Comparing strategies to bound the latencies of the MPPA Network-on-Chipi (Extended version)

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

Comparing strategies to bound the latencies of the MPPA Network-on-Chipi (Extended version)

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-ls-Nancy – France

May 7, 2019

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 for configuring the NoC traffic limiters, in order to 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 latency bounds obtained are then compared on three case studies: two small configurations coming from previous studies, and one 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 packets are of constant size, the Total Flow Analysis gives, on average, bounds 20%-25% smaller than all other methods, since its is the only able, up to now, to accurately and efficiently model these aspects.

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 latency 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 latency, 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 latency bounds.

Getting the best capabilities from such a platform requires efficient methods to compute communication latency bounds. This paper presents and compare several of them, all based on deterministic network calculus. Whereas there exists a large literature on the computation on latency bounds for NoCs, 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 flow limiters (the traffic shapers) and the router switches. Section 3 provides the necessary background on deterministic network calculus. Section 5 introduces notations commons to all the methods presented in this article. Section 6 introduces a new "explicit linear" method for computing the latency 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 compared in Section 8 on three case studies. The two first cases have been already presented in the previous studies Dupont de Dinechin and Graillat (2017), Ayed et al. (2016). It allows to compare the new methods to already published results. Moreover, they are small enough to allow a in-depth interpretation of the results. The last case study is more realistic: each of the 16 clusters sends 4 or 8 independent data flows. Section 8.4 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 cores and 32 management cores on a chip, all implementing the same VLIW core architecture. The MPPA2-256 architecture is clustered with 16 compute clusters and 2 I/O clus-

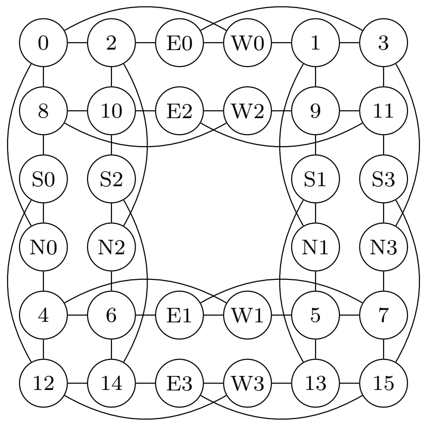


Figure 1: MPPA2 NoC topology unfolded (I/O nodes are labeled N0..N3, E0..E3, S0..S3, W0..W3).

ters, where each cluster is built around a multi-banked local static memory shared by 16+1 (compute cluster) or 4+4 (I/O cluster) processing + management cores. The clusters communicate through a NoC, with one node per compute cluster and 8 nodes per I/O cluster. More details can be found in Saidi et al. (2015).

The MPPA2 NoC is a direct network based on a 2D-torus topology extended with extra links connected to the otherwise unused ports of the NoC nodes on the I/O clusters (see Figure 1).

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.

The motivation for implementing wormhole switching with source routing and without virtual channels is the reduction of complexity of the network elements and interfaces while still supporting services guarantees. However, once a buffer is full, the flow control mechanism of wormhole switching requires that the previous router store 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. 3). 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

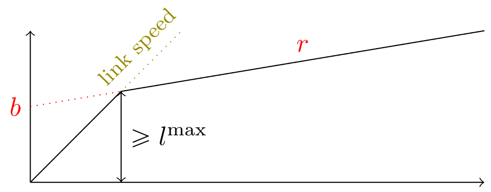


Figure 2: Tocken-bucket traffic limiter.

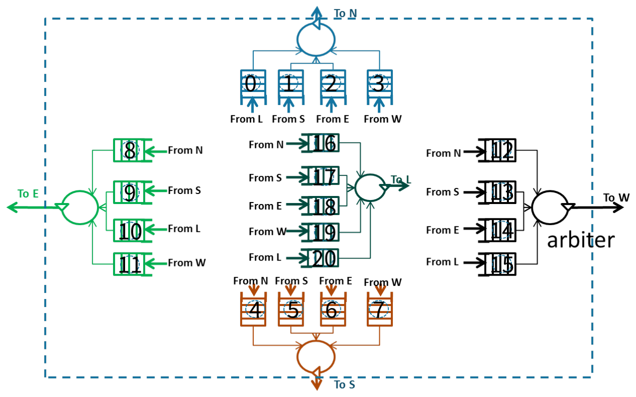


Figure 3: Structure of a MPPA2 NoC router.

in Figure 2 – the exact relation between r, b and the packet size will be given in eq. (21)). 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 that the router queues be full. This can be done by: 1) defining a static set of data flows; 2) allocating to each flow a 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 traffic limiters configuration.

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 macro networks Cruz (1991).

The following is a short summary of the deterministic network calculus the-

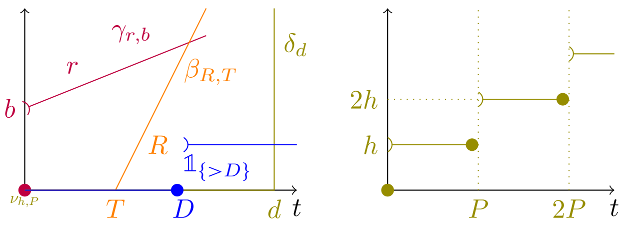


Figure 4: Common curves in network calculus.

ory, 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

The network calculus mainly uses functions from time domain, $\mathbb{R}^+$, to data amount $\mathbb{R}^+$. Let $\mathcal{F}$ denote the set of such functions, and $\mathcal{F}^{\uparrow}$ the subset of non-decreasing functions: $\mathcal{F}^{\uparrow} \overset{def}{=} \{f \in \mathcal{F} \mid \forall t, d \in \mathbb{R}^+ : f(t+d) \geqslant f(t)\}$.

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

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:

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

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

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 $\nu_{h,P}(t) = h \left\lceil \frac{t}{P} \right\rceil$, where $\lceil \cdot \rceil$ is the ceiling function.

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.

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

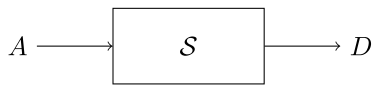


Figure 5: A server crossed by a flow.

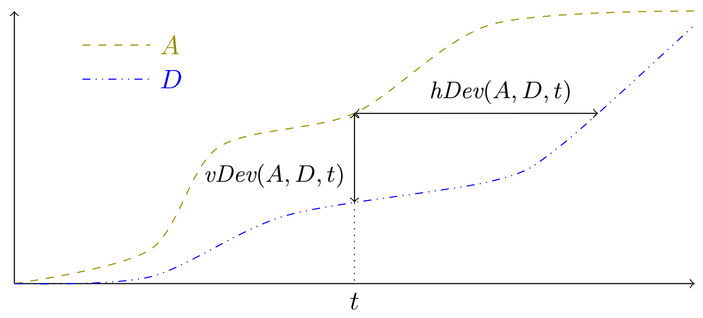


Figure 6: Delay and backlog between arrival and departure flows.

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 $(A, D) \in \mathcal{S}$.

The *delay* and *backlog* associated to a server are defined from the arrival A and departure D cumulative functions. The *delay* at time t is defined as $hDev(A, D, t)$, and the *backlog* at time t is $vDev(A, D, t)$,

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

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

The semantics of the backlog is quite obvious: it is the amount of data held by the server. The one of the delay deserves an explanation: for a bit arrived at time t, it is the duration required for the accumulated departure curve to reach the same amount of data.

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

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

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

In general, a server is shared by several flows, but as will be presented further, a main step in network calculus consists in reducing a server shared by several flows into an "equivalent" server crossed by a single flow.

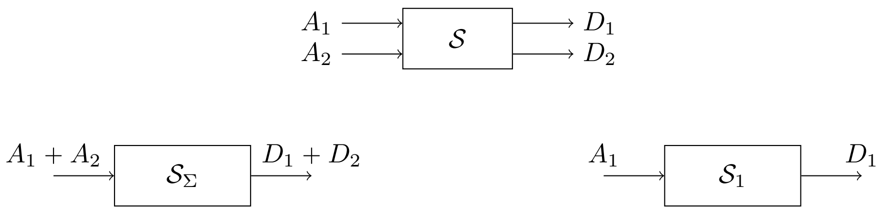


Figure 7: A 2-server, its aggregate server and one residual server.

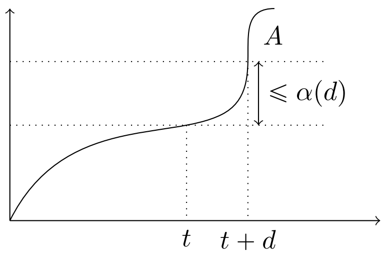


Figure 8: Arrival curve.

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

Given a n-server, its *aggregate server* $\mathcal{S}_\Sigma$ is defined as $A \xrightarrow{\mathcal{S}_\Sigma} D$ if is exists $(A_1, \dots, A_n) \xrightarrow{\mathcal{S}} (D_1, \dots, D_n)$ such that $A = \sum_{i=1}^n A_i$, $D = \sum_{i=1}^n D_i$. And for any $i \in [1,n]$, its *residual server* $\mathcal{S}_i$ is defined by $A_i \xrightarrow{\mathcal{S}_i} D_i$ if it exists $(A_1, \dots, A_n) \xrightarrow{\mathcal{S}} (D_1, \dots, D_n)$.

3.3 Contracts

The exact behavior of a data flow or a server is commonly unknown at design time, or too complex. Then, the performance analysis is made using contracts: the maximal load generated by a flow, and the minimal capacity of a server.

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

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

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.

There exists two contracts on the minimal capacity of a server: 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 curve.

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. Then 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*

$$hDev(A, D) \leqslant hDev(\alpha, \beta), \tag{10}$$
$$vDev(A, D) \leqslant vDev(\alpha, \beta), \tag{11}$$

and D has α' as arrival curve, with

$$\alpha' = (\alpha \oslash \beta) \wedge \sigma. \tag{12}$$

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 let A a cumulative function crossing both in sequence (*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].

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

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*

$$\beta_i^{blind} = \left[\beta - \sum_{j \neq i} \alpha_j\right]_\uparrow^+ . \tag{13}$$

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 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*

$$\beta_i^{g\text{-}FIFO} = \delta_d \text{ with } d = hDev\left(\sum_{j=1}^{n} \alpha_j, \beta\right), \tag{14}$$

$$\beta_i^{\theta-FIFO} = \left[\beta - \sum_{j \neq i} \alpha_j * \delta_\theta\right]^+ \wedge \delta_\theta, \forall \theta \in \mathbb{R}^+. \tag{15}$$

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 $\beta_i^{\theta-\text{FIFO}}$. 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 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 $b_i' = b_i + r_i T_i$.*

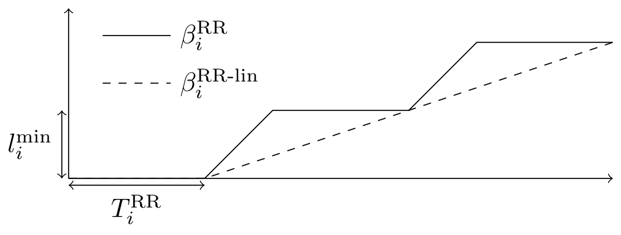


Figure 9: Illustration of WRR residual service, with $\beta(t) = Rt$.

