot{\alpha}_1^1 + \dot{\alpha}_2^1) + (\dot{\alpha}_4^2)\right)$ since $I^3 = \{q^1, q^2\}$, $F^1 \cap F^3 = \{f_1, f_2\}$ and $F^2 \cap F^3 = \{f_4\}$.

7.2 Separated flow analysis

Whereas the Separated Flow Analysis (SFA) is well defined for a tandem network with blind multiplexing policy, its application to the NoC MPPA requires several adaptations, and some trade-offs, presented in this section.

The basic idea of SFA is, given a flow f_i to compute $\beta_i^{\text{SFA}} = *_{q^j \in Q_i} \beta_i^j$, where β_i^j is a residual service for the flow f_i in queue q^j. From a single flow point of view, the MPPA applies a hierarchical scheduling FIFO/RR: the bandwidth is shared between the queues using a RR scheduling and this left-over service is shared by the flows using a FIFO policy.

Then, one may consider several ways to compute the residual service β_i^j: either consider this hierarchical scheduling as a black box, and use the blind multiplexing result (Theorem 3), or first consider the residual service offered to the queue β^j (using either round robin residual service or blind multiplexing, as discussed in Section 7.1 on TFA) and secondly deduce the residual service left by the FIFO policy (using either eq. (14) or eq. (13) or the Cor. 1). Combining all possibilities leads to 7 different expressions, as presented in Figure 16. In fact, not all are of interest.

Considering only blind multiplexing ($\beta_i^{j,\text{Blind}}$) is always worse than modeling the RR arbiter per a blind policy and thereafter modeling the FIFO policy inside

the queue (residual services $\beta_i^{j,\text{Blind/*-FIFO}}$). The reason is that modeling a FIFO policy with blind multiplexing is pessimistic.

For the global delay (g-FIFO residual service), it would lead to the same result than TFA (presented in Section 7.1), and is not considered further.

Similarly, Corollary 1 can be applied only to affine modeling, and would lead to quite the same results than the explicit linear method (presented in Section 6) and is not considered further.

So, either $\beta_i^{j,\text{RR}/\theta\text{-FIFO}}$ or $\beta_i^{j,\text{Blind}/\theta\text{-FIFO}}$ has to be considered.

Note that every value of $\theta \in \mathbb{R}^+$ leads to a possible residual service, so each $\beta_i^{j,\text{RR}/\theta\text{-FIFO}}$ and $\beta_i^{j,\text{Blind}/\theta\text{-FIFO}}$ represents an infinite number of service curves. We postpone the discussion on the choice of θ and start by discussion the choice between $\beta_i^{j,\text{RR}/\theta\text{-FIFO}}$ and $\beta_i^{j,\text{Blind}/\theta\text{-FIFO}}$.

One may want to compute both $\beta^{j,\text{RR}}$ and $\beta^{j,\text{Blind}}$ and keep the maximum of both service curves. But it is not correct in general: it is true that if a server offers two minimal *strict* service of curves β, β', it offers a minimal *strict* service curve $\max\{\beta, \beta'\}$, but the results does not hold for minimal *simple* service, as illustrated at § 5.2.3 in Bouillard et al. (2018). One also may want to compute both for each server, and compute a residual service for all possible combination. But, for a path of length n, it will results in 2^n service curves. The strategy used in this paper consists in computing both $\beta^{j,\text{RR}}$ and $\beta^{j,\text{Blind}}$, and then to choose the one with the smaller TFA delay.

Let now discuss the choice of the θ parameter. The expression of the residual service is recalled here

$$\beta_i^{\theta\text{-FIFO}} = [\beta - \alpha_{\neq j} * \delta_\theta]^+ \wedge \delta_\theta, \tag{33}$$

with $\alpha_{\neq i} = \sum_{j \neq i} \alpha_j$.

The choice of θ is not straightforward. One may notice that setting $\theta = 0$ is equivalent to consider a blind multiplexing, *i.e.* the worst possible scheduling among all others for the flow of interest[9].

The choice of the parameter is a trade-off: let θ, θ' be two parameters, with $\theta < \theta'$, how to compare $\beta_i^{\theta\text{-FIFO}}$ and $\beta_i^{\theta'\text{-FIFO}}$? The convolution by a delay is just a time shift: for any $\theta \in \mathbb{R}^+$, $(f * \delta_\theta)(t) = f([t-\theta]^+)$. Then, on the one hand, $\theta < \theta'$ implies $\alpha_{\neq j} * \delta_\theta > \alpha_{\neq j} * \delta_{\theta'}$, *i.e.* a larger parameter decreases the impact of competing flows, leading to $\beta - \alpha_{\neq j} * \delta_\theta \leqslant \beta - \alpha_{\neq j} * \delta_{\theta'}$. On the other hand, $\theta < \theta' \implies \delta_\theta > \delta_{\theta'}$. Then, in general, $\beta_i^{\theta\text{-FIFO}}$ and $\beta_i^{\theta'\text{-FIFO}}$ are incomparable (cf. Figure 17).

One may nevertheless restrict the range of the parameter. First, notice that $\beta^{\theta\text{-FIFO}} \leqslant \delta_\theta$, then any θ greater than $hDev(\sum_{i=1}^n \alpha_i, \beta)$, will yield a residual service $\beta^{\theta\text{-FIFO}}$ smaller than the one obtained with the g-FIFO solution. So any $\theta > hDev(\sum_{i=1}^n \alpha_i, \beta)$ will give a worst delay than the TFA approach. Second, it is common to have a service curve that is equal to zero up to some value. Let $T_\beta = \inf\{t \mid \beta(t) > 0\}$ (for a rate-latency function $\beta_{R,T}$, this is the latency term, *i.e.* $T_{\beta_{R,T}} = T$). Then, for any $\theta < T_\beta$,

[9] To be exact, with $\theta = 0$, the θ-FIFO residual service can be worst than the blind multiplexing since there is no non-decreasing closure.

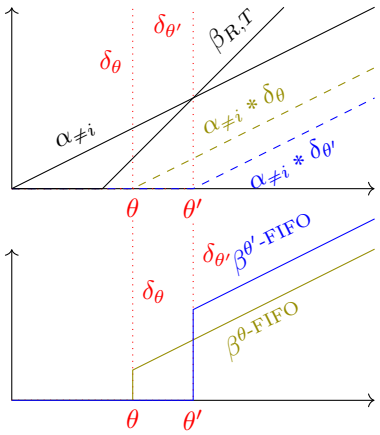

Figure 17: Residual FIFO service with to θ, θ' parameters, $\alpha_{\neq i} = \sum_{j \neq i} \alpha_j$.

$\beta \wedge \delta_\theta = \beta$, leading to $\beta^{\theta\text{-FIFO}} = [\beta - \alpha_{\neq j} * \delta_\theta]^+$. So, considering $\theta < \theta' < T_\beta$, $\beta^{\theta\text{-FIFO}} \leqslant \beta^{\theta'\text{-FIFO}}$, meaning that values of $\theta \in [0, T_\beta]$ have no interest. Then, only values $\theta \in [T_\beta, hDev(\sum_{i=1}^n \alpha_i, \beta)]$ are of interest.

Up to our knowledge, the only work on the choice of the θ parameter can be found in Fidler (2003). While building an end-to-end FIFO service in a tandem network with token-bucket arrival curves and rate-latency service curves it proposes such a value. Considering a flow of interest i and a node j, the proof proposes, to set the parameter as the sum of the latency of the server, plus, for each cross flow j crossing i for the first time in this server, the burst size of the flow j divided by the minimal rate of server on their common path. Using the notations defined in Section 5, considering that the common sub-path between two flows i and i' is $Q_i \cap Q_{i'}$, it gives

$$\theta_i^j = T^j + \sum_{i' \in \text{firstCross}(i,j)} \frac{b_{i'}^j}{\min_{q^{j'} \in Q_i \cap Q_{i'}} R^{j'}} \tag{34}$$

$$\text{firstCross}(i,j) \stackrel{def}{=} \left\{ i' \in F^j \mid Q_i \cap Q_{i'} = q^j \cdots \right\} \tag{35}$$

where $q^j \cdots$ is an abbreviation denoting any sub-path starting at q^j, $b_{i'}^j$ is the burst of the flow i' at q^j input, and $R^{j'}$ the rate of the residual service offered to the queue $q^{j'}$.

When modeling packets of constant size (cf. Section 7.3), we will consider arrival curves that are no token-bucket, where there is no b_j^i parameter, and conversely service curves with no R^j parameter. So, we will generalize this approach by taking $b_j^i = \lim_{\epsilon \to 0, \epsilon > 0} \alpha_j^i(0 - \epsilon)$ the right limit at origin of the arrival curve, and $R^j = \lim_{t \to \infty} \frac{\beta^j(t)}{t}$ the long term rate of the function.

Last, the SFA approach does not specify how are computed the arrival curves

of the competing flows: in each node, for any $j \neq i$, one may compute α_j using TFA, or considering a new SFA problem for this flow (up to this node), or compute both and take the minimum, etc. As shown in Bondorf and Schmitt (2015), computing better bounds for the cross traffic leads to better end-to-end bounds for the flow of interest. In order to ease comparisons, the arrival curves of the competing flows will be the one computed with the TFA approach.

7.3 Constant packet size

Both the TFA and SFA approaches, presented in the previous section, can be seen as black boxes transforming some input arrival and service curves into delay bounds. This section discusses these input curves.

The traffic limiters at the NoC ingress ensure that each flow respects a (configurable) token-bucket shape. Considering also the limited link throughput lead to a T-SPEC arrival curves, as presented in section 6.1 (cf. Figure 3). It belongs to the class of concave piecewise-linear function (CPL). Conversely, the residual service of a round robin arbiter given by eq. (17) is also a convex piecewise-linear function (CxPL). And the residual service of a blind multiplexing is also a CxPL function if the arrival curves are CPL and the aggregate service is CxPL. Using such concave/convex piecewise-linear functions in network calculus is called a *linear*, or *affine* or *fluid* model.

In Boyer et al. (2018), the explicit linear method and the TFA approach with affine curves have been compared on one example.

But such model cannot capture accurately the impact of packetization. Indeed, a flow is made of packets, and in the MPPA NoC (and in the absence of back-pressure activation), when a packet starts is emission, it is sent up to completion at link speed. Modeling this effect allows more accurate arrival and service curve, leading to better (*i.e.* smaller) bounds. This is true at arbiter output, and this behavior is captured by eq. (15). But this is also true at traffic limiters output and this is captured by Theorem 6 when all packets in a flow have the same size.

Indeed, the traffic limiter at ingress of the MPPA NoC, presented in Section 2, ensures by design that the output flow will respect a token-bucket arrival curve $\gamma_{r,b}$. But this limiter only injects full packets, *i.e.* the first flit of a packet is sent only if there will be enough credit to send the full packet without any interruption. When a data flow always sends packets of the same size, it means that not all values of the arrival token-bucket arrival curves can be reached by an actual sequence of packets.

Theorem 6. *Consider a data flow A made only of packets of fixed length l, such that when a packet starts it emission, it is emitted up to completion at a constant rate R. Then if α is a maximal arrival curve for A, also is $\alpha' = l \left\lfloor \frac{\alpha}{l} \right\rfloor \oslash \lambda_R$.*

The cumulative function of such a flow is an alternation of flat segments (no output of data) and segments of slope R, height l and length $\frac{l}{R}$.

Note that this result can be applied for any arrival curve, whereas in the context of the MPPA NoC, it will be used only for functions of the form $\alpha =$

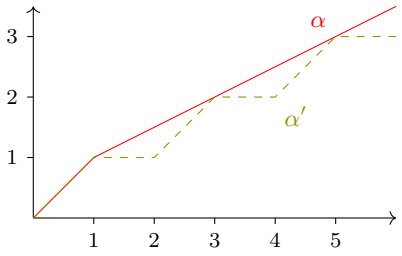

Figure 18: Effect of constant packet size on arrival curve, Thm. 6, with $\alpha = \gamma_{1/2,1/2} \wedge \lambda_1$, $l = 1$.

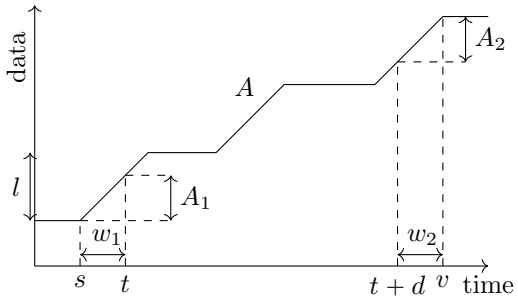

Figure 19: Per packet cumulative function.

$\lambda_R \wedge \gamma_{r,b}$ (as in Figure 18).

Proof. Let $t, d \in \mathbb{R}^+$ be a time instant and a duration, and consider the amount of data $A(t+d) - A(t)$.

Let us first assume that some packet is being sent at instants t and $t+d$.

Let s be the begin of the sending interval containing t, and v the end of the sending interval of $t+d$, as illustrated on Figure 19.

The main step of the proof consists in showing that $A(t+d) - A(t) \leqslant l \left\lfloor \frac{\alpha(d+w)}{l} \right\rfloor - Rw$ with $w = (t-s) + (v-(t+d))$.

Let $w_1 = t - s$, $w_2 = v - (t+d)$, $A_1 = A(t) - A(s)$ the amount of data sent on $[s,t]$, $A_2 = A(v) - A(t+d)$ the one on $[t+d, v]$. Consider the decomposition $A(v) - A(s) = A_1 + A(t+d) - A(t) + A_2$.

On intervals $[s,t]$ and $[t+d, v]$, some part of a packet is sent, as constant speed R, so $A_1 = Rw_1$ and $A_2 = Rw_2$, leading to $A(v) - A(s) = A(t+d) - A(t) + R(w_1 + w_2)$.