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

$$\beta_i^{RR} = \left(\lambda_1 * \nu_{l_i^{\min}, l_i^{\min} + L_{\neq i}^{\max}}\right) \circ \left(\beta - L_{\neq i}^{\max}\right), \tag{16}$$

$$\beta_i^{RR\text{-}lin} = \frac{l_i^{\min}}{l_i^{\min} + L_{\neq i}^{\max}} \left[\beta - L_{\neq i}^{\max}\right]^+ \tag{17}$$

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

$$R_i^{RR} = R \frac{l_i^{\min}}{l_i^{\min} + L_{\neq i}^{\max}}, \qquad T_i^{RR} = \frac{L_{\neq i}^{\max}}{R}. \tag{18}$$

This theorem gives three expressions of residual services, but in fact there is only one, since $\beta_i^{\mathrm{RR\text{-}lin}}$ is just a linear lower bound of β_i^{RR}, and $\beta_{R_i^{\mathrm{RR}}, T_i^{\mathrm{RR}}}$ the expression of $\beta_i^{\mathrm{RR\text{-}lin}}$ when the aggregate service is a constant rate. Their relation is illustrated on Figure 9. The proof can be found in (Bouillard et al., 2018, Thm. 8.6).

3.5 Analysis principles

3.5.1 Local 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 10, 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 configuration (on the middle left), with strict service of curve β_{123}, one may

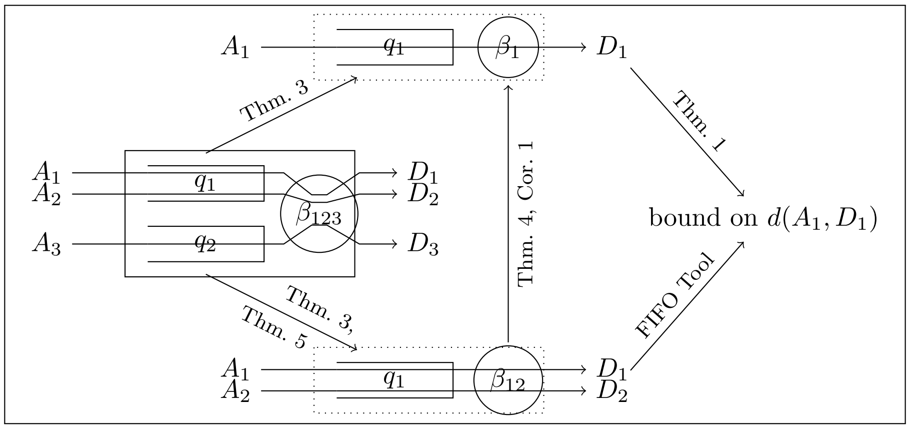


Figure 10: Decomposition of output port in residual servers.

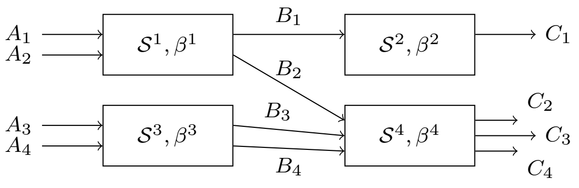


Figure 11: Simple topology.

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 the expressions of the residual curves are different for each theorem, the choice of one or the other will give a different residual curve, and then different bounds 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 then 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 Global analysis

There exist several ways to bound the end-to-end delay of a given flow. 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, 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

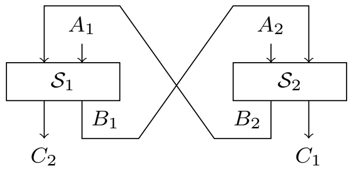


Figure 12: Cyclic dependencies.

in details in Section 7.1. In the topology of Figure 11, 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 $\beta_i^{\text{SFA}} = \mathop{\ast}_{f_i \in F^j} \beta_i^j$ that allows computing $d^{\text{SFA}} = hDev(\alpha_i, \beta_i^{\text{SFA}})$ a bound on the end-to-end delay. In the topology of Figure 11, to bound the delay of f_4, SFA will compute β_4^3 (resp. β_4^3), 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. And none of the algorithm specifies how to compute the arrival curves of the interfering flows (the arrival curves of B_2 and B_3).

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.

A complete different approach has been developed in Bouillard et al. (2010): assuming that all arrival (resp. service) curves are piece-wise linear concave (resp. convex) functions, instead of computing a residual service, all network behaviors are encoded as one mixed-integer linear program.

3.6 Cyclic dependencies

Last, let us illustrate why cyclic dependencies are still an open problem in network calculus.

Consider the topology of Figure 12, and first assume a blind policy. To compute the delay of the flow A_1 in $\mathcal{S}_1$, one may use Theorem 3, but then, the arrival curve of B_2 is required. And to compute this arrival curve, one may use Theorems 1 and 3, but the arrival curve of B_1 is required.

The same applies if $\mathcal{S}_1$ and $\mathcal{S}_2$ apply a FIFO policy. An overview of managing cyclic dependencies can be found at Chapter 12 in Bouillard et al. (2018).

But if $\mathcal{S}_1$ or $\mathcal{S}_2$ uses a round robin policy, alternating service of packets for flows A and B, the problem does not occur anymore since computing the

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

residual service does not require the arrival curve of the competing flows.

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 Karlay 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.

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 reducing the number of conflicts in buffer use and so the number of activations of the back-pressure mechanism.

For example, an active community considers NoC with wormhole switching, in each routers, 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 method that fully takes into account the back-pressure mechanism of the wormhole switching. It has been adapted to the Karlay MPPA NoC in Ayed et al. (2016) and compared with a network-calculus based approach Dupont de Dinechin et al. (2014) on an example, that will also be considered in this article (cf. Section 8.2). 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 latency. In Tobuschat and Ernst (2017), the authors develop a CPA-based method to compute the latency 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, one study takes into account the back-pressure withing network calculus framework, and it is presented in the next section.

Network calculus 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. So, we are going to present first pure network-calculus studies on Weighted Round Robin (WRR), FIFO policy and thereafter their application to NoCs.

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), that also gives a residual service. 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.

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. The choice of a good set of parameters was the core work of the DEBORAH tool, presented in Bisti et al. (2010), Lenzini et al. (2004), Lenzini et al. (2005), Lenzini et al. (2007) and downloadable at Bisti et al. (2011). Since this work 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 compute either optimal delay or arrival curve, without any explicit expression of the θ parameters.

A new approach have been developed in Bouillard et al. (2010): instead of locally computing a residual service, the basic equations of network calculus are encoded as a mixed-integer linear program (MILP), looking only at a set of well defined time variables. Moreover, it does not compute an upper bound on the delay, but *the worst* delay, also know as worst case delay. It has thereafter been adapted to FIFO multiplexing, in Le Boudec and Thiran (2001), Chang (2000), 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, 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”.

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 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).

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 Karlay 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 Karlay 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 13, 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 14 reports a subset of the queues involved in example of Figure 13: 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.

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 $F^6 = \{f_1, f_2\}$.

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

$\{q^1, \dots, q^{n_q}\}$	set of queues
$\{p^1, \dots, p^{n_p}\}$	set of ports
$p(q^i) = p^k$	q^i is an input queue of p^j
$\{f_1, \dots, f_{n_f}\}$	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.

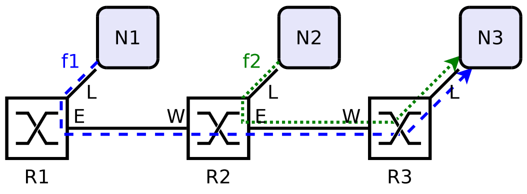


Figure 13: Network elements and flows example to illustrate notations/

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

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

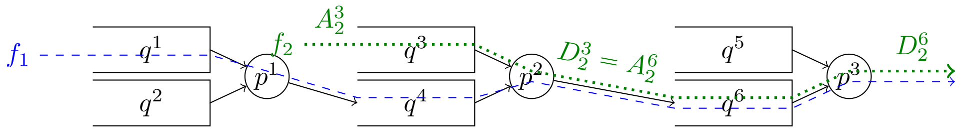


Figure 14: Partial translation of example of Figure 13.

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 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 latencies can be formulated as a Mixed-Integer Linear Problem (MILP). Indeed, by assuming that the flow rates are constant, is is possible to evaluate the delays and backlogs as variables of a linear problem[6].

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 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.

6.1 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

$$r^j = \sum_{f_i \in F^j} r_i, \qquad\qquad b^j = \sum_{f_i \in F^j} b_i^j. \tag{19}$$

But this aggregate flow comes from a link of peak rate r. Then, it also have 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 15).

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

[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 routing is fixed and the MILP variables are specific instants and the values of the cumulative functions at these instants, cf. Bouillard and Stea (2014).

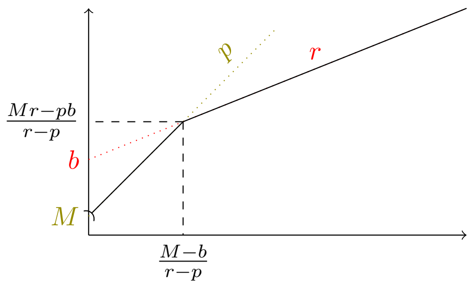


Figure 15: T-SPEC flow caracteristic curve.

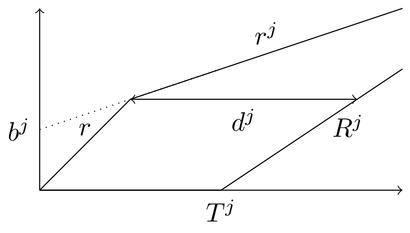


Figure 16: Delay between shaped token-bucket and rate-latency service: $d^j = hDev(\lambda_r \wedge \gamma_{r^j,b^j}, \beta_{R^j,T^j})$.