The flow admits α as arrival curve, so $A(v) - A(s) \leqslant \alpha(v-s)$. But by construction, they are $n \in \mathbb{N}$ full packets of size l sent on $[s,v]$, *i.e.* $A(v) - A(s) = nl$, so $n \leqslant \left\lfloor \frac{\alpha(v-s)}{l} \right\rfloor$ and

$$A(t+d) - A(t) + R(w_1 + w_2) \leqslant l \left\lfloor \frac{\alpha(v-s)}{l} \right\rfloor \tag{36}$$

Let $w = w_1 + w_2$, notice that $v - s$ can be written as $v - s = d + w$, it yields

$$A(t+d) - A(t) \leqslant l \left\lfloor \frac{\alpha(d-w)}{l} \right\rfloor - Rw \tag{37}$$

$$\leqslant \sup_{w \geqslant 0} l \left\lfloor \frac{\alpha(d-w)}{l} \right\rfloor - \lambda_R(w) \tag{38}$$

$$= \left(l \left\lfloor \frac{\alpha}{l} \right\rfloor \oslash \lambda_R \right)(d) \tag{39}$$

Approach \ Model	Fluid (unknown packet size)	Per flow constant packet size	Per flow and per queue constant packet size
End-to-End	Explicit Linear, LP, SFA/Aff	SFA/Fc	SFA/FQc
Local	TFA/Aff	TFA/Fc	TFA/FQc

Table 2: The analysis strategies.

If not packet is sent at time t, let t' be the next instant when some packet starts its emission (if t' does not exist, it means that $A(t+d) = A(t)$ and the result holds). Then $A(t) = A(t')$. Conversely, if no packet is sent at $t+d$, let $d' \leqslant d$ be such that $t+d'$ is the previous instant when some packet ends its emission. It holds $A(t+d') = A(t+d)$. Then, the previous result can be applied: $A(t+d') - A(t') \leqslant \left(l \left\lfloor \frac{\alpha}{l} \right\rfloor \oslash \lambda_R\right)(d')$. By definition of t' and d', $A(t+d) - A(t) = A(t+d') - A(t')$ and since $l \left\lfloor \frac{\alpha}{l} \right\rfloor \oslash \lambda_R$ is non decreasing, and $d' \leqslant d$

$$A(t+d) - A(t) \leqslant \left(l \left\lfloor \frac{\alpha}{l} \right\rfloor \oslash \lambda_R\right)(d). \tag{40}$$

□

8 Comparing strategies

Several approaches and models have been presented in the previous sections. They will be compared on two case studies, a small one, allowing a detailed interpretation of the results, and a large one giving more realistic results.

The approaches have been partitioned in two categories: a first one computing an end-to-end delay, and a second one computing per queue delays (local) and deriving the flow delay as the sum of local delays. Three kinds of models have been considered: either no information on the packet size is modeled ("fluid" model), or we assume that all packets in a given flow have the same size ("per flow constant packet sizes" model) or we also assume that all packets in a given queue have the same size ("per flow and per queue constant packet sizes" model).

The explicit linear method, presented in Section 6, is an end-to-end approach with an affine model. The SFA is the most known end-to-end approach, but in the specific case of concave/convex piecewise-linear arrival and service curves, it is outperformed by the NetCalBounds tool, a free implementation of the LP method developed in Bouillard and Stea (2014). This LP method gives the *exact* worst delay (also known as *tight*), but its computation is exponential in the length of the path. Nevertheless, since the paths on our case studies are not so long, it was possible to use it. However, the SFA/Aff approach is

considered in order to compare it with the other approaches. The modeling of per flow constant packet sizes (use of Theorem 6) lead to non concave arrival curves, and in this case, only SFA can provide an end-to-end delay computation (SFA/Fc). Moreover, when all packets in a queue have the same size, combining SFA and Theorem 5 leads to SFA/FQc.

Conversely, the computation of the flow delay as the sum of the local delays is done with TFA, with either an affine model (TFA/Aff), per flow constant packets sizes (TFA/Fc) and per flow and per queue constant packets sizes (TFA/FQc).

These different methods will be compared on two examples[10], a small one and a large one, with variation on the number of flows per queue and the maximal packet size.

The small example, from Dupont de Dinechin and Graillat (2017), has 4 nodes and 4 routers. A first experiment with this example is presented in Section 8.1 with only 4 flows in order to provide the details of the methods. A second experiment is presented in Section 8.2 with this example, splitting the flows, but keeping the same throughput, in order to illustrate how each method is influenced by the number of flows. And Section 8.3 uses the same example as in Section 8.1 but with larger packets in order to illustrate the influence of packet sizes.

The large example uses all the 32 nodes and routers. In a first experiment, presented in Section 8.4, each node generates 4 flows, leading to a total number of 128 flows. In Section 8.5, the number of flow is doubled, leading to a second experiment on the large example.

The results on the explicit linear method have been obtained using a tool developed by Kalray, presented in Dupont de Dinechin and Graillat (2017). The results on the LP method have been obtained using the NetCalBounds tool from Bouillard (2017), with the same tandem topology than the SFA. The results on the affine Total Flow Analysis have been obtained using the RTaW-Pegase tool, from RealTime-at-Work (2019). All other results have been obtained by a prototype plugin to the RTaW-Pegase tool.

All technological delays in routers (routing, switching, etc.) have been neglected since they are constant and have no influence on the arrival curves.

8.1 An illustrative example with 4 nodes (small example, first experiment)

This section gives details on the different methods using a small example. It comes from Dupont de Dinechin and Graillat (2017). It has 4 nodes, generating 4 flows crossing 4 routers, with routing depicted in Figure 20. All flows have a packet size of 17 flits (considered as typical), all flows have a long-term rate $\frac{1}{3}$

[10] A third example can be found in Boyer et al. (2019), including a comparison with the "recursive calculus" from Ayed et al. (2016).

Flow	f_1	f_2	f_3	f_4
Rate	$\frac{2}{3}$	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{3}$
Max. Packet Size.	17	17	17	17
Burst	$\frac{17}{3}$	$\frac{34}{3}$	$\frac{34}{3}$	$\frac{34}{3}$

Table 3: Flow parameters, small example (topology of Figure 20), first experiment (original values).

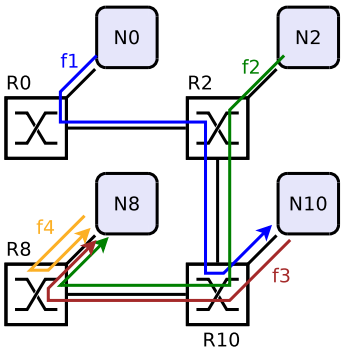

Figure 20: Case study from Dupont de Dinechin and Graillat (2017), 4 nodes.

but f_1 that have $r_1 = \frac{2}{3}$. The admissible bursts at network ingress are $\frac{34}{3}$ but f_1 that has $b_1 = \frac{17}{3}$ (cf. Table 3).

Before presenting how the different methods handle this example, let us come back on queue numbering. When presenting the different methods, for sake of simplicity, we have assumed that the set of ports and the set of queues are indexed by integers. In the small example, there are 4 routers, R0, R2, R10 and R8. Since there is only one active port per router[11], their respective active ports will be denotes p^0, p^2, p^{10}, p^8. Each port has at most two active queues. For example, the port p^2 has two active queues, the "From West", used by f_1, and the "From Local" used by f_2. In this Section, $q^{a,b}$ will denote the active queue in router a used by flow coming from router b (or $q^{a,a}$ for the "From Local" queue). Then, the two queues in R2 will be denoted $q^{2,0}$ (used by the flow f_1) and $q^{2,2}$ (used by the flow f_2). Then, the blind residual service of the queue $q^{2,0}$ will be denoted $\beta^{2,0}_{\text{Blind}}$, its total arrival curve will be denoted $\alpha^{2,0}$, etc.

8.1.1 Details of the explicit linear method

Consider first the flow f_1. It comes out from the traffic limiter with burst $b_1^0 = \frac{17}{3}$, and rate $r_i = \frac{2}{3}$, *i.e.* arrival curve $\lambda_1 \wedge \gamma_{\frac{2}{3},\frac{17}{3}}$ (cf. Figure 14). It path starts at the router R0, and it is the only flow using port p^0. Then, p^0 offers

[11] Except in R10, where both the "To Local" and "To West" are active, but only the "To West" is shared and requires a name.

to the flow f_1 its full capacity of one flit per cycle, λ_1, and by application of eq. (19), the local delay is null, $d^0 = 0$. The arrival curve is not modified, when entering the router R2, $b_1^2 = \frac{17}{3}$.

In R2, two flows f_1 and f_2 are sharing the output port p^2, f^1 (resp. f_2) using the queue $q^{2,0}$ (resp. $q^{2,2}$). Consider f_1: to compute the residual service offered by the link arbiter, one may either use the round-robin formula, eq. (27), leading to $R^2 = \frac{1}{2}$ (half of the bandwidth) and $T^2 = 17$ (one packet from competing flow f_2), or the blind one, eq. (28), leading to $R^2 = 1 - r^2 = \frac{2}{3}$ (since the competing flow f_2 has only a rate $r^2 = \frac{1}{3}$) and $T^2 = \frac{\frac{34}{3}}{R^2} = 17$. The output burst is then $b_1^{10} = 17$ (cf. eq. (21)). For f_2, since it is in competition with a flow of rate $\frac{2}{3}$, it is better to consider the round-robin residual service, which offers the curve $\beta_{\frac{1}{2},17}$, leading to the output burst $b_2^{10} = 17$.

Last, in R10, f_1 uses the port leading to the local cluster, and like in R0 suffers no delay. The end-to-end service offered to f_1 is then $\beta_1^* = \beta_{R_1^*,T_1^*}$, with R_1^* the minimum of all residual rates along the path $R_1^* = \min\left\{1, \frac{2}{3}, 1\right\} = \frac{2}{3}$, and T_1^* the sum of each residual latencies $T_1^* = 0 + 17 + 0 = 17$ (cf. eq. (30)).

The application of eq. (31) yields, for the flow f_1, an upper bound on delay $d_1^* = \frac{51}{2}$.

In R10, the flows f_2 and f_3 share an output port p^{10}, in two different queues, $q^{10,2}$ and $q^{10,10}$ respectively. This case being similar to the port p^2, details are not provided: f_2 (resp. f_3) receives a residual service $\beta_{\frac{2}{3},17}$ and its output burst is $b_2^8 = \frac{68}{3}$ (resp. $\beta_{\frac{2}{3},17}$, $b_3^8 = 17$).

The output port p^8 of the router R8 presents a more interesting situation, since two flows, f_2 and f_3 share a same queue, $q^{8,10}$, competing with the queue $q^{8,8}$ used by the flow f_4.

For the queue $q^{8,8}$ (and its only flow f_4), the round-robin and blind residual services are very different: the round-robin service offers a guaranteed rate $\frac{1}{2}$ and a latency of 17 corresponding to one single frame, whereas the blind multiplexing offers only $\frac{1}{3}$ as guaranteed rate, and a latency $\frac{b_2^8 + b_8^3}{\frac{1}{3}} = 119$ corresponding to the burst of the two flows f_2 and f_3, leading to an end-to-end-delay bound of $d_4^* = 34$.

For the queue $q^{8,10}$ shared by the flows f_2, f_3, the best residual service is the blind one, offering a service $\beta_{\frac{2}{3},17}$. Consider now f_2: it shares the service of the queue $q^{8,10}$ with f_3 with a FIFO policy. By application of eq. (29), it receives the rate-latency service of parameters $R_2^{8,10} = \frac{2}{3} - \frac{1}{3} = \frac{1}{3}$ and $T_2^{8,10} = 17 + \frac{17}{\frac{2}{3}} = 42.5$.

The end-to-end service offered to f_2 is then $\beta_2^* = \beta_{R_2^*,T_2^*}$, with R_2^* the minimum of all residual rates along the path $R_2^* = \min\left\{\frac{1}{2}, \frac{1}{2}, \frac{1}{3}, 1\right\} = \frac{1}{3}$, and T_2^* the sum of each residual latency. $T_2^* = 17 + 17 + 42.5 = 76.5$. This leads to a bound on its end-to-end delay $d_2^* = 110.5$.

The same way, the residual service offered to f_3 is $\beta_{\frac{1}{3},51}$, its end-to-end service $\beta_{\frac{1}{3},68}$, and its end-to-end delay bound $d_3^* = 102$.

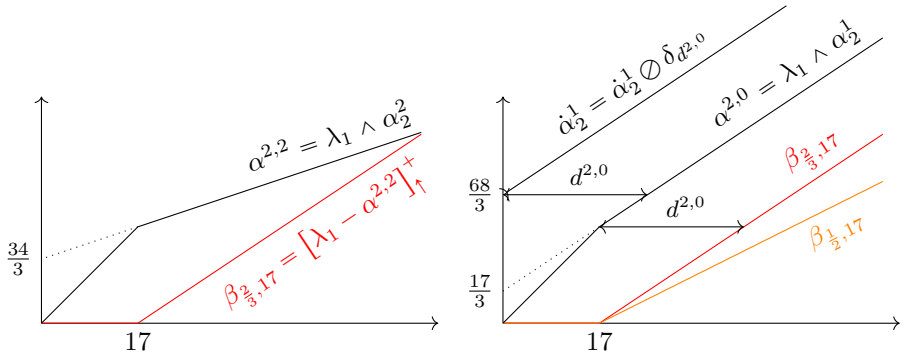

Figure 21: Functions related to the crossing of R2 per flow f_1, TFA method, fluid modeling. $d^{2,0} = hDev\left(\alpha_2^1, \beta_{\frac{2}{3},17}\right) = \frac{51}{2}$.

8.1.2 Details of the TFA/Affine method

Consider first the router R0: it is crossed by only one flow f_1, with arrival curve $\alpha_1^0 = \lambda_1 \wedge \gamma_{\frac{1}{3},\frac{17}{3}}$. Since this is the only flow, the arrival curve of the total input flow is $\alpha^{0,0} = \lambda_1 \wedge \alpha_1^0 = \lambda_1 \wedge \gamma_{\frac{1}{3},\frac{17}{3}}$. The port p^0 offers a service of minimal curve λ_1, and $d^1 = hDev(\alpha^{0,0}, \lambda_1) = 0$, the local delay is null. The arrival curve of f_1 at the output is equal to the one at the input: $\dot{\alpha}_1^0 = \alpha_1^0$.

In R2, they are two actives queues, $q^{2,0}$ and $q^{2,2}$, used each by only one flow, respectively f_1 and f_2, leading to the total arrival curves $\alpha^{2,0} = \lambda_1 \wedge \alpha_1^2$, $\alpha^{2,2} = \lambda_1 \wedge \alpha_2^2$. Queue $q^{2,0}$ receives a residual blind service $\beta^{2,0}_{\text{blind}} = \left[\lambda_1 - \alpha^{2,2}\right]^+_\uparrow = \beta_{\frac{2}{3},17}$ and a residual round-robin service $\beta^{2,0}_{\text{RR}} = \beta_{\frac{1}{2},17}$ (cf. Figure 21). The round-robin residual rate is not sufficient for the flow f_1: $hDev(\alpha^{2,0}, \beta^{2,0}_{\text{RR}}) = hDev(\lambda_1 \wedge \gamma_{\frac{2}{3},17}) = \infty$, but the blind one leads to the bound $hDev(\alpha^{2,0}, \beta^{2,0}_{\text{blind}}) = \frac{51}{2}$. The arrival curve of f_1 at R2 output is $\dot{\alpha}_1^2 = \left(\alpha_1^2 \oslash \delta_{\frac{51}{2}}\right) \wedge \delta_0 = \gamma_{\frac{1}{3},\frac{68}{3}}$.

The situation of f_2 in R2 is similar, except that in this case, the residual round-robin service is better than the residual blind. Its input arrival curve is $\alpha_2^2 = \lambda_1 \wedge \gamma_{\frac{1}{3},\frac{34}{3}}$, its local delay bound is 34, and its output arrival curves is $\dot{\alpha}_2^2 = \gamma_{\frac{1}{3},\frac{68}{3}}$.

In R10, the flow f_1 crosses the output port toward the local cluster: it is the only one using this port and then has a null delay. The flows f_2 and f_3 are competing for the output port p^{10}, respectively in queues $q^{10,2}$ and $q^{10,10}$. For the queue $q^{10,2}$ the round-robin policy offers a residual service curve $\beta_{\frac{1}{2},17}$ whereas the one offered by the blind multiplexing is $\beta_{\frac{2}{3},17}$. The minimal delay bound is then $51 \wedge 34 = 34$.

On the opposite, for the queue $q^{10,10}$, the smaller delay bound is the one computed by the round-robin residual service. Indeed, $\beta^{10,10}_{\text{RR}} = \beta_{\frac{1}{2},17}$, whereas the large burst of f_2 leads to a residual blind service $\beta^{10,10}_{\text{Blind}} = \beta_{\frac{2}{3},34}$. In this

case, both residual functions are incomparable.

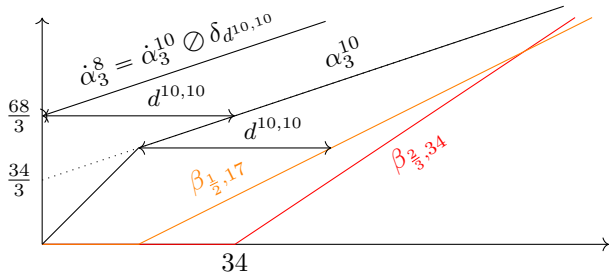

Figure 22: Functions related to the crossing of R10 per flow f_3, TFA method, fluid modeling. $d^{10,10} = hDev\left(\alpha^{10,10}, \beta_{\frac{1}{2},17}\right) = 34$.

In R8, the output port p^8 has two actives queues, $q^{8,8}$ used only by flow f_4 and $q^{8,10}$ used by flows f_2, f_3. The total arrival curve of queue $q^{8,10}$ is the sum of the arrival curves of flow f_2, f_3, shaped by the link capacity, *i.e.* $\alpha^{8,10} = \lambda_1 \wedge \left(\alpha_2^8 + \alpha_3^8\right)$. The two residual services computed for $q^{8,8}$ are $\beta_{\text{RR}}^{8,8} = \beta_{\frac{1}{2},17}$ and $\beta_{\text{Blind}}^{8,8} = \beta_{\frac{1}{3},170}$ leading to the delay bound 34. For the queue $q^{8,10}$, the two residual services are $\beta_{\text{RR}}^{8,10} = \beta_{\frac{1}{2},17}$ $\beta_{\text{Blind}}^{8,10} = \beta_{\frac{2}{3},17}$ leading to delay bound 102 as illustrated on Figure 23.

8.1.3 Details of the SFA/Affine method

The SFA method considers a flow of interest and extracts from the full topology a sub-path, a tandem topology: the sequence of queues crossed by the flow (and possibly other flows). In this study, for each queue, the residual service considered is the one computed by the TFA method. For each competing flow crossing this sub-path, the arrival curve at the entrance in the sub-path is also the one computed by the TFA method. Since all flows in this sub-path are served with a FIFO policy, the equations (14) and (34) will be used to extract the residual service for the flow of interest.

The flow f_1 crosses a sequence of 3 routers, R0, R2, R10, but is alone in the output port of R0 and R10. In both, it receives the full service λ_1. In R2, f_1 crosses the queue $q^{2,0}$, that offers the service $\beta_{\text{Blind}}^{2,0} = \beta_{\frac{2}{3},17}$. The end-to-end service is then $\beta_1^{\text{SFA}} = \lambda_1 * \beta_{\frac{2}{3},17} * \lambda_1 = \beta_{\frac{2}{3},17}$. The end-to-end delay bound is then $hDev\left(\lambda_1 \wedge \gamma_{\frac{2}{3},\frac{17}{3}}, \beta_{\frac{2}{3},17}\right) = \frac{51}{2}$.

The flow f_3 first crosses output the east output port of the router R10, p^{10}. They are two active queues, one coming from router R2, $q^{10,2}$, used by the flow f_2, and one coming from the local cluster, $q^{10,10}$.

Considering the residual service offered to $q^{10,10}$, and then to f_3 the only flow in this queue, one may either use the round-robin residual service (Thm. 5) or the blind one (Thm. 3). The round-robin residual service is $\beta_{1/2,17}$ corresponding to

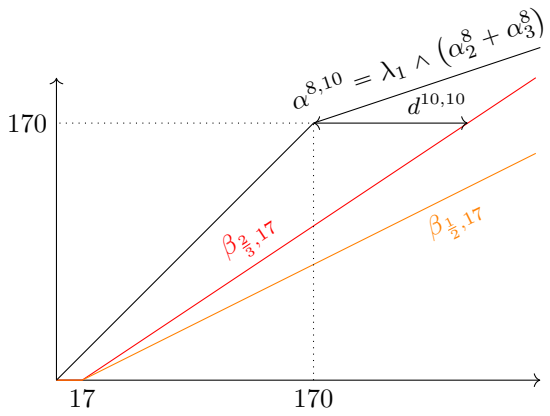

Figure 23: Functions related to the crossing of R8 per flows f_2 and f_3, TFA method, fluid modeling. $d^{8,10} = hDev\left(\alpha^{8,10}, \beta_{\frac{2}{3},17}\right) = 102$.

a latency of one frame (in a round-robin cycle) and half of the bandwidth (since they are two queues) whereas the blind one is $\beta_{2/3,34}$ since the competing flow f_2 has a rate of $1/3$ but a burstiness of two frames. Note that both are incomparable ($\forall x \leqslant 35, \beta_{2/3,34}(x) \leqslant \beta_{1/2,17}(x)$ and $\forall x > 35, \beta_{2/3,34}(x) > \beta_{1/2,17}(x)$). Since the round-robin policy gives a smaller bound on the delay, this is the one that is kept for the SFA end-to-end service, *i.e.* $\beta_3^{10} = \beta_{1/2,17}$.

The next router on its path is R8. The flow f_3 goes through the output port going to the local cluster, called p^8. It is set in the queue "from east", $q^{8,10}$, with the flow f_2. Since the queue $q^{8,10}$ is competing with only another queue, the round-robin residual service is $\beta_{1/2,17}$, whereas the blind is $\beta_{2/3,17}$. To derive the residual service offered to f_3 in $q^{8,10}$, one have to remove the part that is used by f_2. To do so, we use eq. (14) from Thm. 4, and chose a value for the θ parameter. The first term of eq. (34) is the latency of the service offered to the queue, 17. The second is a sum, looping on flows interfering for the first time. The set of interfering flow $F^{8,10} \backslash \{f_3\}$ is simply the singleton $\{f_2\}$. And f_2 is crossing f_3 for the first time in this server. Then the value of $\theta_2^{8,10}$ is $\theta_2^{8,10} = T^{8,10} + \frac{b_2^8}{R^{8,10}} = 68$ where $b_2^8 = 34$ is the burstiness of flow f_2 entering router R8, $R^{8,10}$ is the rate of the residual service offered to the queue $q^{8,10}$ and $T^{8,10} = 17$ is its latency.

This leads to the residual service β_3^8 presented in Figure 24.

The end-to-end service β_3^{SFA} is the convolution of the two residual services, and is plotted in Figure 25. The end-to-end delay of the flow f_3 is bounded by the horizontal deviation $hDev(\alpha_3^{10}, \beta_3^{\text{SFA}}) = 119$.

The details of the methods on flows f_2 and f_4 are not given.

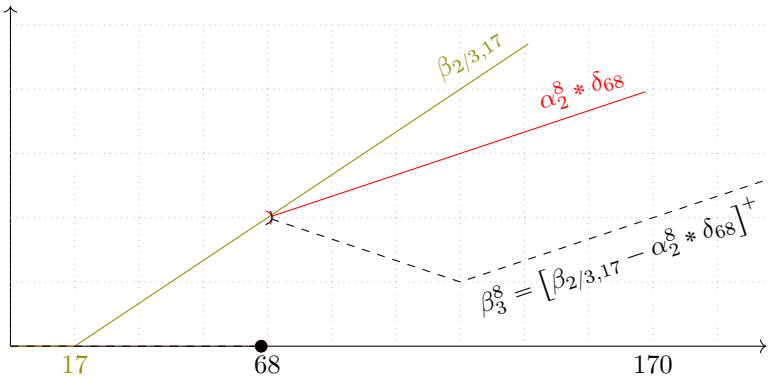

Figure 24: Functions related to the FIFO residual service for flow f_3 in router R8, SFA method, fluid modeling. The background dotted grid has step 17.

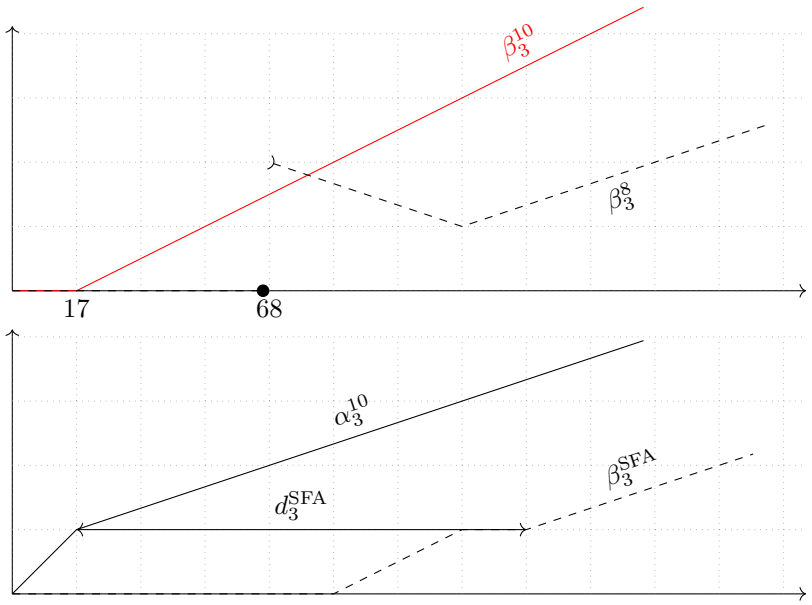

Figure 25: Local and end-to-end residual services functions for the flow f_3, SFA method, fluid modeling. The background dotted grid has step 17. $\beta_3^{\text{SFA}} = \beta_3^{10} * \beta_3^8, d_3^{\text{SFA}} = hDev\left(\alpha_3^{10}, \beta_3^{\text{SFA}}\right)$.

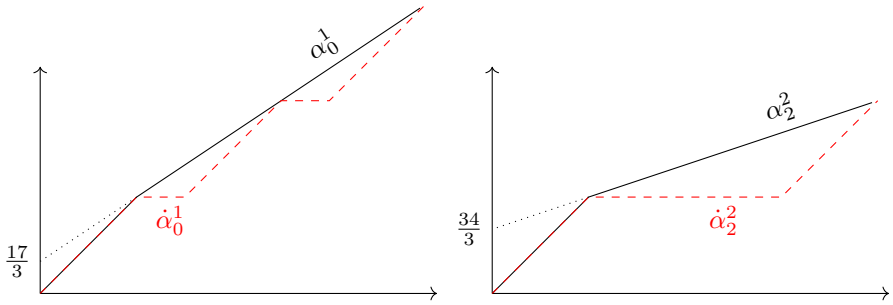

Figure 26: Arrival curves for flows f_1 and f_2: fluid and with constant packet sizes.

α_0^1: fluid arrival curve of f_1 at R0 ingress,
$\dot{\alpha}_0^1$: arrival curve of f_1 considering constant packet sizes at R0 ingress
α_2^2: fluid arrival curve of f_2 at R2 ingress,
$\dot{\alpha}_2^2$: arrival curve of f_2 considering constant packet sizes at R2 ingress

8.1.4 Illustration of TFA/Fc

We are going here to present the main differences between the "fluid" modeling (TFA/Aff), the "per flow constant packet sizes" (TFA/Fc), on some examples with the flows f_1 and f_3.

To distinguish TFA/Aff and TFA/Fc related variables, a dot is added on top of the TFA/Aff ones. Note that only the arrival curves, the blind residual curves and the delays are modified, the round-robin residual service is not modified, and then have no dot.

Consider f_1: considering that all its packets have the same size of 17 flits transforms (by application of Theorem 6) its fluid arrival curve α_0^1 at R0 ingress into $\dot{\alpha}_0^1$, as plotted in Figure 26. It is the only flow in router R0, and like in the fluid model (TFA/Aff), suffers no delay.

In the router R2, the flow f_1, in the queue $q^{2,0}$ is in competition with f_2, in the queue $q^{2,2}$. Like in the TFA/Aff case, the round-robin residual service is not sufficient (f_1 has a rate of $\frac{2}{3}$ whereas round-robin offers only $\frac{1}{2}$). But the arrival curve of the competing flow f_2 is now an alternation of slopes and plateaus, the shape of the blind residual service received by the queue $q^{2,0}$, $\dot{\beta}^{2,0}_{\text{Blind}} = \left[\lambda_1 - \left(\lambda_1 \wedge \dot{\alpha}_2^2\right)\right]^+_\uparrow$, is also such an alternation, as illustrated in Figure 27, leading to a delay bound equals to 17, whereas the fluid modeling (plotted with a dotted line) gives 21.5 as delay bound.

The delay bound computed for flow f_2 in R2 is the same in the TFA/Fc and TFA/Aff cases: even if the new arrival curve $\dot{\alpha}_2^2$ is smaller than α_2^2, the horizontal distance with the residual round-robin service is the same, as illustrated in Figure 28. Also note that the blind residual service is locally greater than the round-robin one, but this has no influence on the delay bound.