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 16, application of the T-SPEC arrival curve on such service curve yields

$$d^j = T^j + \frac{b^j(r - R^j)}{R^j(r - r^j)}. \tag{20}$$

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

$$\forall f_i \in F : b_i \geqslant b_i^{\min} \overset{def}{=} l_i^{\max} \frac{r - r_i}{r}. \tag{21}$$

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

$$b_i^j = b_i^k + r_i T^k. \tag{22}$$

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.*

$$r_{\neq i}^j = \sum_{f_l \in F^j, l \neq i} r_l, \qquad b_{\neq i}^j = \sum_{f_l \in F^j, l \neq i} b_l^j. \tag{23}$$

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 $u \leqslant \frac{b_{\neq i}}{r - r^j_{\neq i}}$ and to the second otherwise. Then

$$\sup_{u \geqslant 0} \alpha_2(u) + r_i u - R^j \tag{24}$$

$$= \left(\sup_{0 \leqslant u \leqslant \frac{b_{\neq i}}{r - r^j_{\neq i}}} (r + r_i - R^j) u \right) \vee \left(\sup_{u \geqslant \frac{b_{\neq i}}{r - r^j_{\neq i}}} (r^j_{\neq i} + r_i - R) u + b_{\neq i} \right) \tag{25}$$

$$= b_{\neq i} \frac{r + r_i - R^j}{r - r^j_{\neq i}}. \tag{26}$$

Application of Prop. 1 leads to

$$b^j_i = b^k_i + r_i \left(T^j + \frac{b^j_{\neq i}(r + r_i - R^j)}{R^j(r - r^j_{\neq i})} \right). \tag{27}$$

Note that the use of Cor. 1 would lead to $b^k_i + r_i \left(T^j + \frac{b^j_{\neq i}}{R^j} \right)$ 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 $l^{\min}_i$ and a maximum $l^{\max}_i$. Let $l^{\min}_{F^j} \overset{def}{=} \min_{f_i \in F^j} l^{\min}_i$ and $l^{\max}_{F^j} \overset{def}{=} \max_{f_i \in F^j} l^{\max}_i$ 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 $Q^{\neq j} \overset{def}{=} \left\{ q^k \mid p(q^k) = p(q^j), k \neq j \right\}$ be the set of queues sharing the same arbiter than q^j. By application of eq. (18), the general round-robin service curve $\beta^j = \beta_{R^j,T^j}$ for q^j is

$$R^j = \frac{r l^{\min}_{F^j}}{l^{\min}_{F^j} + \sum_{k \in Q^{\neq j}} l^{\max}_{F^k}}, \qquad T^j = \frac{\sum_{k \in Q^{\neq j}} l^{\max}_{F^k}}{r}. \tag{28}$$

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

$$R^j = r - \sum_{k \in Q^{\neq j}} r^k, \qquad T^j = \frac{\sum_{k \in Q^{\neq j}} b^k}{r - \sum_{k \in Q^{\neq j}} r^k}. \tag{29}$$

The blind multiplexing service curve must be used whenever the sum of flow rates inside q^j exceeds R^j in Eq. (28). 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 latency 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 latency 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 (28) or (29). 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:

$$R_i^j = R^j - r_{\neq i}^j F^j, \qquad T_i^j = T^j + \frac{b_{\neq i}^j}{R^j}. \tag{30}$$

For the second step, we compute the convolution $\beta_i^* = \ast_{q^j \in Q_i} \beta_i^j$ 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:

$$R_i^* = \min_{j \in Q_i} R_i^j, \qquad T_i^* = \sum_{j \in Q_i} T_i^j. \tag{31}$$

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. (20):

$$d_i^* = T_i^* + \frac{b_i(r - R_i^*)}{R_i^*(r - r_i)}. \tag{32}$$

7 Adaptation of generic algorithms to the MPPA NoC

Section 6 has presented a modeling of the MPPA NoC using linear constraints for respecting deadline and buffer 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 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 to compute 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. (12) and eq. (14)).

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

$$\alpha^j = \lambda_r \wedge \sum_{q^k \in I^j} \sum_{f_i \in F^k \cap F^j} \dot{\alpha}_i^k. \tag{33}$$

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 to compute the arrival curve of the competing flows[8]. It can be of interest when a queue shares the output

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

[8] This can be done using eq. (33). 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.

$$\begin{tikzcd}
 & & \beta_i^{j,\text{Blind}} \\
 & & \beta^{j,\text{RR/g-FIFO}} \\
\beta \arrow[uurr, "\text{Thm. 3}"] \arrow[r, "\text{Thm. 5}"] \arrow[dddr, "\text{Thm. 3}"'] & \beta^{j,\text{RR}} \arrow[ur, "\text{eq. (14)}"] \arrow[r, "\text{eq. (15)}"] \arrow[dr, "\text{Cor. 1}"'] & \beta_i^{j,\text{RR}/\theta\text{-FIFO}} \\
 & & \beta_i^{j,\text{RR}/\beta\text{-FIFO}} \\
 & & \beta_i^{j,\text{Blind/g-FIFO}} \\
 & \beta^{j,\text{Blind}} \arrow[ur, "\text{eq. (14)}"] \arrow[r, "\text{eq. (15)}"] \arrow[dr, "\text{Cor. 1}"'] & \beta_i^{j,\text{Blind}/\theta\text{-FIFO}} \\
 & & \beta_i^{j,\text{Blind}/\beta\text{-FIFO}}
\end{tikzcd}$$

Figure 17: Different ways to compute a residual service.

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 $d^j = hDev(\alpha^j, \beta^j_{\text{Blind}}) \wedge hDev(\alpha^j, \beta^j_{\text{RR}})$.

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 $\beta_i^{\text{SFA}} = \ast_{q^j \in Q_i} \beta_i^j$, 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. (15) or eq. (14) or the Cor. 1). Combining all possibilities leads to 7 different expressions, as presented in Figure 17. 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.

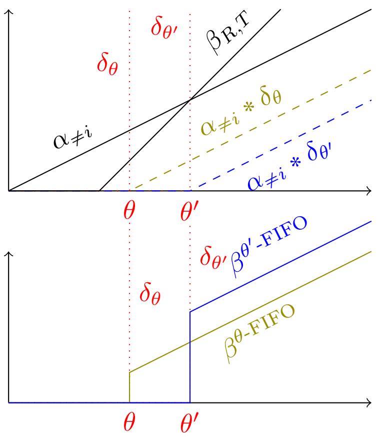


Figure 18: Residual FIFO service with to θ, θ' parameters, $\alpha_{\neq i} = \sum_{j \neq i} \alpha_j$.

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

One may want to compute both $\beta^{j,\mathrm{RR}}$ and $\beta^{j,\mathrm{Blind}}$ 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 $\beta^{j,\mathrm{RR}}$ and $\beta^{j,\mathrm{Blind}}$, 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

$$\beta_i^{\theta\text{-FIFO}} = [\beta - \alpha_{\neq j} * \delta_\theta]^+ \wedge \delta_\theta, \tag{34}$$

with $\alpha_{\neq i} = \sum_{j \neq i} \alpha_j$. To the best of our knowledge, there is no general result on the best, neither any good, θ parameter. The works presented in the state of the art consider only affine or piece-wise linear concave/convex functions, and do not give any explicit expression of this θ parameter.

Nevertheless, 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].

[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.

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 $\beta - \alpha_{\neq j} * \delta_\theta < \beta - \alpha_{\neq j} * \delta_{\theta'}$. 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 18).

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 $\theta < T_\beta$, $\beta \wedge \delta_\theta = \beta$, leading to $\beta^{\theta\text{-FIFO}} = [\beta - \alpha_{\neq j} * \delta_\theta]^+$. So, considering $\theta < \theta' < T_\beta$, $\beta^{\theta\text{-FIFO}} < \beta^{\theta'\text{-FIFO}}$, 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.

To sum up, the value 0 reduces FIFO to blind multiplexing, the values in $[0, T_\beta)$ are worst than T_β and the value $hDev(\sum_{i=1}^n \alpha_i, \beta)$ gives the same result than TFA. So, in this study, the value $\theta = T_\beta$ will be considered. The definition of a strategy for computing a better parameter value is out of the scope of this study.

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. 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 2). It belongs to the class of concave piecewise-linear function (CPL). Conversely, the residual service of a round robin arbiter given by eq. 18 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 has been compared on one example (that will be reused in

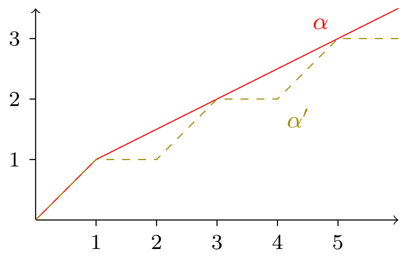


Figure 19: Effect of constant packet size on arrival curve, Thm. 6, with $\alpha = \gamma_{1/2,1/2} \wedge \lambda_1$, $l = 1$.

Section 8.2).

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. 16. 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 $\alpha = \lambda_R \wedge \gamma_{r,b}$ (as in Figure 19).

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 20.

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

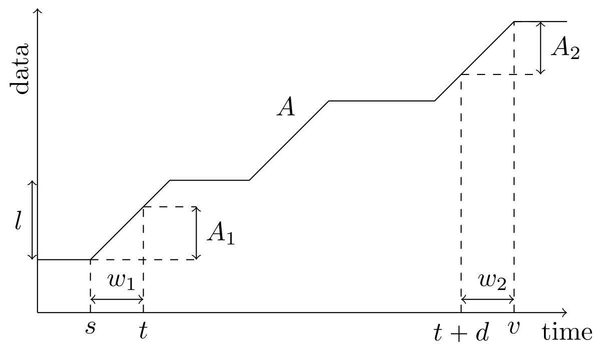


Figure 20: Per packet cumulative function.

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

$$A(t + d) - A(t) + R(w_1 + w_2) \leqslant l \left\lfloor \frac{\alpha(v - s)}{l} \right\rfloor \tag{35}$$

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

$$A(t + d) - A(t) \leqslant l \left\lfloor \frac{\alpha(d - w)}{l} \right\rfloor - Rw \tag{36}$$

$$\leqslant \sup_{w \geqslant 0} l \left\lfloor \frac{\alpha(d - w)}{l} \right\rfloor - \lambda_R(w) \tag{37}$$

$$= \left(l \left\lfloor \frac{\alpha}{l} \right\rfloor \oslash \lambda_R \right)(d) \tag{38}$$

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 $d' \leqslant d$

$$A(t + d) - A(t) \leqslant \left(l \left\lfloor \frac{\alpha}{l} \right\rfloor \oslash \lambda_R \right)(d). \tag{39}$$

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

Table 2: The analysis strategies.

□

8 Comparing strategies

Several approaches and models have been presented in the previous sections. They will be compared on several case studies, with increasing size to ease interpretation of the results.

The approaches have been partitioned in two categories: a first one computing a end-to-end delay, and a second one computing local per queue 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. So, LP is used instead of SFA for the affine model. The modeling of per flow constant packet sizes (use of Theorem 6) lead to non concave arrival curves, and in this case, we use SFA for the end-to-end delay computation (SFA/Fc). Moreover, the model considers that all packets in a queue have the same size (use of Theorem 5); this is method 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, with variation on three parameters, the maximal packet size, the load, and the number of flows per queue.

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, first example (topology of Figure 21), first experiment (original values).