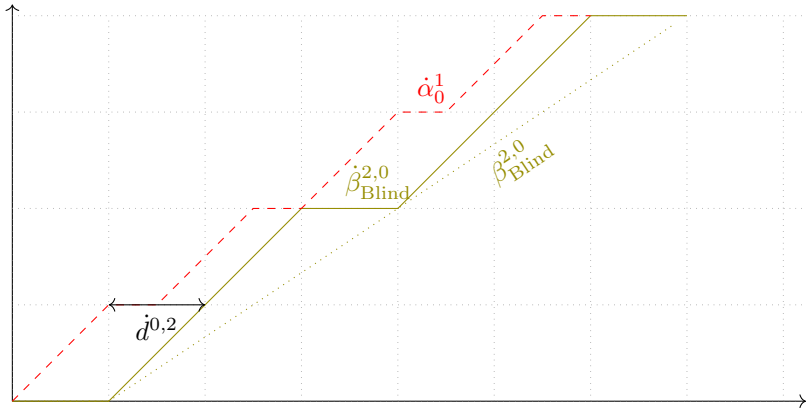

Figure 27: Functions related to the crossing of R2 per flow f_1, TFA method, modeling per flow constant packet size. The background dotted grid has step 17. $\dot{\alpha}^{2,0} = \dot{\alpha}_0^2, \dot{d}^{0,2} = hDev\left(\dot{\alpha}_0^2, \dot{\beta}^{0,2}_{\text{Blind}}\right)$.

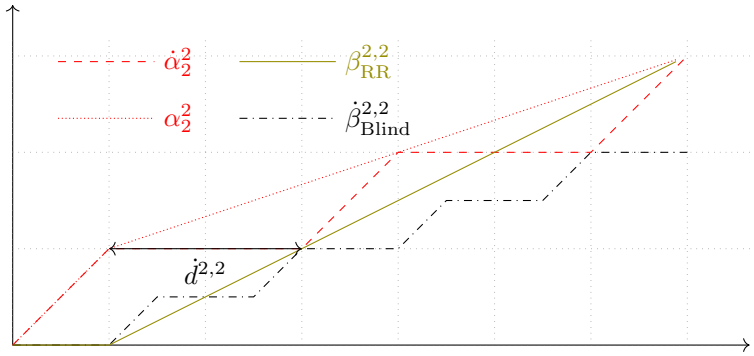

Figure 28: Functions related to the crossing of R2 per flow f_2, TFA method, modeling per flow constant packet size. The background dotted grid has step 17. $\dot{\alpha}^{2,2} = \dot{\alpha}_2^2, \dot{d}^{2,2} = hDev\left(\dot{\alpha}_2^2, \beta^{2,2}_{\text{RR}}\right)$.

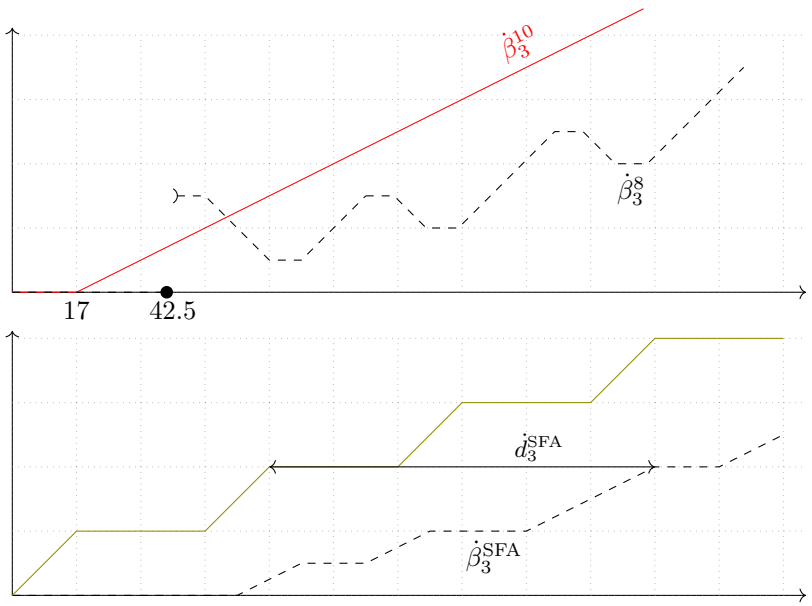

Figure 29: Local and end-to-end residual services functions for the flow f_3, SFA method, modeling per flow constant packet size. The background dotted grid has step 17. $\dot{\beta}_3^{\mathrm{SFA}} = \dot{\beta}_3^{10} * \dot{\beta}_3^{8}, \dot{d}_3^{\mathrm{SFA}} = hDev\left(\dot{\alpha}_3^{10}, \dot{\beta}_3^{\mathrm{SFA}}\right)$.

8.1.5 Illustration of SFA/Fc

Like TFA, the use of better arrival curves (and blind residual curves) allows SFA to compute better bound. Considering flow f_3, as plotted in Figure 29 (that can be compared with the SFA/Aff, plotted in Figure 25), a smaller interfering curve for f_4 in R8 leads to a larger residual service, with a smaller choice of the θ parameter, leading to $\dot{\beta}_3^8$. The end-to-end service is then $\dot{\beta}_3^{\mathrm{SFA}}$. It yields the delay bound $\dot{d}_3^{\mathrm{SFA}} = 102$.

8.1.6 Illustration of TFA/FQc

Considering that all packets in a queue have the same size allows to compute a better (*i.e.* greater) residual service for the round-robin policy. It has no impact on the residual blind multiplexing and so provides an enhancement only for flows using the round-robin residual service on their path. This is the case for the flow f_2 (in R2, where it competes with f_1 of rate $\frac{2}{3}$), and for flow f_3 (in R10, where the large burst of f_2 makes the round-robin residual service better than the blind one).

To ease comparison with the TFA/Fc case, the flow f_2 is considered.

To distinguish TFA/FQc related variables, a double dot is added on top of new residual round-robin service and on top of the associated delay.

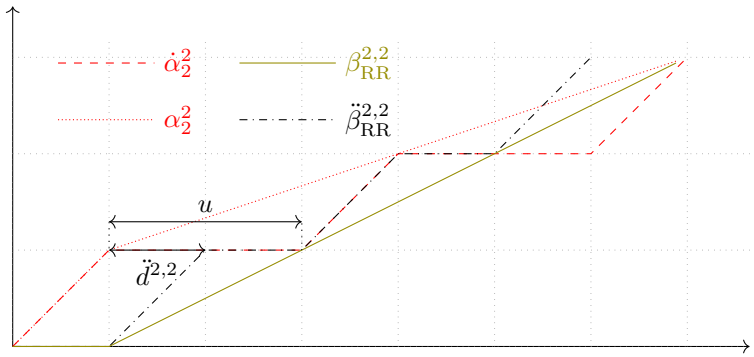

Figure 30: Functions related to the crossing of R2 per flow f_2, TFA method, modeling per flow and per queue constant packet size. The background dotted grid has step 17. $\dot{\alpha}^{2,2} = \dot{\alpha}_2^2, \ddot{d}^{2,2} = hDev\left(\dot{\alpha}_2^2, \ddot{\beta}_{\mathrm{RR}}^{2,2}\right), u = hDev\left(\alpha_2^2, \ddot{\beta}_{\mathrm{RR}}^{2,2}\right)$.

The new residual round-robin service curve $\ddot{\beta}_{\mathrm{RR}}^{2,2}$ is plotted with the previous one $\beta_{\mathrm{RR}}^{2,2}$ on Figure 30. Since it encodes the fact that the frame is served at full link speed once it is served, the curve is an alternation of slopes and plateaus, with the same long term rate. It yields $\ddot{d}^{2,2} = hDev\left(\dot{\alpha}_2^2, \ddot{\beta}_{\mathrm{RR}}^{2,2}\right) = 17$, whereas $\dot{d}^{2,2} = hDev\left(\dot{\alpha}_2^2, \beta_{\mathrm{RR}}^{2,2}\right) = 34$ (cf. Figure 28).

Also notice that the modeling of packet size must be done both for the arrival curve and the service curve to get the best bound, since $hDev\left(\alpha^{2,2}, \ddot{\beta}_{\mathrm{RR}}^{2,2}\right) = 34$.

8.1.7 Illustration of SFA/FQc

Like for TFA, modeling the fact that all packets in a queue have a constant size allows to get a better round-robin residual service and then a better end-to-end service, as illustrated in Figure 31 (to be compared with Figure 29), leading to a delay bound $\ddot{d}_3^{\mathrm{SFA}} = 85$.

8.1.8 Results sum up

The upper bounds on delays for this example are displayed in Figure 32. Even this simple example shows interesting trends that will be mainly confirmed by the other experiments.

First, the explicit linear method, which has been designed to compute also routing and allocate burst and throughput budget, gives good results w.r.t. other methods.

Second, the SFA method gives better results than the TFA one: it benefits from the Pay Burst Only Once effect, whereas TFA pays the burst in each server. Note nevertheless that the choice of the θ pays a major role in the accuracy of the result: the same experiment has been done in a previous study Boyer et al.

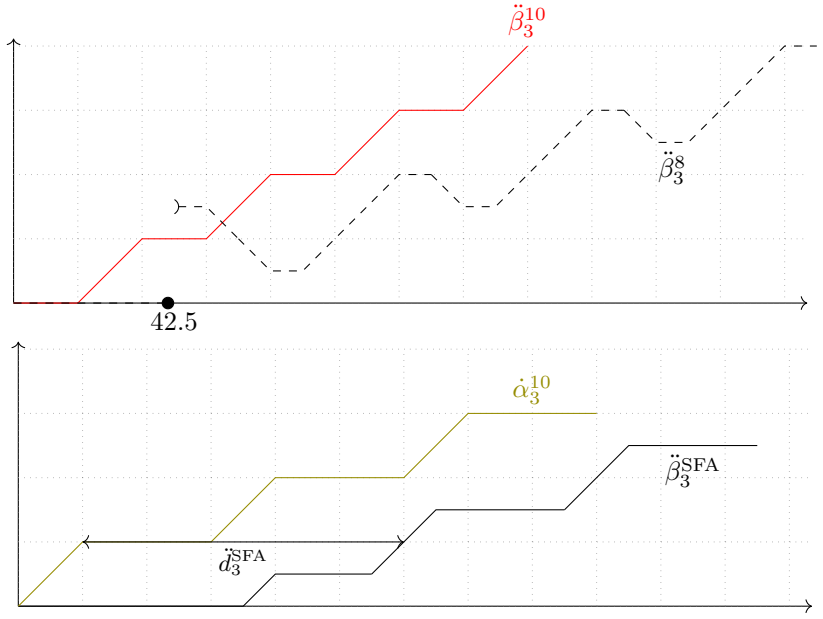

Figure 31: Local and end-to-end residual services functions for the flow f_3, SFA method, modeling per flow and per queue constant packet size. The background dotted grid has step 17. $\ddot{\beta}_3^{\mathrm{SFA}} = \ddot{\beta}_3^{10} * \ddot{\beta}_3^{8}, \ddot{d}_3^{\mathrm{SFA}} = hDev\left(\dot{\alpha}_3^{10}, \ddot{\beta}_3^{\mathrm{SFA}}\right)$.

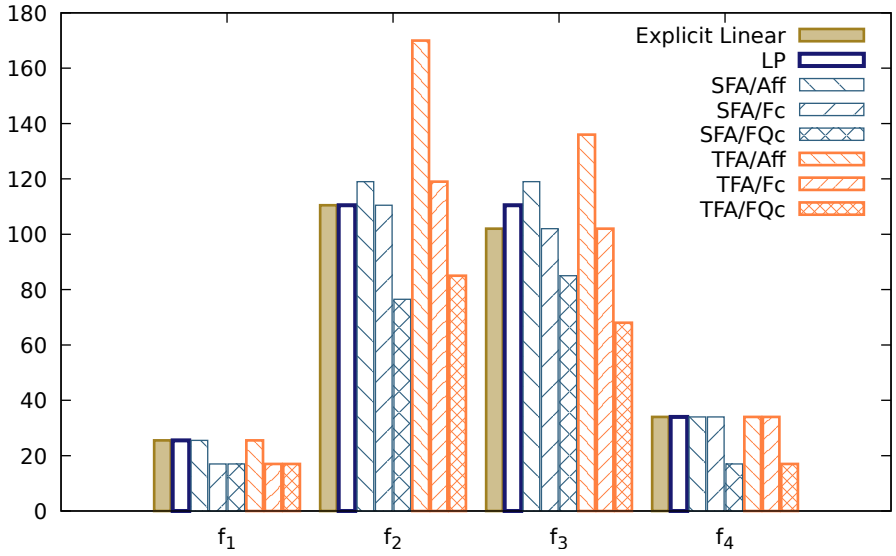

Figure 32: Upper bounds on delay, per flow and per method, first example (topology from Figure 20), first experiment (original values, parameters from Table 3).

(2019) with a different value of θ, and in this case, SFA was often worst than TFA. Note that for f_4, the TFA/Aff method gives a better bound than the SFA/Aff. This is related to the choice of the θ value that is not the best.

The results of the LP method deserve a discussion: whereas the LP method has been designed to compute the exact worst case, the explicit linear method result is smaller for flow f_3. The reason is that the LP method does not model the *shaping* introduced by the link. Having stronger assumptions, the explicit linear method reduces the set of admissible flows, and even if it does not compute the maximum of this set, but only an upper bound, this upper bound is smaller than the maximum of the larger set where no shaping constraint exists. The same happens when considering packets of fixed sizes in SFA: with more assumptions, and considering non concave/convex piecewise-linear functions (Figures 8, 18), the bounds are better, even if the core of the resolution method is worse.

8.2 Small example, second experiment, splitting flows

The second experiment is a modification of the first one: each flow f_i is split into two flows $f_{i,1}$, $f_{i,2}$ with the same routing; a flow rate divided by two; $f_{i,1}$ has maximal packet size 9; and $f_{i,2}$ has maximal packet size 8. This example has more flows, each queue is used by at least two flows, and one cannot assume that all packets in a queue have the same size. The parameters are listed in

Flow	$f_{1,1}$	$f_{2,1}$	$f_{3,1}$	$f_{4,1}$	$f_{1,2}$	$f_{2,2}$	$f_{3,2}$	$f_{4,2}$
Rate	$\frac{1}{3}$	$\frac{1}{6}$	$\frac{1}{6}$	$\frac{1}{6}$	$\frac{1}{3}$	$\frac{1}{6}$	$\frac{1}{6}$	$\frac{1}{6}$
Max. Packet Size.	9	9	9	9	8	8	8	8
Burst	6	7.5	7.5	7.5	$\frac{16}{3}$	$\frac{20}{3}$	$\frac{20}{3}$	$\frac{20}{3}$

Table 4: Flow parameters, small example (topology of Figure 20), second experiment (splitting flows).