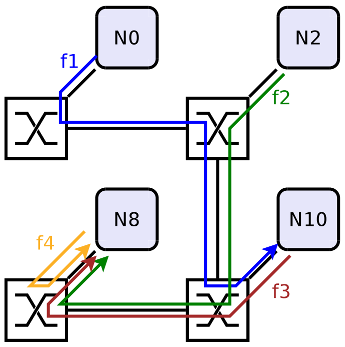


Figure 21: Case study from Dupont de Dinechin and Graillat (2017), 4 nodes.

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). The results on the affine Total Flow Analysis have been obtained using the RTaW-Pegase tool, from at Work (2019). All other results have been obtained by a prototype plugin to the RTaW-Pegase tool.

8.1 First example: 4 nodes

The first example is from Dupont de Dinechin and Graillat (2017). It has 4 nodes, generating 4 flows crossing 4 routers, with routing depicted in Figure 21.

8.1.1 First experiment, original values

In a first experiment, all flows have a packet size of 17 flits (considered as typical), all flows have a long-term rate $\frac{1}{3}$ but f_1 that have $r_1 = \frac{2}{3}$. The admissible bursts at network ingress are $\frac{34}{3}$ but f_1 that have $b_1 = \frac{17}{3}$ (cf. Table 3).

The upper bounds on delays for this example are displayed in Figure 22. 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. The interpretation of the TFA method is also simple: whereas TFA is a perhaps the simplest approach, it is the one that captures in the

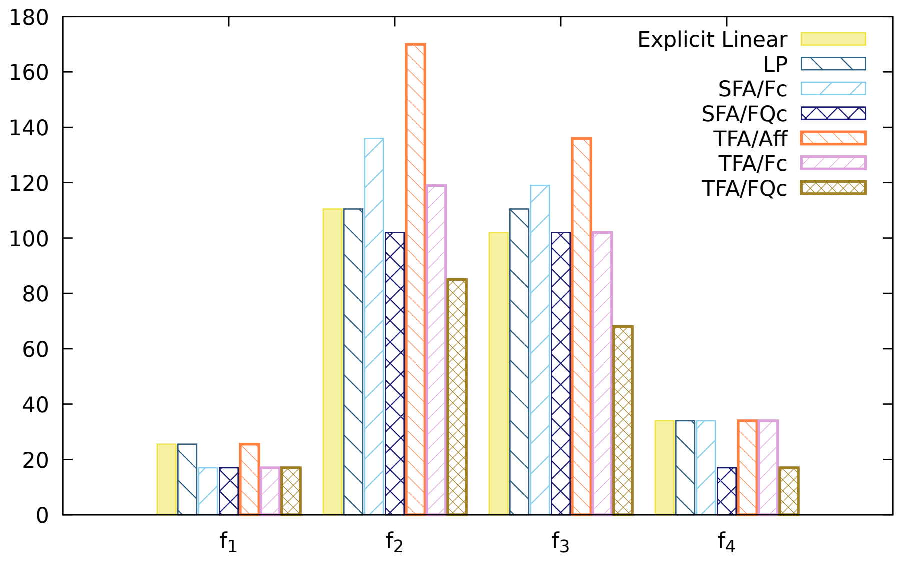


Figure 22: Upper bounds on delay, per flow and per method, first example (topology from Figure 21), first experiment (original values, parameters from Table 3).

most efficient way the packetisation effect of data flows. Whereas the fluid TFA perfoms worst for all flows, TFA with constant packet size per flow give results comparable to other methods, and if moreover all packets in each queue have the same size, it gives the best results.

The end-to-end computation deserve a discussion: whereas the LP methods 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 exist. The same happens when considering packets of fixed sizes in SFA: with more assumptions, and considering non concave/convex piecewise-linear functions (Figures 9, 19), the bounds are better, even if the core of the resolution method is worse.

8.1.2 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, first example (topology of Figure 21), second experiment (splitting flows).

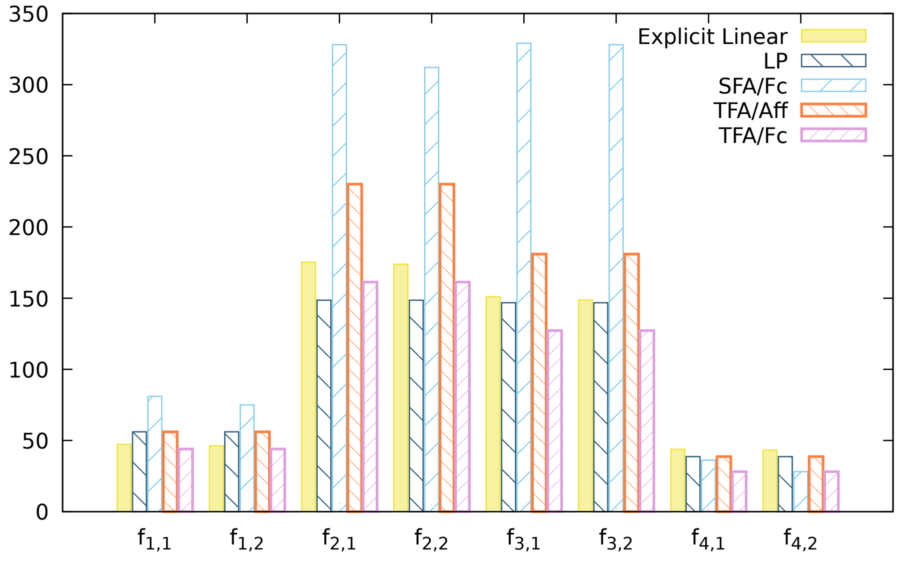


Figure 23: Upper bounds on delay, per flow and per method, first example (topology from Figure 21), second experiment (splitting flows, parameters from Table 4).

Table 4. Note that splitting a flow increases the initial burst[10]. 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 2 and eq. (21)).

The results are reported in Figure 23. The results are comparable to those of the previous experiment. The explicit linear method does not give the best results, but nevertheless give good bounds. The LP method is in general better than the other affine methods (explicit linear and fluid TFA), and even gives the best results for flows $f_{2,1}, f_{2,2}$. For all other flows, if all packets of a given flow have the same size, the per flow constant size TFA can model it and gives the bests results.

[10]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 $b(f_{i,1})+b(f_{i,2}) > b(f_i)$

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.	7O	70	70	70
Burst	$\frac{70}{3}$	$\frac{140}{3}$	$\frac{140}{3}$	$\frac{140}{3}$

Table 5: Flow parameters, first example (topology of Figure 21), third experiment (large packet size).

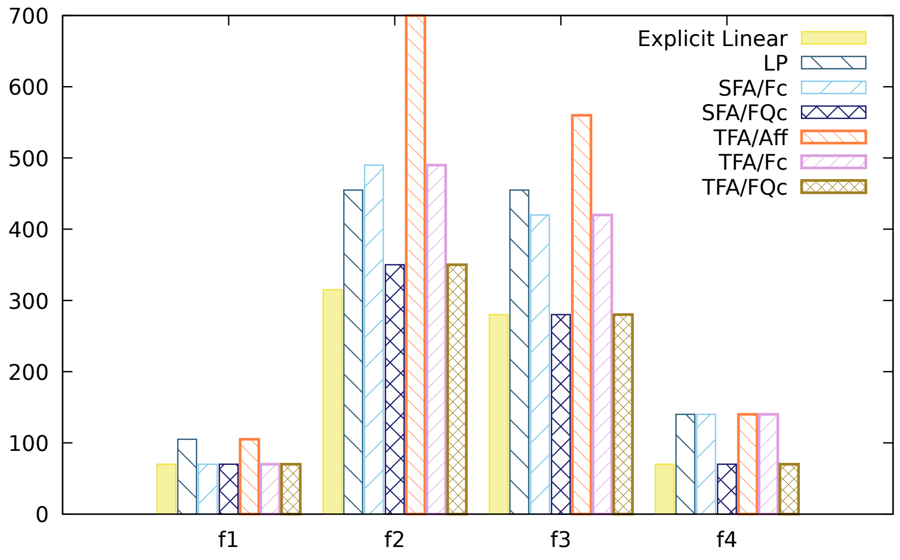


Figure 24: Upper bounds on delay, per flow and per method, first example (topology from Figure 21), third experiment (large packets, parameters from Table 5).

8.1.3 Third experiment, large packet size

The third experiment uses the same parameters as the initial experiment (Section 8.1.3), but with large packet size (70 flits). The flow parameters are given in Table 5, and the results are reported in Figure 24.

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 22), it is [0,700] in Figure 24. Since the packet and burst sizes are $\frac{70}{17} \approx 4.11$ larger, the delay also are globally four times larger. The main difference is that the explicit linear method is now the best. The reason is that this method uses Proposition 1, that captures in a very efficient way the limited impact of FIFO policy on burst sizes. Its increase of delay is only around 2.5.

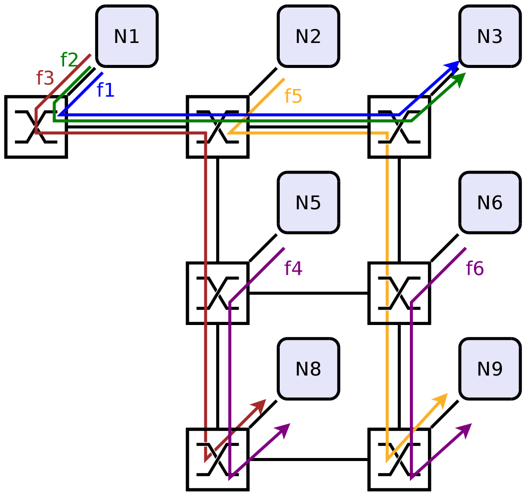


Figure 25: Case study from Ayed et al. (2016), 7 nodes.

8.2 Second Example: 7 nodes

The second example comes from Ayed et al. (2016). It is made of 6 flows, $f_1, \ldots, f_6$. The routing is given in Figure 25. This case study has been used to illustrate the "Recursive Calculus", a method developed in Ayed et al. (2016).

8.2.1 First experiment, original parameters

The first experiment uses the values of the parameters used in Ayed et al. (2016): all packets have a constant size of 50 flits, and all flows have a period of 1000 cycles. Furthermore all flows have a rate of 0.05 and a burst of 47.5.

On this example with very small loads (from 5% to 15%), the burst is the parameter that have the main influence. The results are reported in Figure 26. The bounds of the "Recursive Calculus" from Ayed et al. (2016) have been reported[11]. In this example, the TFA approach outperforms all others, except for flow f_5. This is mainly because this flow is very long, with very few interference. The recursive calculus, that has been designed to analyze both Tilera and MPPA NoCs, with possible back-pressure activation, gives results comparable to the SFA approach (flows f_2, f_4, f_5, f_6) or to the explicit linear model (flows f_1, f_3).

Since the different methods gives very different values in this example, it well illustrates some of their differences.

Consider the flow f_1. The explicit linear method computes a delay d_1^* of 206 cycles, decomposed into a latency of $T_1^* = 145$ and a "burst absorption

[11]Note that the values presented here are not exactly the same as in Ayed et al. (2016): as far as we understand, in Ayed et al. (2016) the delays include the arbitration in the node, whereas the methods presented here only consider the NoC delays. Then, the node arbitration delays have been removed for flows f_1, f_2 and f_3.

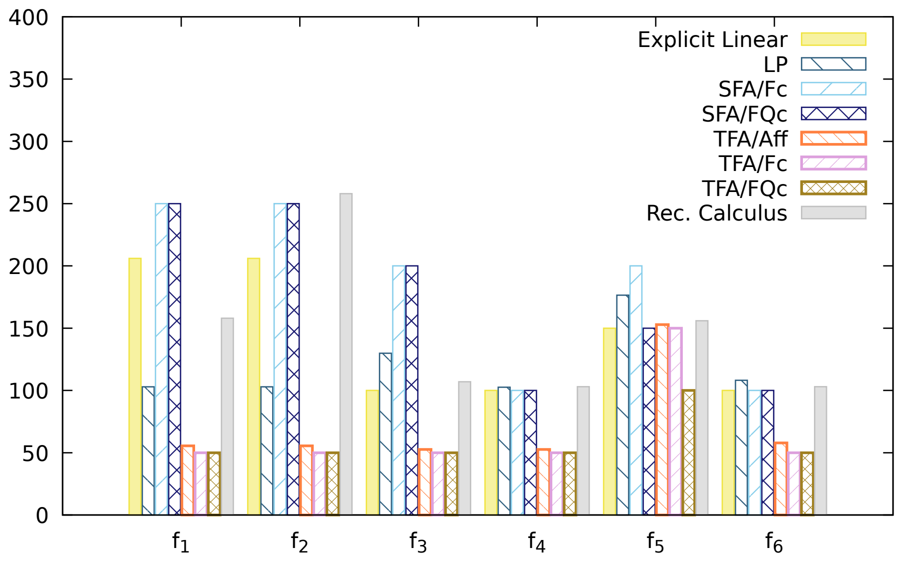


Figure 26: Upper bounds on delay, per flow and per method, second example (topology from Figure 25), first experiment (original parameters).

time of" 61, cf. eq (32). This latency is due to the traversal of one round-robin arbitration (in router R2), whereas the burst related term is related to the FIFO sharing of the queue in routers R1 and R3. The LP formulation also counts a latency of 50 due to arbitration in router R2 and reduces the FIFO interference all along the path to 50. The TFA approach computes a per router delay. But it computes a null delay in routers R1 and R3: indeed, in R1, the three flows f_1,f_2 and f_3 are shaped (*i.e.* serialized) at the input link, then since the router uses a cut-through forwarding, there is a null delay[12]. Then, the only delay is related to the arbitration in output port of R2.

Note that small delay in R1 is true, but related to the fact that the contention between the flows has been resolved in the node N1 itself.

8.2.2 Second experiment, realistic load

The second experiment considers the same routing than the previous one, but with maximal packet size of 17 and rates computed in order to ensure fairness and efficient link utilization (all parameters are presented in Table 6).

The delay bounds are plotted in Figure 27. The first observation is that even if the rates are quite 10 times bigger, the delay bounds are about 2 times smaller. This is due to packet size reduction, since the packet size influences both the flow burst size and the latency of the round-robin arbiter.

[12]In fact, it exists some cycles related to the routing and the computation, but since this delay is small and constant, it has not be modeled in any method.

Flow	f_1	f_2	f_3	f_4	f_5	f_6
Rate	1/3	1/3	1/3	2/3	1/3	2/3
Max. Packet Size.	17	17	17	17	17	17
Burst	34/3	34/3	34/3	17/3	34/3	17/3

Table 6: Flow parameters, second example (topology of Figure 25), second experiment (realistic configuration).

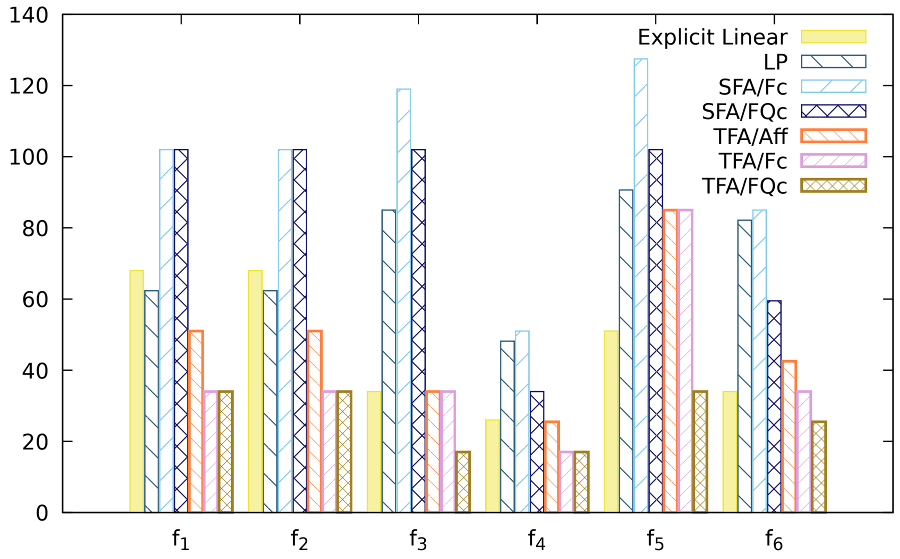


Figure 27: Upper bounds on delay, per flow and per method, second example (topology from Figure 25), second experiment (realistic configuration, parameters from Table 6).

Flow	f_1	f_2	f_3	f_4	f_5	f_6
Rate	1/3	1/3	1/3	2/3	1/3	2/3
Max. Packet Size.	50	50	50	50	50	50
Burst	100/3	100/3	100/3	50/3	100/3	50/3

Table 7: Flow parameters, second example (topology of Figure 25), third experiment (loaded configuration).

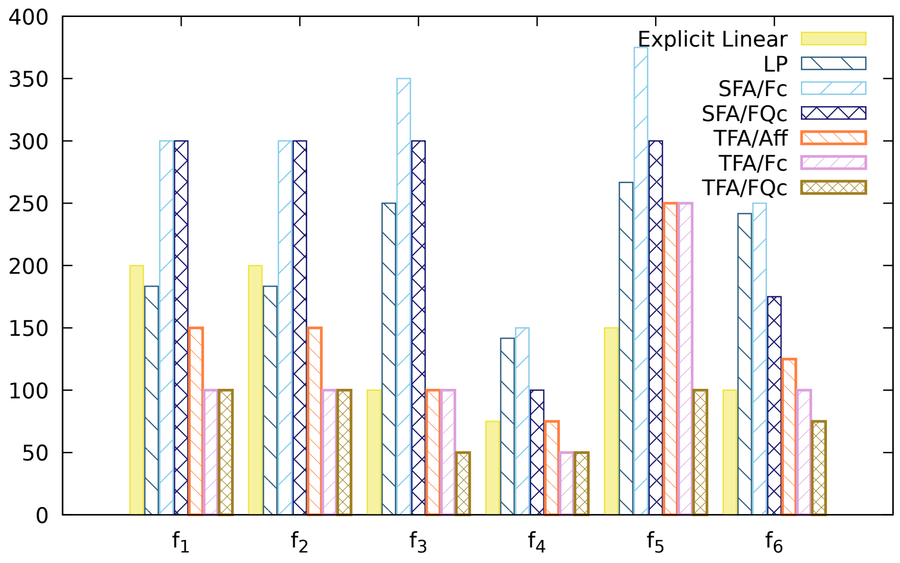


Figure 28: Upper bounds on delay, per flow and per method, second example (topology of Figure 25), third experiment (loaded configuration, parameters from Table 7).

Like for other case studies, even for the fluid model, there is no best solution: depending on the flow, the best bound is given either by explicit linear method (f_5, f_6) or the affine TFA (f_1, f_2). The LP is never the best, meaning that shaping has a strong influence on this case study. And if all packets have the same size, the TFA approaches gives the best results.

8.2.3 Third experiment, loaded configuration

The third experiment considers the large packet size of the first experiment (50 flits) and the flow rates of the second (cf. Table 7). The results appear in Figure 28, with the same scale as in Figure 26.

Looking at load change (*i.e.* comparing with the first experiment, where maximal packet size is the same, but the load is smaller), the impact is very different on each method. The explicit linear method computes quite the same value in both experiments. On this example, the explicit linear method is mainly

Flows ($k \in \{1,2\}$)	$f_{1,k}$	$f_{2,k}$	$f_{3,k}$	$f_{4,k}$	$f_{5,k}$	$f_{6,k}$
Rate	1/6	1/6	1/6	1/3	1/6	1/3
Max. Packet Size.	25	25	25	25	25	25
Burst	125/6	125/6	125/6	50/3	125/6	50/3

Table 8: Flow parameters, second example (topology of Figure 25), fourth experiment (loaded configuration, doubling number of flows).

influenced by the packet size and very few by the load. The other methods are more influenced by this change of load, and this changes the relative quality of the different methods.

Looking at maximal packet size change, (*i.e.* comparing with the second experiment, where maximal packet size is 17 instead of 50, but the load is the same) a remarkable effect appears: the ratio between the bounds computed in both experiments is exactly $\frac{50}{17} \pm 1\%$, for each methods and each flow. This is due to the fact that both the arbiter latency (round-robin) and the burst size are proportional to this maximal packet size (all other parameters being unchanged).

8.2.4 Fourth experiment, loaded configuration, doubling number of flows

The fourth experiment is based on the third one, where each flow f_i is split into two flows $f_{i,1}$, $f_{i,2}$ with the maximal packet size and the throughput divided by two, as shown in Table 8.

It means that this experiment has somehow the same global load (except the burst that are slightly higher), but there are two times more flows per queue. Moreover, since the maximal packet size is smaller, the blocking time associated to the round-robin arbiter is also smaller, even if the long-term rate is the same. Both effects have opposite impact in the delay. On the one hand, the increase in the number of flows, and the associated small increase of the burst both lead to a per flow delay increase. On the other hand, the reduction of the round-robin blocking time decreases the per flow delay.

Then, for a given method, the difference in the result between this experiment and the third one depends on how the method handle the FIFO policy (inside each queue) and the round-robin policy (between queues).

The results are displayed in Figure 29 (except the values of methods SFA/Fc and SFA/FQc, which is equal to 750 cycles).

As in previous experiments, in case of a fluid model, the explicit linear, LP and TFA methods give comparable results (within a factor of 2), but none dominates the others. But in case of packets of constant size, per flow or also per queue, only the TFA approach captures this effect and always give the better bounds. Since the TFA approach computes an aggregate arrival curve by summing all arrival curves in a queue, it is insensitive to the number of flows, as long as the total load and burst is the same.