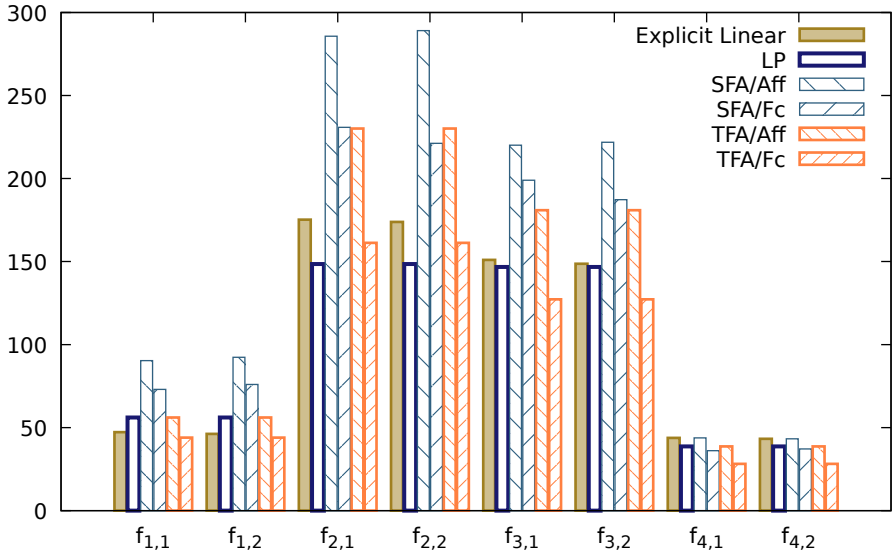

Figure 33: Upper bounds on delay, per flow and per method, small example (topology from Figure 20), second experiment (splitting flows, parameters from Table 4).

Table 4. Note that splitting a flow increases the initial burst[12]. This is due to the fact that the MPPA NoC ingress traffic limiter must always allow a packet to be fully sent at ingress: then, reducing the per-flow rate increases the burst size w.r.t. the packet size (cf. Figure 3 and eq. (20)).

The results are reported in Figure 33. The explicit linear method does not give the best results, but nevertheless give good bounds. The LP method is better than the other affine methods (explicit linear, fluid TFA and fluid SFA), and even gives the best results for the flows $f_{2,1}, f_{2,2}$. The fluid SFA gives the worst results for $f_{1,\cdot}, f_{2,\cdot}, f_{4,\cdot}$ but is better than the fluid TFA for $f_{3,\cdot}$.

If all packets of a given flow have the same size, modeling it clearly improves the bound.

The difference between TFA and SFA can be illustrated on the affine case for

[12] If $r(f)$ (resp. $l^{\max}(f)$ and $b(f)$) denotes the rate (resp. maximal size and burst) of the flow f, then $r(f_{i,1})+r(f_{i,2}) = r(f_i)$, $l^{\max}(f_{i,1})+l^{\max}(f_{i,2}) = l^{\max}(f_i)$, but $b(f_{i,1})+b(f_{i,2}) > b(f_i)$

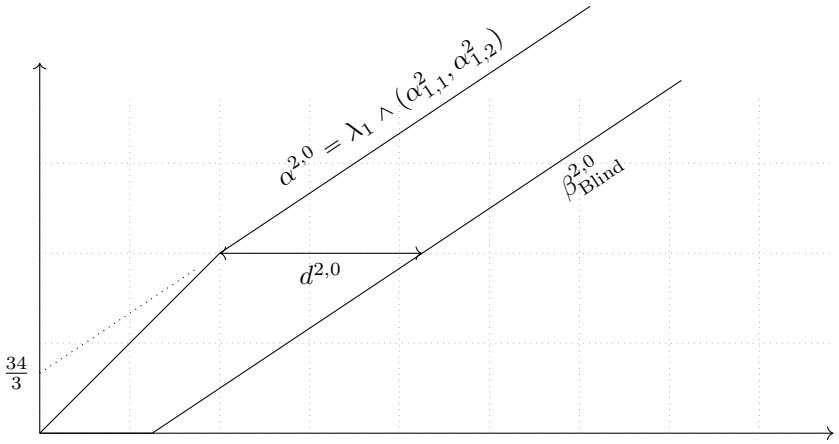

Figure 34: Functions related to the crossing of R2 per flow $f_{1,1}, f_{1,2}$, TFA method, fluid modeling, small example, second experiment. The background dotted grid has step 17. $d^{2,0} = hDev\left(\alpha^{2,0}, \beta^{2,0}_{\text{Blind}}\right) = \frac{153}{4}$, $\frac{34}{3} = 6 + \frac{16}{3}$

the flow $f_{1,1}$. It crosses the sequence of queues $q^{0,0}, q^{2,0}, q^{10,2}$, and shares this path with the flow $f_{1,2}$. Let $\alpha^{0,0}_{1,1}, \alpha^{2,0}_{1,1}, \alpha^{0,0}_{1,2}, \alpha^{2,0}_{1,2}$ denote respectively the arrival curves of the flow $f_{1,1}$ at $q^{0,0}$ and $q^{2,0}$ input and of the flow $f_{1,2}$ at $q^{0,0}$ and $q^{2,0}$ input.

The per queue residual servers are $\beta^{0,0} = \lambda_1$, $\beta^{2,0}_{\text{Blind}} = \beta_{\frac{2}{3}, \frac{85}{4}}$, $\beta^{10,2} = \lambda_1$.

In the first router, R0, due to the shaping, TFA computes a null delay: $d^{0,0} = hDev(\alpha^{0,0}, \beta^{0,0}) = hDev\left(\lambda_1 \wedge (\alpha^0_{1,1} + \alpha^0_{1,2}), \lambda_1\right) = 0$. Then the arrival curves are not modified, *i.e.* $\alpha^{2,0}_{1,i} = \alpha^{0,0}_{1,i}$ for $i \in \{1, 2\}$. In the second router, R2, $d^{2,0} = hDev\left(\alpha^{2,0}, \beta^{2,0}_{\text{Blind}}\right) = \frac{153}{4} = 38.25$ as illustrated in Figure 34. Note that the aggregate burst $\frac{34}{3}$ is the sum of the bursts of flows $f_{1,1}$ and $f_{1,2}$. In the third router, R10, like in the first one, the flows suffer no delay.

SFA computes a FIFO residual service per server, and requires a θ parameter in each server. Let θ^{R0}, θ^{R2} and θ^{R10} be the respective values in routers R0, R2, R10. In the first router R0, $f_{1,1}$ is competing with flow $f_{1,2}$. In this router, the burst of $f_{1,2}$ is $\frac{16}{3}$. Along their common path, the minimal service rate is the one offered in R2, $\frac{2}{3}$. Then, eq. (34) yields $\theta^{\text{R0}} = 0 + \frac{16/3}{2/3} = 8$ and $\beta^0_{1,1} = \left[\lambda_1 - (\alpha^0_{1,2} * \delta_8)\right]^+ \wedge \delta_8$ as illustrated in Figure 35. In the second router R2, there is no new interfering flow, so $\theta^{\text{R2}} = \frac{85}{4} + 0$ and $\beta^2_{1,1} = \beta_{\frac{1}{3}, \frac{149}{3}}$. The same, in the third router R10, $\theta^{\text{R10}} = 0 + 0 = 0$ and $\beta^{10}_{1,1} = \beta_{\frac{2}{3}, \frac{217}{8}}$.

The end-to-end service computed by SFA is $\beta^{\text{SFA}}_{1,1}$, plotted in Figure 35, leads to a delay bound $d^{\text{SFA}}_{1,1} = hDev\left(\alpha^0_{1,1}, \beta^{\text{SFA}}_{1,1}\right) = \frac{723}{8} = 90.375$. Notice that the step in curve $\beta^0_{1,1}$ is absent of the end-to-end service $\beta^{\text{SFA}}_{1,1}$.

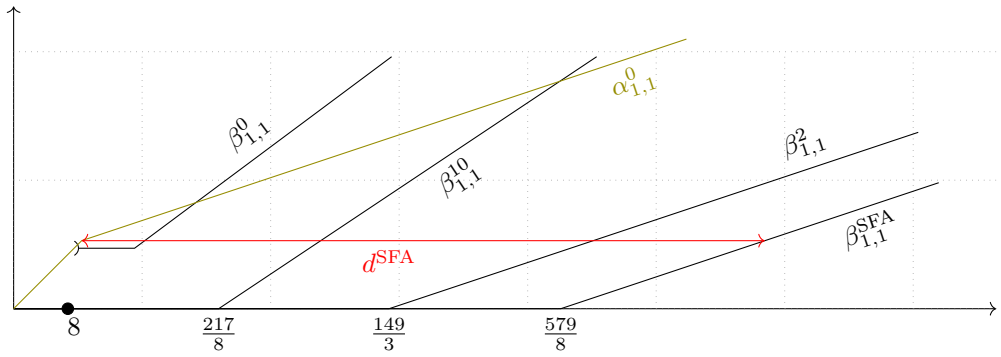

Figure 35: Local and end-to-end residual services functions for the flow $f_{1,1}$, SFA method, fluid modeling, small example, second experiment. The background dotted grid has step 17. $\beta^{\mathrm{SFA}}_{1,1} = \beta^0_{1,1} * \beta^2_{1,1} * \beta^{10}_{1,1}$, $d^{\mathrm{SFA}}_{1,1} = hDev\left(\alpha^0_{1,1}, \beta^{\mathrm{SFA}}_{1,1}\right)$.

Flow	f_1	f_2	f_3	f_4
Rate	$\frac{2}{3}$	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{3}$
Max. Packet Size.	70	70	70	70
Burst	$\frac{70}{3}$	$\frac{140}{3}$	$\frac{140}{3}$	$\frac{140}{3}$

Table 5: Flow parameters, small example (topology of Figure 20), third experiment (large packet size).

Another configuration of θ parameters has been tested: instead of considering the burst in the first router R0, let consider $\theta^{\mathrm{R0}} = 0$, $\theta^{\mathrm{R2}} = \frac{145}{4}$, $\theta^{\mathrm{R10}} = 0$. The intuition here was to set the burst at the server with the smaller rate, to set an initial value of the burst that aligns the step and the inflection point of the arrival curve, and to update it by dichotomy up to a reaching the minimal end-to-end delay bound value $\frac{611}{8} = 76.375$. The service curves are plotted in Figure 36.

8.3 Small example, third experiment, large packet size

The third experiment uses the same parameters as the initial experiment (Section 8.1), but with large packet size (70 flits). The flow parameters are given in Table 5, and the results are reported in Figure 37.

The results look very similar to those of the first experiment, but one has to pay attention that the range of values is very different: whereas the range of values was [0,180] in the first experiment (Figure 32), it is [0,700] in Figure 37. Since the packet and burst sizes are $\frac{70}{17} \approx 4.11$ larger, the delays also are globally four times larger.

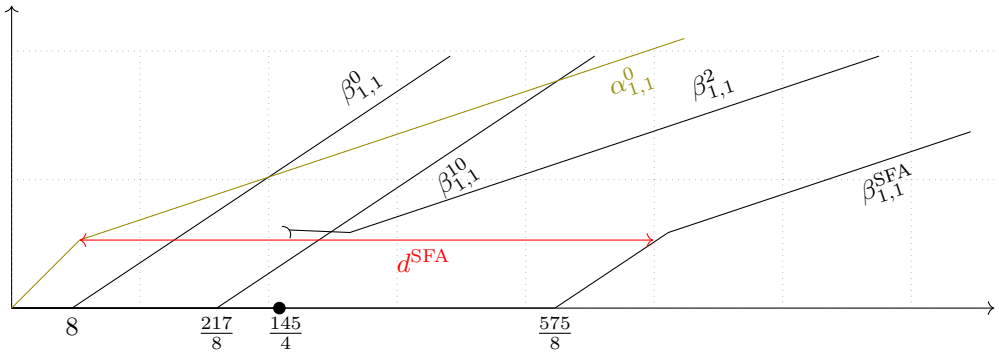

Figure 36: Local and end-to-end residual services functions for the flow $f_{1,1}$, SFA method with $\theta^{\mathrm{R0}} = 0$, $\theta^{\mathrm{R2}} = \frac{145}{4}$, $\theta^{\mathrm{R10}} = 0$, fluid modeling, small example, second experiment. The background dotted grid has step 17. $\beta^{\mathrm{SFA}}_{1,1} = \beta^0_{1,1} * \beta^2_{1,1} * \beta^{10}_{1,1}$, $d^{\mathrm{SFA}}_{1,1} = hDev\left(\alpha^0_{1,1}, \beta^{\mathrm{SFA}}_{1,1}\right)$.

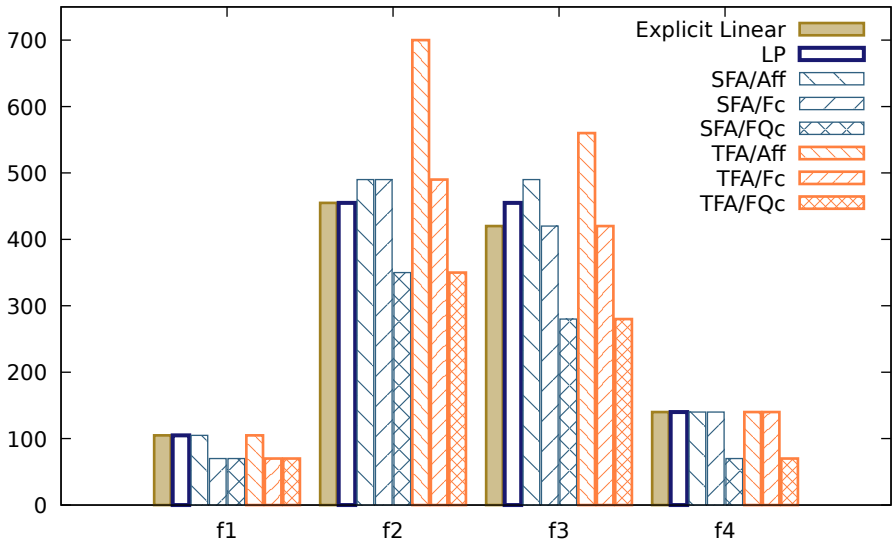

Figure 37: Upper bounds on delay, per flow and per method, small example (topology from Figure 20), third experiment (large packets, parameters from Table 5).