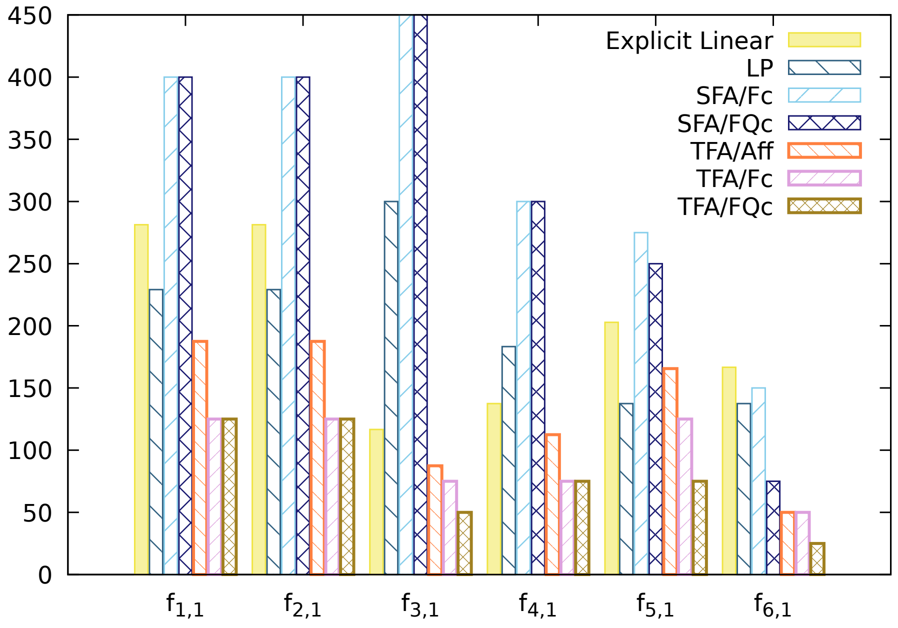


Figure 29: Upper bounds on delay, per flow and per method, second example (topology of Figure 25), fourth experiment (loaded configuration, doubling number of flows, parameters from Table 8).

Flows ($k \in \{1,\dots,5\}$)	$f_{1,k}$	$f_{2,k}$	$f_{3,k}$	$f_{4,k}$	$f_{5,k}$	$f_{6,k}$
Rate	1/15	1/15	1/15	2/15	1/15	2/15
Max. Packet Size.	10	10	10	10	10	10
Burst	28/3	28/3	28/3	26/3	28/3	26/3

Table 9: Flow parameters, second example (topology of Figure 25), fifth experiment (loaded configuration, large number of flows).

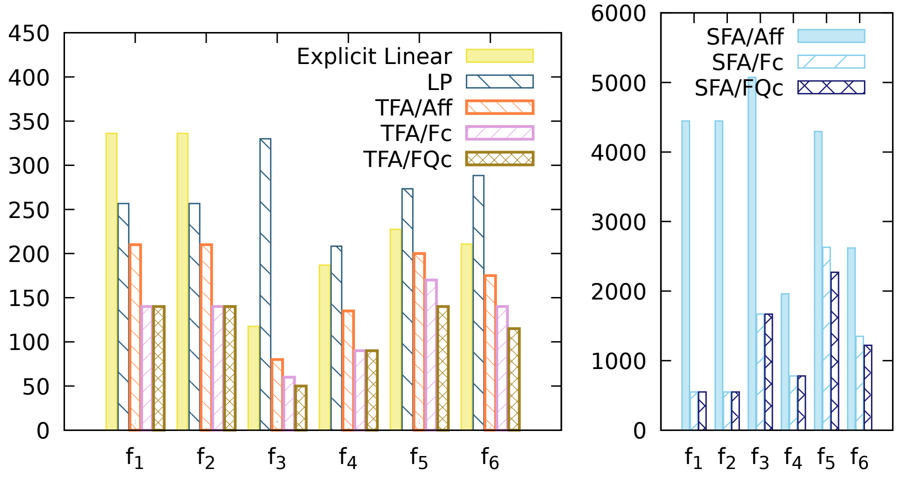


Figure 30: Upper bounds on delay, per flow and per method, second example (topology of Figure 25), fifth experiment (loaded configuration, large number of flows, parameters from Table 9).

8.2.5 Fifth experiment, loaded configuration, large number of flows

The fifth experiment is based on the third one, where each flow f_i is split into five flows $f_{i,1},\dots,f_{i,5}$ with the maximal packet size and the throughput divided by five, as shown in Table 9.

The results are displayed in Figure 30, where the SFA results have been plotted separately because of their very large values. Note that this time, the affine SFA method has been plotted, even if it has no practical interest since it is outperformed by the LP method with equivalent hypotheses (piecewise-linear concave/convex functions, without modeling of shaping for LP).

Considering a fluid model, the TFA approach gives this time the best results for all flows, whereas the explicit linear and the LP methods are incomparable, but always in the same range of values.

On the opposite, the SFA approach gives bounds that are 5 to 10 times bigger than the corresponding TFA approach. When packets are of constant size, the TFA approach can improve its bounds.

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

8.3 Third example, full MPPA NoC

8.3.1 First experiment: 128 flows

The third 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 10. 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 10. 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 are outperformed by the TFA/FQc method. Therefor, they have not been used in this experiment.

The bound computed for each flow with each method appears in Figure 31, 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 32), 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.

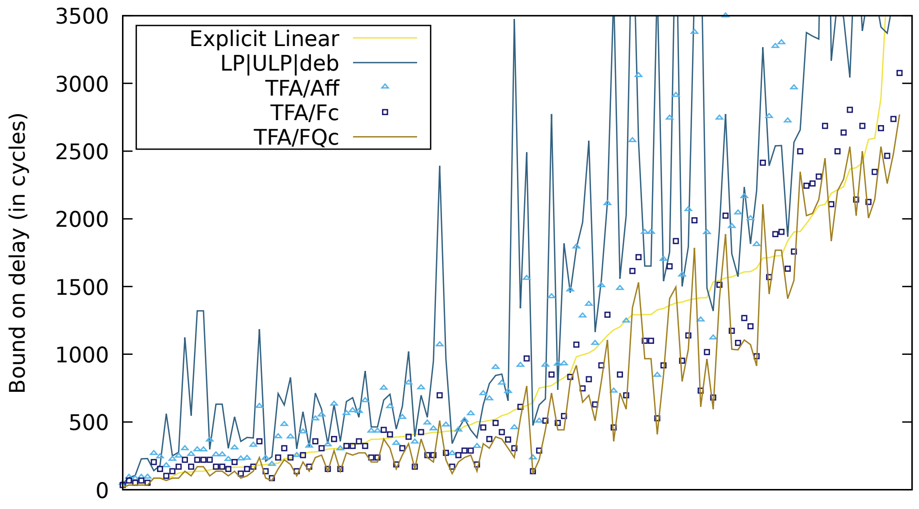


Figure 31: Upper bounds on delay, per flow and per method, third example, first experiment. The flow are sorted by bound value with explicit linear method.

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 33 plots the same bounds as Figure 31, 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 34 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 33. Figure 32 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.

8.3.2 Second experiment: 256 flows

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

The distribution of flow length is given in Table 11, 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 flow length are given in Figure 35. Since they are more flows, they are more conflicts per router, and the worst case delays are increased. The mean gain

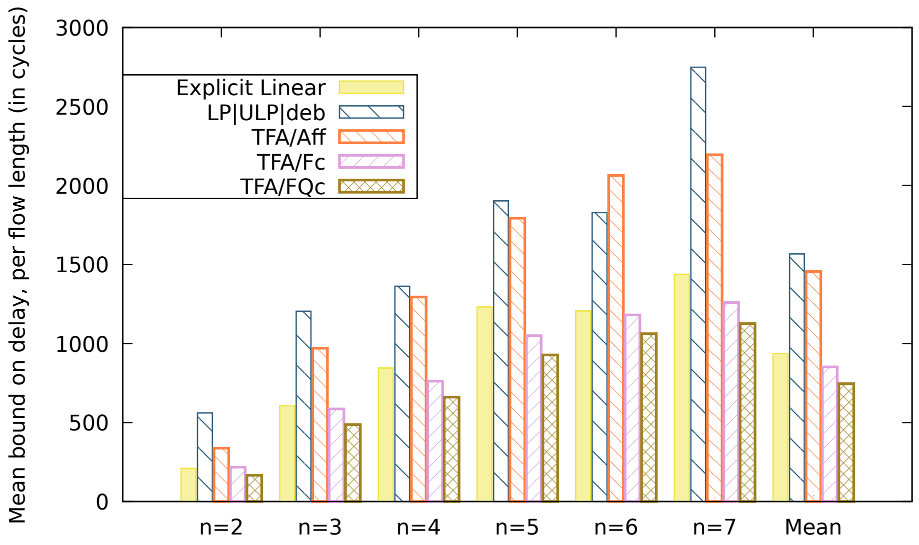


Figure 32: Mean value of bounds (in cycle), per flow length n, third example, first experiment.

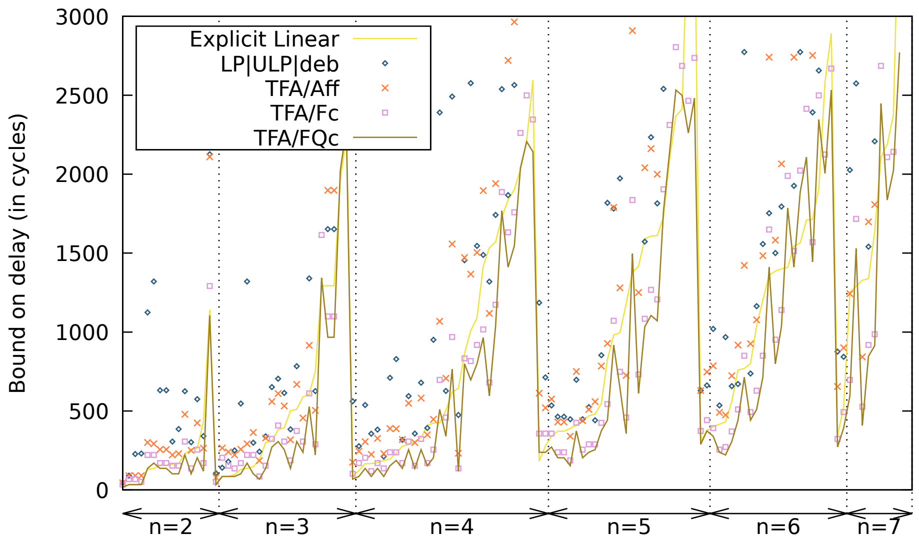


Figure 33: Upper bounds on delay, per flow and per method, third example, first experiment. The flow are sorted first by flow length n then by bound value with explicit linear method.

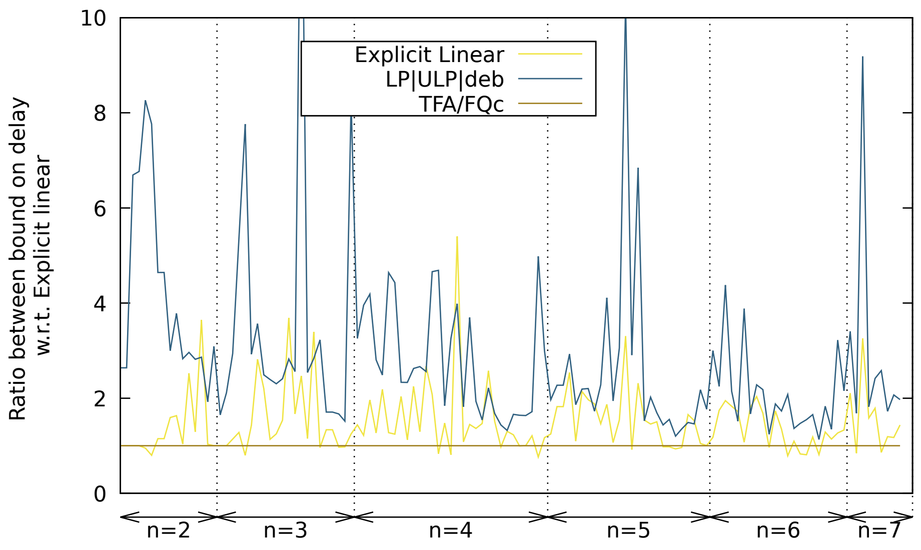


Figure 34: Ratio between each method and the TFA/FQc one, third example, first experiment, same sorting as in Figure 33.

.

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

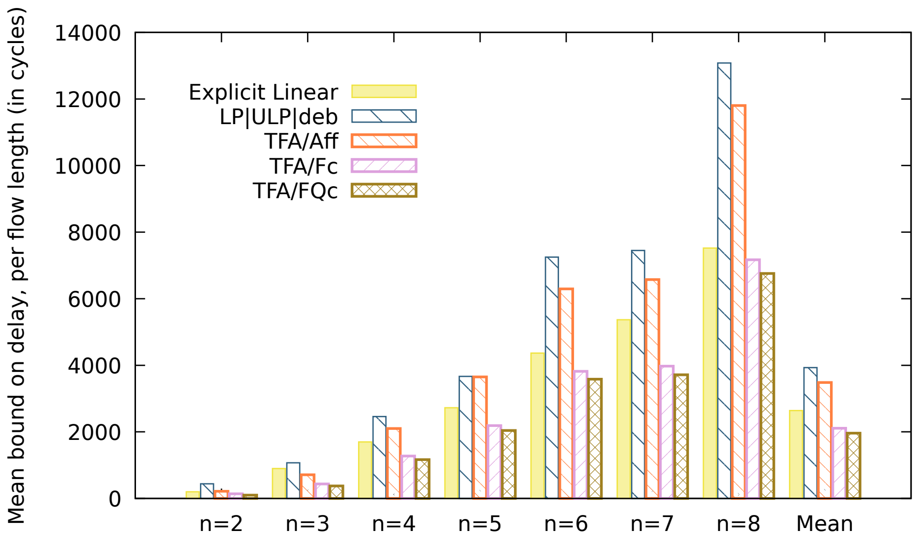


Figure 35: Mean value of bounds (in cycle), per flow length n, third example, second experiment.

between the explicit linear and TFA/FQc methods becomes 25%, whereas it was 20% in the previous experiment with 128 flows.

8.4 Conclusions on the case studies

Theses experiments on a realistic case study give us several insights, some confirming well known features of network calculus, and some more unexpected.

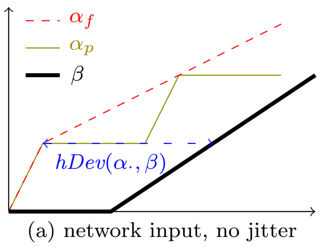
The well known result is that modeling the shaping introduced by link capacity has a significant impact on delay bounds. Theses has been shown in the context of AFDX network in Frances et al. (2006) and confirmed in Boyer and Fraboul (2008), Scharbarg et al. (2009), Zhao et al. (2013): while the LP method computes the exact worst case for the FIFO policy (without considering shaping), it is outperformed by methods that model link shaping: explicit linear method (in all cases), and TFA methods on the first and second case studies.

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

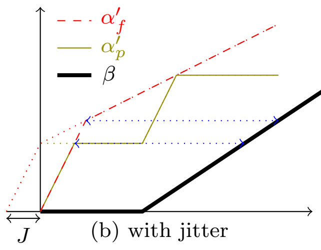
$$\mathtt{Dti} \geqslant \mathtt{Atj} + R(\mathtt{ti} - \mathtt{tj} - T) \tag{40}$$

where $\mathtt{ti}, \mathtt{tj}, \mathtt{Dti}, \mathtt{Atj}$ 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

$$\mathtt{Dtm} + \mathtt{Dtn} \leqslant C(\mathtt{tm} - \mathtt{tn}). \tag{41}$$



(a) network input, no jitter



(b) with jitter

Figure 36: Gain related to modeling of packets in arrival curve.

for all variables Dtm, Dtn related to the function D and the instants t_m, t_n in the program.

A second expected result is that the 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 19, 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 36 illustrate 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 delay associated with each arrival curve are different, as illustrated in Figure 36.b.

Conversely, the modeling of packet sizes in the round robin scheduling policy gives a bigger service curve, as illustrated in Figure 9. But the benefit in the delay evaluation is related to the modeling of the packet size in the arrival curve also. Figure 37 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 38, and even if there is a gain at modeling packet size, it may be smaller.

Nevertheless, an accurate modeling of packet sizes requires to abandon 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). There encoding in a integer linear program is not so straightforward and may moreover increase the computation time.

On the side of unexpected result, a first one was the usability of the LP

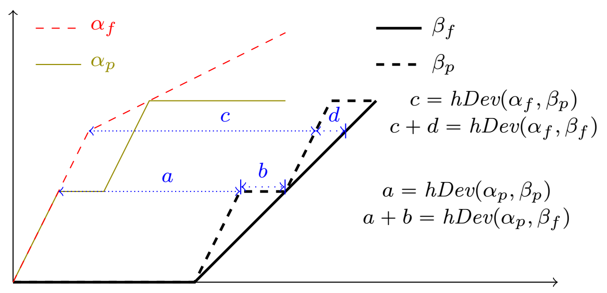

α_f
α_p
β_f
β_p
c
d
a
b
$c = hDev(\alpha_f, \beta_p)$
$c + d = hDev(\alpha_f, \beta_f)$
$a = hDev(\alpha_p, \beta_p)$
$a + b = hDev(\alpha_p, \beta_f)$

Figure 37: Gain related to modeling of packets in both arrival and service curves, when service packet size fits arrival packet size.

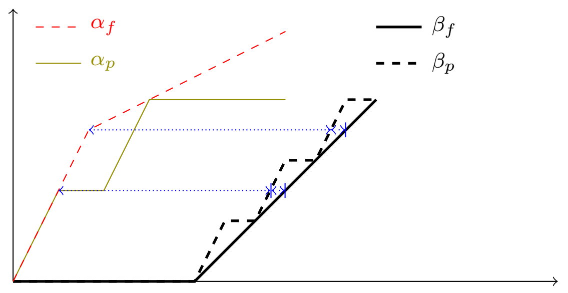


Figure 38: Gain related to modeling of packets in both arrival and service curves, when service packet size does not fit arrival packet size.

solution. Whereas it has a theoretical exponential cost, related to the length of paths, it can in practice be used for NoC, since the paths are not very long. Table 10 shows that even if it was no able to deal with paths of size greater than 6 (in less than 2mn), its 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.

But the main unexpected result is the lower accuracy of the Single Flow Analysis (SFA) approach w.r.t. Total Flow Analysis (TFA) approach. 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. But for the FIFO policy, SFA requires the choice of a θ parameter (the choice of its value has been discussed in section 7.2). It may happen that with a better choice of this parameter, SFA can give better results than TFA, 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 32, 35), for a given flow, the difference may be very large (cf. Figure 34). 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 an architecture, one needs some method to bound the communication latency 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 small already published examples, to get a comprehensive view on their differences, and to compare new methods with the previous one on known examples. They also have been compared with the Recursive Calculus approach on one 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": even if all methods give similar average results on the large case study, no method always have the best bound. But 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 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 Section 7.2). 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 Guan and Yi (2013); Lampka et al. (2016).

Computing better bounds with the LP method will require to encode 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, we expect that it may be added without introducing new variables, as presented in Section 8.4, 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.

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 International Symposium 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

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

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) Proceedings of the 4th International Symposium On Leveraging Applications of Formal Methods, Verification and Validation (ISoLA 2010), Springer, LNCS

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

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), Munster, Deutchland

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

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

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

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

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, URL `http://dx.doi.org/10.1007/s10626-007-0028-x`, http://www.springerlink.com/content/876x51r6647r8g68/

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 industrial Electrony Society, pp 11–20

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

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 http://doi.acm.org/10.1145/2071589.2071595, URL http://doi.acm.org/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

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

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

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, pp 184–195

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

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 http://dx.doi.org/10.1016/S0166-5316(02)00116-5, URL http://www.sciencedirect.com/science/article/pii/S0166531602001165

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

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

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

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

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

Firoiu V, Boudec JYL, Towsley D, Zhang ZL (2002) Theories and models for internet quality of service. Procof the IEEE 90(9):1565–1591

Frances F, Fraboul C, Grieu J (2006) Using network calculus to optimize AFDX network. In: Proceeding 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

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 https://doi.org/10.1016/j.comcom.2016.09.006, URL http://www.sciencedirect.com/science/article/pii/S0140366416303358

Georges J, Divoux T, Rondeau E (2005) Strict priority versus weighted fair queueing in switched ethernet networks for time critical applications. In: 19th IEEE International Parallel and Distributed Processing Symposium, 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 International Conference on Computer Communications (INFOCOM 2019), Paris, France

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, URL `https://doi.org/10.1007/s11241-015-9227-y`

Guan N, Yi W (2013) Finitary real-time calculus: Efficient performance analysis of distributed embedded systems. In: Proc. or the IEEE 34th Real-Time Systems Symposium (RTSS'2013), IEEE, pp 330–339

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

Ito Y, Tasaka S, Ishibashi Y (2002) Variably weighted round robin queueing for core ip routers. In: Performance, Computing, and Communications Conference, 2002. 21st IEEE International, IEEE, pp 159–166

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

Katevenis M, Sidiropoulos S, Courcoubetis C (1991) Weighted round-robin cell multiplexing in a general-purpose atm switch chip. Selected Areas in Communications, IEEE Journal on 9(8):1265–1279