Length	2	3	4	5	6	7	8
Number of flows	16	22	31	26	22	10	1
Number of LP time-out	0	0	0	2	4	8	1

Table 6: Number of flows with a given length (Mean: 4.4) and number of time-out with method LP (with time-out at 2mn), large example, first experiment.

8.4 Large example (full MPPA NoC), first experiment: 128 flows

The large example is based on the MPPA architecture presented in Figure 1. Each node is the source of 4 flows, with randomly chosen destination, leading to 128 flows. Each flow has a constant packet size of 17 flits, and the routing and rate allocations have been generated using the strategies presented in Dupont de Dinechin et al. (2014), Boyer et al. (2018). The average flow length is 4.4, and the length distribution is listed in Table 6. The average link load is 44% (168 links are used), 4 links have a load of 100%, 7 of load in $[80\%, 89\%]$ and 36 a load in $[50\%, 79\%]$.

Furthermore, the upper bounds on delays have been computed using some of the method presented in the previous sections. The NetCalBounds tool, implementing the LP method from Bouillard and Stea (2014) has been limited to 2mn of computing time for each flow[13]. In case of timeout, two upper bounds have been computed: one using the NetCalBounds tool with the ULP method, and the other with the deborah tool, and the minimum of both is used. The bound obtained will be denoted LP|ULP|deb. The number of LP timeout is listed Table 6. The SFA/Fc and SFA/FQc methods require the computation of the convolution between complex service curves, and its leads to very long computation times. Moreover, the previous experiments have shown that they give similar results to the TFA/Fc and TFA/FQc methods. Therefor, they have not been used in this experiment.

The bound computed for each flow with each method appears in Figure 38, where flows have been sorted w.r.t. the bound computed by the explicit linear method (which yields to a smooth curve for this method).

The results are quite similar to the one on the small test cases: the TFA/FQc method (that captures both shaping and the fixed packet size nature of flows) outperforms all other methods in most cases. The LP, ULP or deborah tools (that does not capture the shaping neither the packet sizes) gives almost always a worse value than the explicit linear methods (that captures the shaping but not accurately the packet sizes). The TFA/Aff and TFA/Fc methods behave sometime better, sometime worse than explicit linear or LP|ULP|deb. When considering average values (last column of Figure 39), the importance of shaping appears clearly: the explicit linear gives bound one third less than LP|ULP|deb. The TFA methods is quite poor with an affine model, but once

[13]To be exact, the NetCalBounds tool generates a MILP problem, in negligible time. The computation time comes from the lp_solve tool, that we used to solve the MILP problem.

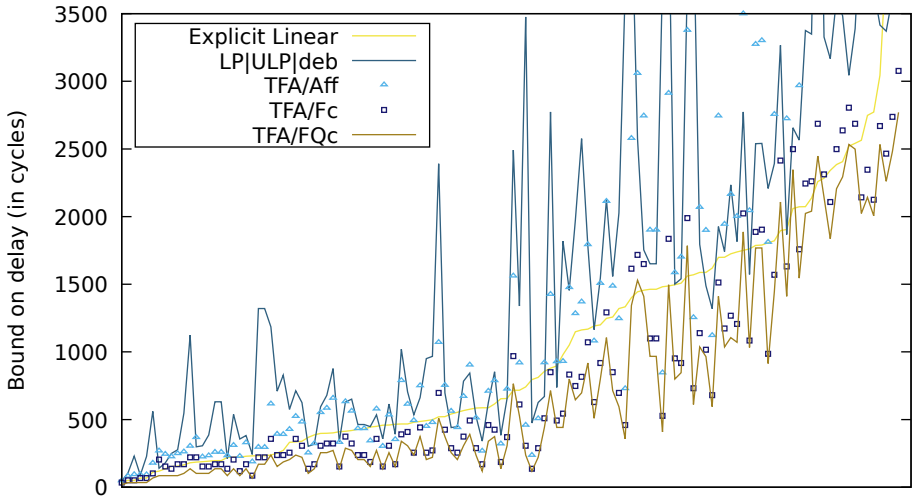

Figure 38: Upper bounds on delay, per flow and per method, large example, first experiment. The flows are sorted by bound value with explicit linear method.

modeling constant packet sizes, its gives the best results.

One may wonder if the path length has an influence, guessing that methods using the PBOO principle may have better results on long paths. Then, Figure 40 plots the same bounds as Figure 38, but flows are grouped by flow length before being sorted by bound of the explicit linear method. It appears clearly that longer paths have larger delays, but showing the relation between methods requires other figures. Figure 41 displays, for each flow, the ratio between explicit linear and LP|ULP|deb w.r.t TFA/FQc, using the same flow ordering as in Figure 40. Figure 39 shows the average bound computed by each method, depending on the flow length.

Both results confirm the relations obtained between methods, independently of the path length. The gain obtained by the PBOO principle is mitigated by a looser modeling of the residual service in each router.

8.5 Large experiment (full MPPA NoC), second experiment: 256 flows

The second experiment on the full MPPA NoC topology considers 8 flows per cluster. Since there are 8 traffic limiters per cluster, this is the maximum that can be done on the MPPA processor.

The distribution of flow length is given in Table 7, and the average length is 4.6. In this example, the average link load is 34%, and only 2 links have a 100% load, 5 are in interval $[90\%, 99\%]$ and 20 in interval $[50\%, 89\%]$.

The individual bounds per flow are not displayed, since the relations between the methods are the same as in previous experiment. Only the mean bound per

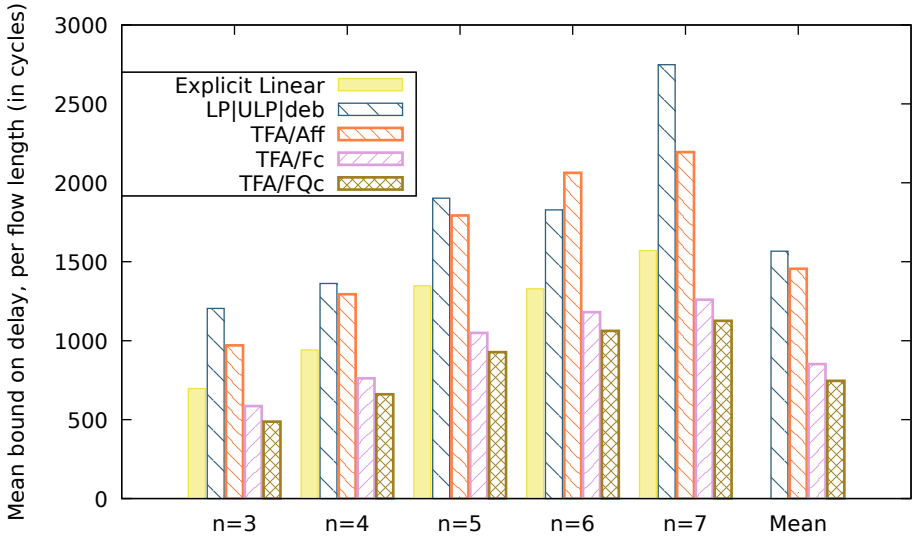

Figure 39: Mean value of bounds (in cycle), per flow length n, large example, first experiment.

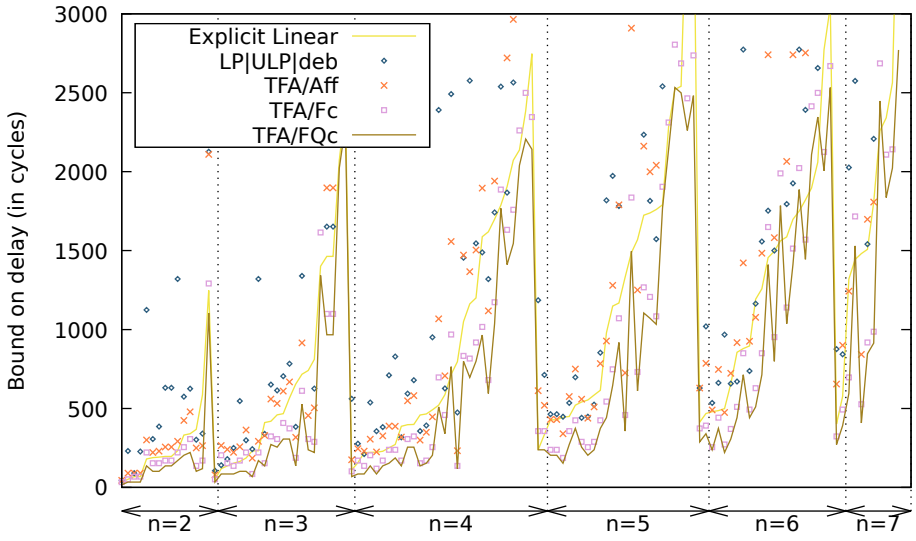

Figure 40: Upper bounds on delay, per flow and per method, large example, first experiment. The flows are sorted first by flow length n then by bound value with explicit linear method.

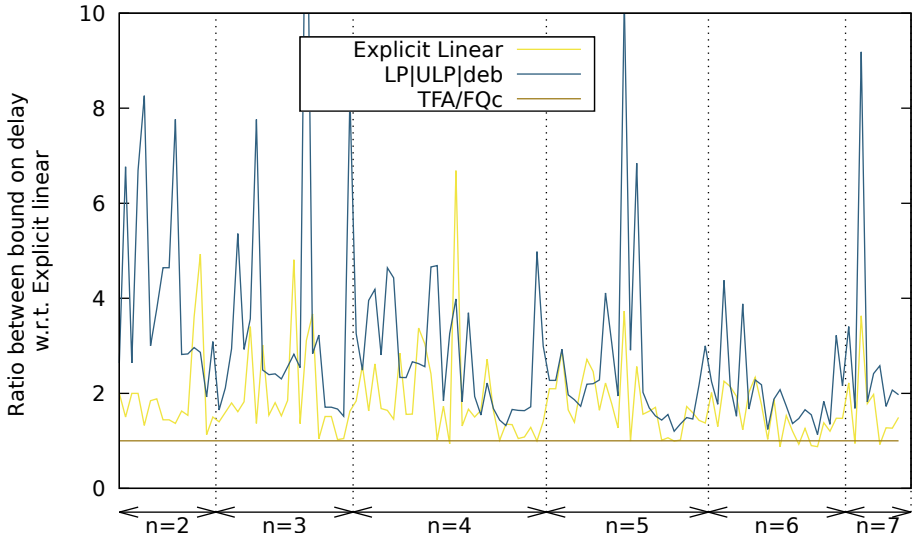

Figure 41: Ratio between each method w.r.t. TFA/FQc one, large example, first experiment, same sorting as in Figure 40.

.

Length	2	3	4	5	6	7	8
Number of flows	18	45	57	60	49	21	6
Number of LP time-out	0	0	0	12	31	21	6

Table 7: Number of flows with a given length (Mean: 4.4) and number of time-out with method LP (with time-out at 2mn), large example, second experiment.

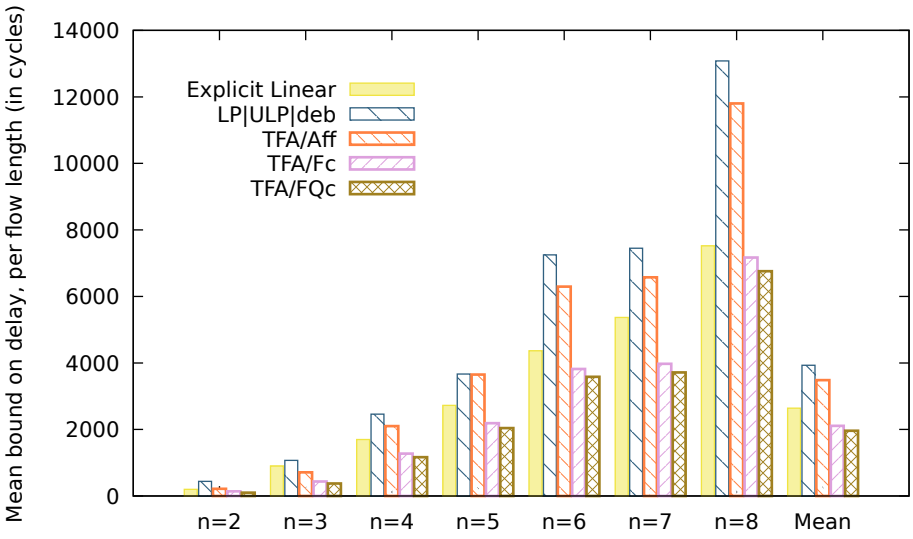

Figure 42: Mean value of bounds (in cycle), per flow length n, large example, second experiment.

flow length are given in Figure 42. Since they are more flows, they are more conflicts per router, and the worst-case delays are increased. The mean gain between the explicit linear and TFA/FQc methods becomes 25%, whereas it was 20% in the previous experiment with 128 flows.

8.6 Conclusions on the case studies

Theses experiments on a realistic case study present some trends on the applicability of network calculus for NoC and some insights on research opportunities.

Usability of LP Whereas the LP approach has a theoretical exponential cost, related to the length of paths, it can in practice be used for NoC, as long as the paths are not very long. Table 6 shows that even if it was not able to deal with paths of size greater than 6 (in less than 2mn), it has computed bounds for 17 out of 22 paths of length 6 (*i.e.* 77%) and 24 out of 26 (*i.e.* 92%) of paths of length 5, and all for smaller paths. Notice also that the LP problems has been solved using the `lp_solve` solver Berkelaar et al. (2018). Other solvers may have different resolution time Bondorf et al. (2017).

Influence of shaping Modeling the shaping has a major influence on the accuracy of the results. It has been previously shown in the context of AFDX network in Frances et al. (2006) and confirmed in Boyer and Fraboul (2008), Scharbarg et al. (2009), Zhao et al. (2013). It is confirmed on these experiments on the MPPA NoC. While the LP method computes the exact worst case for the

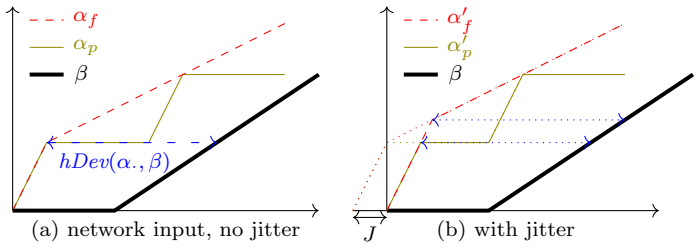

Figure 43: Gain related to modeling of packets in arrival curve.

FIFO policy (without considering shaping), it is often outperformed by methods that model link shaping in several cases.

The shaping can be easily added in the LP method, since it only amount to adding new linear constraints. The LP method introduces some well chosen time instants t_i and keeps only from the relation $D \geqslant A * \beta_{R,T}$ the inequality $D(t_i) \geqslant A(t_j) + R(t_i - t_j)$ as a constraint

$$\texttt{Dti} \geqslant \texttt{Atj} + R(\texttt{ti} - \texttt{tj} - T) \tag{41}$$

where $\texttt{ti}, \texttt{tj}, \texttt{Dti}, \texttt{Atj}$ are program variables that respectively represents the two instants t_i, t_j and flow departure and arrival values $D(t_i), A(t_j)$. Then, encoding the fact that departure D is constrained by shaping of a link of constant capacity C can be encoded as

$$\texttt{Dtm} + \texttt{Dtn} \leqslant C(\texttt{tm} - \texttt{tn}). \tag{42}$$

for all variables $\texttt{Dtm}, \texttt{Dtn}$ related to the function D and the instants t_m, t_n in the program. When this departure is the aggregation of several flows ($D = D_1 + \cdots + D_n$), a similar expression has to be used.

Influence of packet size Modeling of packet size in data flows has also a beneficial impact (it has been shown in the context of AFDX in Boyer et al. (2011b),Boyer et al. (2012)), since it gives smaller arrival curves, and gives lower burst in the network. Looking at Figure 18, it is obvious that, α_p, the arrival curve modeling constant packet size is smaller than α_f, but the impact on the delay bound is not so obvious. Figure 43 illustrates this relation between the per packet arrival and the delay. Consider a fluid flow arrival, α_f and the associated per packet flow arrival α_p; even if $\alpha_f \leqslant \alpha_p$ they both have the same horizontal deviation with the service curve β. But after crossing the first node, the flow has a new arrival curve. To ease the discussion, assume that this server creates a jitter J, and has a shaping curve equals to the link input shaping. Then, the respective arrival curves at the next node will be α'_f and α'_p. And in this case, the delays associated with each arrival curve are different, as illustrated in Figure 43.b.

Conversely, the modeling of packet sizes in the round robin scheduling policy gives a bigger service curve, as illustrated in Figure 8. But the benefit in the

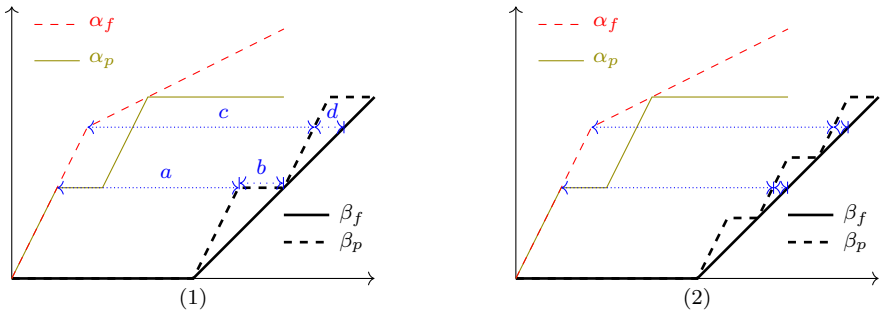

Figure 44: Gain related to modeling of packets in both arrival and service curves, whether service packet size fits arrival packet size (case (1)), or not (case (2)). $a = hDev(\alpha_p, \beta_p)$, $a+b = hDev(\alpha_p, \beta_f)$, $c = hDev(\alpha_f, \beta_p)$, $c+d = hDev(\alpha_f, \beta_f)$,

delay evaluation is related to the modeling of the packet size in the arrival curve also. Figure 44-(1) illustrates the situation where a single flow is entering a round-robin arbiter, and all packets in the flow have the same size. This flow (resp. arbiter) can be modeled using either a fluid arrival curve α_f or a packetized one α_p (resp. a fluid service curve β_f or a packetized one β_p). Then, the burst fits exactly the height of the first step of the curve, and the delay a is smaller than considering a fluid residual service (delay $a+b$), or even considering a fluid arrival curve and a fluid residual service (delay $c+d$). But it might also happen that both sizes do not fit, like in Figure 44-(2), and even if there is a gain at modeling packet size, it may be smaller.

Nevertheless, an accurate modeling of packet sizes requires abandoning the efficient class of piecewise-linear concave/convex functions to handle more general classes, like the Ultimately Pseudo Periodic class defined in Bouillard and Thierry (2008). The encoding of the constraints in an integer linear program is not so straightforward and may moreover increase the computation time.

SFA and TFA Almost all published studies in network calculus report that the TFA is the less effective approach, except in a few specific cases, as presented in Bondorf and Schmitt (2016). But all these studies consider *blind multiplexing*, and the residual service computed in this case with Theorem 3 is known to be tight, whereas for the FIFO policy, SFA requires the choice of a θ parameter in each server. In the experiments, a strategy inspired by Fidler (2003) has been used. Depending on the case study, this lead to bounds smaller or greater than the one of TFA. A better strategy can lead to better result: for example, different values of θ have been used for a single flow, leading to a smaller bound (cf. Section 8.2 and Figures 35, 36). But it does not exist, up to our knowledge, any strategy for choosing this parameter in the general case (and in the specific case of piecewise-linear concave/convex function, one better have to use the LP method). In other words, SFA is certainly a good approach when a good residual service per flow is known, which is not the case for the FIFO multiplexing policy

up to now.

Last, one have to pay attention to the fact that even if all methods give similar results *on average* (cf. Figures 39, 42), for a given flow, the difference may be very large (cf. Figure 41). Nevertheless, since all are valid bounds, one may run several or all methods and take the minimum of all bounds.

9 Conclusion

The MPPA2-256 processor, presented in Saidi et al. (2015), integrates 256 processing cores and 32 management cores on a chip, communicating through a shared NoC. Before embedding critical real-time application on such architecture, one needs some method to compute some upper bound on the communication delay introduced by the NoC resource sharing between communication flows.

In this paper, we have presented different ways to model the MPPA NoC using the deterministic network calculus framework: the explicit linear model, with flow burstiness as the main variables; the general purpose LP method, developed to get the exact worst case in case of FIFO network with piecewise-linear arrival functions and service curves; the SFA and TFA approaches, that have been adapted to per queue round-robin and per flow FIFO policies, and enhanced in the specific case of flows with constant packet sizes.

They have been compared, first on a small already published example, to get a comprehensive view on their differences, and to compare new methods with the previous one on a known example. Thereafter, they have been compared on a larger case study, with 128 and 256 data flows.

All experiments confirm a well known fact: when the flow burstiness is limited by link capacity, modeling this shaping has a major impact on results. Moreover, when all packets in a flow have the same size, modeling this also improves the bounds, especially in the case of the round-robin policy. And modeling these aspects of the system can outperform exact approaches that do not. In other words, there always is a trade-off between the accuracy of the model and the tightness of the approach. In the case of the MPPA NoC, shaping by the link capacity and the effects of the packet sizes are major parts that must be modeled to get good bounds.

Moreover, as claimed in Bondorf and Schmitt (2016), "there is a job for everyone": on the small case studies, no method always gives better results than the others in case of packets of variables sizes (fluid modeling), but in case of packets of constant size, the SFA and TFA algorithms with "packet-accurate" arrival and service curves give the best results. The same results appear on the large case studies: in case of packets of constant size, the TFA algorithms with "packet-accurate" arrival and service curve currently gives bounds 20%-25% smaller than any other, on average.

However, it does not mean that the TFA approach is, inherently, better than other approaches. The TFA approach is somehow the simplest analysis,

and this is perhaps the reason why it was easier to model and compute shaping and constant packet size in TFA.

Computing better bounds with the SFA approach will face two challenges. The first is the computation of a good residual service with FIFO multiplexing, *i.e.* the choice of a good θ_i^j parameter for each flow i in each crossed queue j (as discussed in Sections 7.2 and 8.2). Some machine learning techniques may be applied, like in Geyer and Bondorf (2019). The second one is the computation of convolutions with non convex residual service functions, which is currently too costly. One solution could be to compute only a finite prefix, like in Lampka et al. (2017).

Computing better bounds with the LP method will require encoding in the MILP the shaping introduced by the link and the non-convex curves. Since the shaping of the link is just another linear constraint, it can we expect that it may be added without introducing new variables, as presented in Section 8.6, and with limited impact on resolution time. On the opposite, the encoding of non concave/convex constraint will certainly imply the introduction of several Boolean variables. And this will likely negatively impact the resolution time.

Also note that the fluid approaches, using only rate-latency service curves, (Explicit Linear and TFA/Aff) may benefit from an enhancement of delay dedicated to the case of rate-latency service presented in Mohammadpour et al. (2019).

References

Abdallah L, Jan M, Ermont J, Fraboul C (2015) Wormhole networks properties and their use for optimizing worst case delay analysis of many-cores. In: Proc. of the 10th IEEE Int. Symp. on Industrial Embedded Systems (SIES 2015), pp 1–10, DOI 10.1109/SIES.2015.7185041

Ayed H, Ermont J, Scharbarg JL, Fraboul C (2016) Towards a unified approach for worst-case analysis of Tilera-like and Kalray-like NoC architectures. In: Proc. of the 12th IEEE World Conf. on Factory Communication Systems (WFCS 2016), WiP Session, IEEE, Aveiro, Portugal, DOI 10.1109/WFCS.2016.7496535

Berkelaar M, Eikland K, Notebaert P (2018) lp_solve. http://lpsolve.sourceforge.net/

Bisti L, Lenzini L, Mingozzi E, Stea G (2008) Estimating the worst-case delay in fifo tandems using network calculus. In: Proc. of the 3rd Int. Conf. on Performance Evaluation Methodologies and Tools, ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), ICST, Brussels, Belgium, Belgium, ValueTools '08, pp 1–67, DOI 10.4108/ICST.VALUETOOLS2008.4388

Bisti L, Lenzini L, Mingozzi E, Stea G (2010) DEBORAH: a tool for worst-case analysis of FIFO tandems. In: Margaria T, Steffen B (eds) Proc. of the 4th

Int. Symp. On Leveraging Applications of Formal Methods, Verification and Validation (ISoLA 2010), Springer, LNCS, DOI 10.1007/978-3-642-16558-0_15

Bisti L, Lenzini L, Mingozzi E, Stea G (2011) Deborah home page. http://cng1.iet.unipi.it/wiki/index.php/Deborah

Bondorf S (2017) Better bounds by worse assumptions – improving network calculus accuracy by adding pessimism to the network model. In: Proc. of the IEEE Int. Conference on Communications (ICC 2017), Symposium on Communications QoS, Reliability and Modeling (CQRM), IEEE, Paris, DOI 10.1109/ICC.2017.7996996

Bondorf S, Schmitt J (2016) Improving cross-traffic bounds in feed-forward networks – there is a job for everyone. In: Proc. of the 18th Int. GI/ITG Conf. on "Measurement, Modelling and Evaluation of Computing Systems" and "Dependability and Fault Tolerance" (MMB&DFT 2016), Deutchland, DOI 10.1007/978-3-319-31559-1_3

Bondorf S, Schmitt JB (2015) Calculating accurate end-to-end delay bounds – you better known your cross-traffic. In: Proc. of the 9th EAI Int. Conf. on Performance Evaluation Methodologies and Tools (ValueTools 2015), Berlin, Germany, DOI doi.org/10.4108

Bondorf S, Nikolaus P, Schmitt JB (2017) Quality and cost of deterministic network calculus – design and evaluation of an accurate and fast analysis. Proc of the ACM on Measurement and Analysis of Computing Systems (POMACS) 1(1):34, DOI 10.1145/3084453

Bouillard A (2011) Composition of service curves in network calculus. In: Proceedings of the 1st International Workshop on Worst-Case Traversal Time (WCTT'2011), WCTT '11, pp 35–42, DOI 10.1145/2071589.2071594

Bouillard A (2017) Netcalbounds home page. URL `https://github.com/annebouillard/NetCalBounds`

Bouillard A, Stea G (2012) Exact worst-case delay for FIFO-multiplexing tandems. In: Proc. of the 6th International Conference on Performance Evaluation Methodologies and Tools (ValueTools 2012), Cargese, France, DOI 10.4108/valuetools.2012.250090

Bouillard A, Stea G (2014) Exact worst-case delay for FIFO-multiplexing feed-forward networks. IEEE/ACM Transactions on Networking DOI 10.1109/TNET.2014.233207

Bouillard A, Stea G (2015) Worst-Case Analysis of Tandem Queueing Systems Using Network Calculus. In: Bruneo, Distefano (eds) Quantitative Assessments of Distributed Systems

Bouillard A, Thierry E (2008) An algorithmic toolbox for network calculus. Discrete Event Dynamic Systems 18(1):3–49, DOI 10.1007/s10626-007-0028-x

Bouillard A, Gaujal B, Lagrange S, Thierry E (2008) Optimal routing for end-to-end guarantees using network calculus. Performance Evaluation 65(11-12):883–906, DOI 10.1016/j.peva.2008.04.008

Bouillard A, Jouhet L, Thierry E (2010) Tight performance bounds in the worst-case analysis of feed-forward networks. In: Proceedings of the 29th Conference on Computer Communications (INFOCOM 2010), pp 1–9, DOI 10.1109/INFCOM.2010.5461912

Bouillard A, Boyer M, Le Corronc E (2018) Deterministic Network Calculus – From theory to practical implementation. ISBN: 978-1-119-56341-9, Wiley

Boyer M, Fraboul C (2008) 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), IEEE, pp 11–20, DOI 10.1109/WFCS.2008.4638728

Boyer M, Migge J, Fumey M (2011a) PEGASE, a robust and efficient tool for worst case network traversal time. In: Proc. of the SAE 2011 AeroTech Congress & Exhibition, SAE International, Toulouse, France, DOI 10.4271/2011-01-2711

Boyer M, Migge J, Navet N (2011b) An efficient and simple class of functions to model arrival curve of packetised flows. In: Proc. of the 1st Int. Workshop on Worst-Case Traversal Time (WCTT'2011), ACM, New York, NY, USA, pp 43–50, DOI 10.1145/2071589.2071595