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

Lampka K, Bondorf S, Schmitt J (2016) Achieving efficiency without sacrificing model accuracy: Network calculus on compact domains. In: Proc. of the 24th IEEE International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS 2016), IEEE, pp 313–318

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

Lenzini L, Mingozzi E, Stea G (2004) Delay bounds for FIFO aggegates: a case study. Computer Communications 28:287–299

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

Lenzini L, Mingozzi E, Stea G (2007) End-to-end delay bounds in FIFO-multiplexing tandems. In: Glynn P (ed) Proc. of the 2nd International Conference 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

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, pp 1–4

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

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

Nikolić B, Tobuschat S, SoaresIndrusiak 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, URL `https://doi.org/10.1007/s11241-018-9312-0`

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: Proceedings of the 24th International Conference on Real-Time Networks and Systems (RTNS 2016), ACM, pp 235–244

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, pp 1–11

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 International Symposium on Networks-on-Chip (NoCS 2009), IEEE, pp 44–53

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 International Conference on Hardware/Software Codesign and System Synthesis, CODES+ISSS 2015, Amsterdam, Netherlands, October 4-9, 2015, pp 220–229

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, numéro spécial: actes de la conférence sur la modélisation des systèmes réactifs (MSR'09)

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

Shi Z, Burns A (2008) Real-time communication analysis for on-chip networks with wormhole switching. In: Networks-on-Chip, 2008. NoCS 2008. Second ACM/IEEE International Symposium on, IEEE, pp 161–170

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, pp 590–595

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

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

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

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

A Residual service of weighted round robin

This appendix is devoted to the modeling of the Weighted Round Robin (WRR) policy in the network calculus framework. The MPPA arbiters only apply a round robin (RR) policy, but the generalization to the weighted round robin is straightforward, so is presented the generalization.

In WRR, a parameter $w_i \in \mathbb{N}$, the *weight*, is associated to each queue. The set of queues is logically seen as a ring, and each queues is selected one after the other, in the ring order, starting again once a complete turn is done. Each time a queue i is selected, up to w_i packets are sent, as presented in algorithm 1.

```
Input: Per flow weight: $w_1..w_n$ (Integer)
Data: Counter: $k$ (Integer)
while True do
    for $i$= 1 to $n$ do
        print (“session start”, now(), $i$) ;
        $k \leftarrow 1$ ;
        while (not empty($i$)) and ($k \leqslant w_i$) do
            send(head($i$)) ;
            removeHead($i$) ;
            $k \leftarrow k + 1$ ;
        end
    end
end
```

Algorithm 1: WRR algorithm. `print` is a pseudo instruction, used to simplify the proof.

Also note that it exists two variants of the WRR policy.

The most commonly accepted behavior is the one presented here, cf. Ito et al. (2002),Georges et al. (2011), whereas the initial paper was slightly different. In Katevenis et al. (1991), which is to your knowledge the first paper on a weighted version of round robin, a counter c is incremented at each new turn, from 1 up to $w = \max_i w_i$. In a given turn, only the queues with $w_i \geqslant c$ are allowed to send only one packet. This policy avoids long bursts from a single queue, and have better worst-case performance. But this is not the one kept by the community.

Here comes the main result, given as a lemma to decompose the proof. Note that the condition on the backlog period really looks like the definition of a strict service, but in this lemma, only an existential quantification is used, whereas the strict service requires a universal quantification.

Lemma 1 (WRR output share)**.** *Let S be a server shared by n flows, denoted by $(A_1, \ldots, A_n) \xrightarrow{S} (D_1, \ldots, D_n)$, applying a weighted round robin policy of $w_1, \ldots, w_n \in \mathbb{N}$. For any $i \in [1, n]$, let $l_i^{\min}, l_i^{\max}$ be some lower and upper bound on the packet size the i-flow.*

Let $i \in [1, n]$ be one flow. Then, if it exists $s, t \in \mathbb{R}^+$ such that $(s, t]$ is a backlog period for the flow i, and

$$\sum_{k=1}^{n} D_k(t) - \sum_{k=1}^{n} D_k(s) \geqslant \beta(t - s),$$

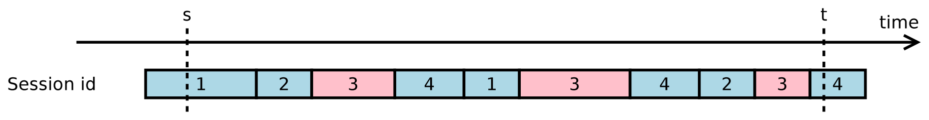


Figure 39: Alternation of service sessions.

then

$$D_i(t) - D_i(s) \geqslant (\lambda_1 * \nu_{q_i,q_i+Q_i})\left(\beta(t-s) - Q_i\right) \tag{42}$$

$$D_i(t) - D_i(s) \geqslant \frac{q_i}{q_i + Q_i}\left(\beta(t-s) - Q_i\right) \tag{43}$$

with $q_i = w_i l_i^{\min}$ *and* $Q_i = \sum_{j \neq i} w_j l_j^{\max}$.

The idea of the proof is the following: considering the interval $[s,t)$, the number of turns of the main loop during this interval is denoted p. If the flow of interest has index i, they are at least pw_i packets of this flow, and at most $p\sum_{j\neq i} w_j$ of the other flows. Combining both allows to eliminate p, leading to equation (48) which can be written $f(D_i(t) - D_i(s)) \geqslant \beta(t-s)$ where f is a increasing function. Then, it admits an inverse, leading to eq. (42), which can be under-approximated by eq. (43).

Proof. Let define a *full service session* for any flow j a (possibly empty) time interval $(x,y]$ such there is in the trace a "session start" of index j and instant x and the next "session start" is of index $((j+1) \mod n) + 1$ and at instant y.

Then, for each full service session of any flow j, at most w_j packets are sent, and for a sequence of k full services session, they are at most kw_j packets sent.

Now, let $(s,t]$ be a backlog period of the flow of interest, i.

Let p denote the number of full service sessions of the flow of interest i in the interval $(s,t]$. In this interval, there were at most $(p+1)$ full service sessions of each other flow, leading to

$$\forall j : D_j(t) - D_j(s) \leqslant (p+1) w_j l_j^{\max} \tag{44}$$

Moreover, since $(s,t]$ is a backlog period for the i-th flow, they always are packets of the i-th flow to send, leading to

$$D_i(t) - D_i(s) \geqslant p w_i l_i^{\min} \tag{45}$$

Now, these results can be injected in the initial assumption:

$$\sum_{k=1}^{n} D_k(t) - \sum_{k=1}^{n} D_k(s) \geqslant \beta(t-s)$$

$$\overset{(44)}{\Longrightarrow} \; D_i(t) - D_i(s) + \sum_{j\neq i} (p+1) w_j l_j^{\max} \geqslant \beta(t-s) \tag{46}$$

$$\iff \; D_i(t) - D_i(s) + Q_i(p+1) \geqslant \beta(t-s) \tag{47}$$

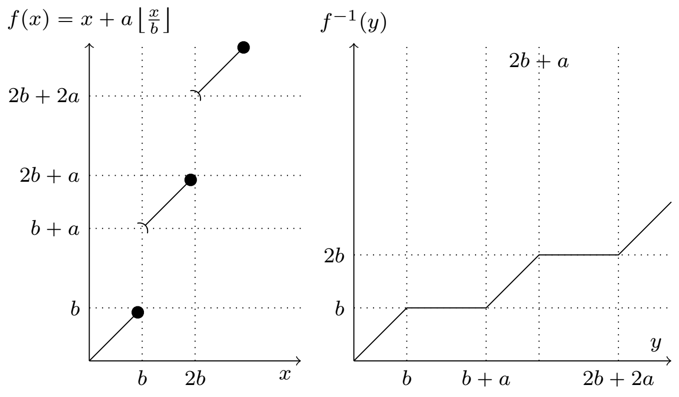


Figure 40: Function $x \mapsto x + a\left\lfloor \frac{x}{b} \right\rfloor$ and its inverse $\lambda_1 * \nu_{b,a+b}$.

From eq. (45), an upper bound on p can be deduced: $p \leqslant \frac{D_i(t)-D_i(s)}{w_i l_i^{\min}} = \frac{D_i(t)-D_i(s)}{q_i}$. Moreover, p is a natural number, so $p \leqslant \left\lfloor \frac{D_i(t)-D_i(s)}{q_i} \right\rfloor$, leading to

$$D_i(t) - D_i(s) + Q_i \left\lfloor 1 + \frac{D_i(t) - D_i(s)}{q_i} \right\rfloor \geqslant \beta(t-s) \tag{48}$$

Using the relation $1+\lfloor x \rfloor = \lfloor 1 + x \rfloor$, the expression can also be slightly simplified into $D_i(t) - D_i(s) + Q_i \left\lfloor \frac{D_i(t)-D_i(s)}{q_i} \right\rfloor \geqslant \beta(t-s) - Q_i$.

Let define $f : \mathbb{R}^+ \to \mathbb{R}$ by $f(x) = x + Q_i \left\lfloor \frac{x}{q_i} \right\rfloor$. This function is increasing (the function $\lfloor \cdot \rfloor$ is non-decreasing, and the identity is increasing), and admits an inverse. The previous relation can be written

$$\begin{aligned} & f(D_i(t) - D_i(s)) \geqslant \beta(t-s) - Q_i \\ \implies & D_i(t) - D_i(s) \geqslant f^{-1}(\beta(t-s) - Q_i). \end{aligned}$$

The inverse of a function $x \mapsto x + a\left\lfloor \frac{x}{b} \right\rfloor$ is $\lambda_1 * \nu_{b,a+b}$ (see Figure 40). Then, eq. (42) holds.

The eq. (43) is derived from the fact that $\lambda_1 * \nu_{b,a+b} \geqslant \lambda_{\frac{a}{a+b}}$.

□

Now comes the proof of Theorem 5. This is a direct application of Lemma 1, but Lemma 1 can be used for other kind of service, like weakly strict service Bouillard (2011). Such generalization is out of the scope of this paper, but the Lemma has been introduced to let it possible in future work.

Proof of Theorem 5. Let $(s,t]$ be a backlog period for the flow i. Then, it also is a backlog period for the aggregate flow, and since $\mathcal{S}$ is offers a strict service, the relation $\sum_{i=1}^{n} D_i(t) - \sum_{i=1}^{n} D_i(s) \geqslant \beta(t-s)$ holds. Applying Lemma 1 leads to $D_i(t) - D_i(s) \geqslant (\lambda_1 * \nu_{q_i, q_i + Q_i})\,(\beta(t-s) - Q_i)$ and $D_i(t) - D_i(s) \geqslant \frac{q_i}{q_i + Q_i}\,(\beta(t-s) - Q_i)$, which is the condition of a strict service. $\square$