Boyer M, Navet N, Fumey M (2012) Experimental assessment of timing verification techniques for afdx. In: Proc. of the 6th Int. Congress on Embedded Real Time Software and Systems, Toulouse, France, URL `https://hal.archives-ouvertes.fr/hal-02189869`

Boyer M, Dufour G, Santinelli L (2013) Continuity for network calculus. In: Proc of the 21th International Conference on Real-Time and Network Systems (RTNS 2013), ACM, Sophia Antipolis, France, pp 235–244, DOI 10.1145/2516821.2516840

Boyer M, Dupont de Dinechin B, Graillat A, Havet L (2018) Computing routes and delay bounds for the network-on-chip of the Kalray MPPA2 processor. In: Proc. of the 9th European Congress on Embedded Real Time Software and Systems (ERTS^2 2018), Toulouse, France, URL `https://hal.archives-ouvertes.fr/hal-01707911`

Boyer M, Graillat A, Dupont De Dinechin B, Migge J (2019) Comparing strategies to bound the latencies of the MPPA Network-on-Chip (Extended version), URL `https://hal.archives-ouvertes.fr/hal-02122874`, working paper

Burns A, Harbin J, Indrusiak L (2014) A wormhole NoC protocol for mixed criticality systems. In: Proc. of the IEEE Real-Time Systems Symposium (RTSS 2014), IEEE, Rome, Italy, pp 184–195, DOI 10.1109/RTSS.2014.13

Carle T, Djemal M, Potop-Butucaru D, De Simone R, Zhang Z (2014) Static mapping of real-time applications onto massively parallel processor arrays. In: Proc. of the 14th Int. Conf. on Application of Concurrency to System Design (ACSD 2014), IEEE, pp 112–121, DOI 10.1109/ACSD.2014.19

Chang CS (2000) Performance Guarantees in communication networks. Telecommunication Networks and Computer Systems, Springer

Cholvi V, Echagüe J, Le Boudec JY (2002) Worst case burstiness increase due to FIFO multiplexing. Performance Evaluation 49(1–4):491 – 506, DOI 10.1016/S0166-5316(02)00116-5

Cruz RL (1991) A calculus for network delay, part I: Network elements in isolation. IEEE Transactions on information theory 37(1):114–131, DOI 10.1109/18.61109

Dupont de Dinechin B, Graillat A (2017) Network-on-chip service guarantees on the Kalray MPPA-256 bostan processor. In: Proc. of the 2nd Inter. Workshop on Advanced Interconnect Solutions and Technologies for Emerging Computing Systems (AISTECS'17), DOI 10.1145/3073763.3073770

Dupont de Dinechin B, Van Amstel D, Poulhiès M, Lager G (2014) Time-critical computing on a single-chip massively parallel processor. In: Proc. of the Design, Automation and Test in Europe Conference and Exhibition (DATE 2014), IEEE, pp 1–6, DOI 10.7873/DATE.2014.110

Ferrandiz T, France F, Fraboul C (2009) A method of computation for worst-case delay analysis on SpaceWire networks. In: Proc. of the IEEE Symposium on Industrial Embedded Systems (SIES'09), Ecole Polythechnique de Lausane, Switzerland, pp 19–27, DOI 10.1109/SIES.2009.5196187

Ferrandiz T, Frances F, Fraboul C (2011) Worst-case end-to-end delays evaluation for spacewire networks. Discrete Event Dynamic Systems 21(3):339–357, DOI 10.1007/s10626-011-0103-1

Fidler M (2003) Extending the network calculus pay bursts only once principle to aggregate scheduling. In: Proc. of the Second International Workshop on Quality of Service in Multiservice IP Network (QoS-IP 2003), Milano, Italy, LNCS, vol 2601, pp 19–34, DOI 10.1007/3-540-36480-3_2

Firoiu V, Le Boudec JY, Towsley JY, Zhang ZL (2002) Theories and models for internet quality of service. Procof the IEEE 90(9):1565–1591, DOI 10.1109/JPROC.2002.802002

Frances F, Fraboul C, Grieu J (2006) Using network calculus to optimize AFDX network. In: Proc. of the 3thd European congress on Embedded Real Time Software (ERTS06), Toulouse

Frangioni A, Galli L, Stea G (2014) Optimal joint path computation and rate allocation for real-time traffic. The Computer Journal 58(6):1416–1430, DOI 10.1093/comjnl/bxu053

Frangioni A, Galli L, Stea G (2017) Qos routing with worst-case delay constraints: Models, algorithms and performance analysis. Computer Communications 103(Supplement C):104 – 115, DOI 10.1016/j.comcom.2016.09.006

Georges J, Divoux T, Rondeau E (2005) Strict priority versus weighted fair queueing in switched ethernet networks for time critical applications. In: Proc. of the 19th IEEE International Parallel and Distributed Processing Symposium (IPDPS 2005), Denver, CO, USA, pp 141–141, DOI 10.1109/IPDPS.2005.413

Georges JP, Divoux T, Rondeau E (2011) Network calculus: application to switched real-time networking. In: Proc. of the 5th Int. ICST Conf. on Performance Evaluation Methodologies and Tools, ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), ICST, Brussels, Belgium, Belgium, VALUETOOLS '11, pp 399–407, URL `http://dl.acm.org/citation.cfm?id=2151688.2151733`

Geyer F, Bondorf S (2019) DeepTMA: Predicting effective contention models for network calculus using graph neural networks. In: Proc. of the IEEE Int. Conf. on Computer Communications (INFOCOM 2019), IEEE, Paris, France, DOI 10.1109/INFOCOM.2019.8737496

Giannopoulou G, Stoimenov N, Huang P, Thiele L, Dupont de Dinechin B (2016) Mixed-criticality scheduling on cluster-based manycores with shared communication and storage resources. Real-Time Systems 52(4):399–449, DOI 10.1007/s11241-015-9227-y

Giroudot F, Mifdaoui A (2020) Graph-based approach for buffer-aware timing analysis of heterogeneous wormhole nocs under bursty traffic. IEEE Access 8:32442–32463, DOI 10.1109/ACCESS.2020.2973891, URL `https://oatao.univ-toulouse.fr/25485/`

Henia R, Hamann A, Jersak M, Racu R, Richter K, Ernst R (2005) System level performance analysis - the SymTA/S approach. IEEE Proceedings on Computers and Digital Techniques 152(2):148 – 166, DOI 10.1049/ip-cdt:20045088

Jafari F, Lu Z, Jantsch A, Yaghmaee MH (2010) Optimal regulation of traffic flows in networks-on-chip. In: Proc. of the Conf. on Design, Automation Test in Europe (DATE 2010), pp 1621–1624, DOI 10.1109/DATE.2010.5457070

Kiasari AE, Jantsch A, Lu Z (2013) Mathematical formalisms for performance evaluation of networks-on-chip. ACM Computing Surveys (CSUR) 45(3):38, DOI 10.1145/2480741.2480755

Lampka K, Bondorf S, Schmitt JB, Guan N, Yi W (2017) Generalized finitary real-time calculus. In: IEEE Conference on Computer Communications (INFOCOM 2017), pp 1–9

Lenzini L, Mingozzi E, Stea G (2004) Delay bounds for FIFO aggregates: a case study. Computer Communications 28:287–299, DOI 10.1016/j.comcom.2004.10.003

Lenzini L, Martorini L, Mingozzi E, Stea G (2005) Tight end-to-end per-flow delay bounds in FIFO multiplexing sink-tree network. Performance Evaluations 63:956–987, DOI 10.1016/j.peva.2005.10.003

Lenzini L, Mingozzi E, Stea G (2007) End-to-end delay bounds in FIFO-multiplexing tandems. In: Glynn P (ed) Proc. of the 2nd Int. Conf. on Performance Evaluation Methodologies and Tools (ValueTool07, ICST, Nantes, France

Li X, Cros O, George L (2014) The trajectory approach for AFDX FIFO networks revisited and corrected. In: Proc. of the 20th Int. Conf. on Embedded and Real-Time Computing Systems and Applications (RTCSA'2014), Chongqing, China, DOI 10.1109/RTCSA.2014.6910523

Long Y, Lu Z, Yan X (2014) Analysis and evaluation of per-flow delay bound for multiplexing models. In: Proc. of the Design, Automation and Test in Europe Conference and Exhibition (DATE), 2014, IEEE, Dresden, Germany, pp 1–4, DOI 10.7873/DATE.2014.264

Martin S, Minet P (2004) The trajectory approach for the end-to-end response times with non-preemptive FP/EDF*. In: Proceedings of the Int. Conf. on Software Engineering Research and Applications (SERA'04), Springer, LNCS, vol 3647, pp 229–247, DOI 10.1007/11668855_17

Mohammadpour E, Stai E, Le Boudec JY (2019) Improved delay bound for a service curve element with known transmission rate. IEEE Networking Letters pp 1–1, DOI 10.1109/LNET.2019.2925176, URL `http://infoscience.epfl.ch/record/267840`

Nikolić B, Yomsi PM, Petters SM (2016) Worst-case communication delay analysis for noc-based many-cores using a limited migrative model. Journal of Signal Processing Systems 84(1):25–46, DOI 10.1007/s11265-015-0992-6

Nikolić B, Tobuschat S, Soares Indrusiak L, Ernst R, Burns A (2018) Real-time analysis of priority-preemptive nocs with arbitrary buffer sizes and router delays. Real-Time Systems DOI 10.1007/s11241-018-9312-0

Le Boudec JY, Thiran P (2001) Network Calculus, LNCS, vol 2050. Springer Verlag, http://lrcwww.epfl.ch/PS_files/NetCal.htm

Papastefanakis E, Li X, George L (2015) Deterministic scheduling in network-on-chip using the trajectory approach. In: Proc. of the IEEE 18th International Symposium on Real-Time Distributed Computing (ISORC 2015), pp 60–65, DOI 10.1109/ISORC.2015.25

Perret Q, Maurère P, Noulard É, Pagetti C, Sainrat P, Triquet B (2016a) Mapping hard real-time applications on many-core processors. In: Proc. of the 24th Int. Conf. on Real-Time Networks and Systems (RTNS 2016), ACM, pp 235–244, DOI 10.1145/2997465.2997496

Perret Q, Maurere P, Noulard E, Pagetti C, Sainrat P, Triquet B (2016b) Temporal isolation of hard real-time applications on many-core processors. In: Real-Time and Embedded Technology and Applications Symposium (RTAS), 2016 IEEE, IEEE, Vienna, Austria, pp 1–11, DOI 10.1109/RTAS.2016.7461363

Qian Y, Lu Z, Dou W (2009a) Analysis of communication delay bounds for network on chips. In: Proc. of the 14th Asia and South Pacific Design Automation Conference (ASP-DAC 2009), Yokohama, Japan, pp 7–12, DOI 10.1109/ASPDAC.2009.4796433

Qian Y, Lu Z, Dou W (2009b) Analysis of worst-case delay bounds for best-effort communication in wormhole networks on chip. In: Proc. of the 3rd ACM/IEEE Int. Symp. on Networks-on-Chip (NoCS 2009), IEEE, San Diego, CA, USA, DOI 10.1109/NOCS.2009.5071444

RealTime-at-Work (2019) RTaW-Pegase home page. URL `https://www.realtimeatwork.com/software/rtaw-pegase/`

Saidi S, Ernst R, Uhrig S, Theiling H, Dupont de Dinechin B (2015) The shift to multicores in real-time and safety-critical systems. In: 2015 Int. Conf. on Hardware/Software Codesign and System Synthesis, CODES+ISSS 2015, Amsterdam, Netherlands, DOI 10.1109/CODESISSS.2015.7331385

Scharbarg JL, Ermont J, Bauer H, Fraboul C (2009) Analyse des délais de bout en bout pire cas dans les réseaux avioniques. Journal européen des systèmes automatisés 43(7-8-9):953–967, URL `http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.699.3977&rep=rep1&type=pdf#page=41`, numéro spécial: actes de la conférence sur la modélisation des systèmes réactifs (MSR'09)

Schmitt J, Zdarsky F, Fidler M (2008a) Delay bounds under arbitrary multiplexing: When network calculus leaves you in the lurch... In: The 27th IEEE Conference on Computer Communications (INFOCOM 2008), pp 1669 –1677, DOI 10.1109/INFOCOM.2008.228

Schmitt JB, Zdarsky FA (2006) The DISCO network calculator - a toolbox for worst case analysis. In: Proc. of the First Int. Conf. on Performance Evaluation Methodologies and Tools (VALUETOOLS'06), ACM, Pisa, Italy, DOI 10.1145/1190095.1190105

Schmitt JB, Zdarsky FA, Martinovic I (2008b) Improving performance bounds in feed-forward networks by paying multiplexing only once. In: Proc. of 14th GI/ITG Conf. on Measuring, Modelling and Evaluation of Computer and Communication Systems (MMB 2008), VDE, pp 1–15

Shi Z, Burns A (2008) Real-time communication analysis for on-chip networks with wormhole switching. In: Proc. of the second ACM/IEEE International Symposium on Networks-on-Chip (NoCS 2008), IEEE, Newcastle upon Tyne, UK, pp 161–170, DOI 10.1109/NOCS.2008.4492735

Tobuschat S, Ernst R (2017) Real-time communication analysis for networks-on-chip with backpressure. In: Proc. of the Design, Automation & Test in Europe Conference & Exhibition (DATE 2017), IEEE, Lausanne, Switzerland, pp 590–595, DOI 10.23919/DATE.2017.7927055

Xiong Q, Wu F, Lu Z, Xie C (2017) Extending real-time analysis for wormhole NoCs. IEEE Transactions on Computers DOI 10.1109/TC.2017.2686391

Zhan J, Stoimenov N, Ouyang J, Thiele L, Narayanan V, Xie Y (2013) Designing energy-efficient noc for real-time embedded systems through slack optimization. In: Proc. of the 50th ACM/EDAC/IEEE Design Automation Conference (DAC 2013), pp 1–6, URL `https://ieeexplore.ieee.org/abstract/document/6560630`

Zhao L, Li Q, Xiong Y, Zheng Z, Xiong H (2013) Using multi-link grouping technique to achieve tight latency in network calculus. In: Proc. of the 32nd IEEE/AIAA Digital Avionics Systems Conference (DASC 2013), East Syracuse, NY, USA, pp 2E3–1–2E3–10, DOI 10.1109/DASC.2013.6712551