Non preemptive static priority in network calculus: accuracy and integration with P-GPS

William Mangoua Sofack, Marc Boyer
ONERA – The French Aerospace Lab
F31055 Toulouse, France
{William.Mangoua_Sofack,Marc.Boyer}@onera.fr

Abstract

The real-time behaviour of critical real-time systems relies on the real-time behaviour of the network and some method to compute bounds on the traversal time (worst case traversal time – WCTT). Network calculus (NC) is a method designed to compute such bounds, and one challenge is to have tight bounds, i.e. *reaching the exact worst case, to avoid over-provisioning. In this paper, we present a new way to handle non-preemptive static priority (NP-SP) with NC, very accurate (it reaches the real worst cases in all thousands tested configurations), and to combine it with packet-based generalised processor sharing (P-GPS), providing bounds on NP-SP/P-GPS in network calculus*[1].

1. Introduction

Static priorities (SP) and Generalised Processor Sharing (GPS) are two common scheduling policy used to share the bandwidth of a network element, responding to two different needs: SP ensure that some critical flow will not be disrupted by some less critical one, and GSP ensure a fair sharing between different flows. Moreover, because of non-preemption in networks, non-preemptive SP (NP-SP) and packet-based GPS (P-GPS, aka WFQ [9]) are used in place of SP and GPS. One may like, in real-time systems, to combine these two policies, having at first level a strict static priority scheduling, ensuring the full server speed to some flows with low latency, and some fair sharing between flows sharing a same priority level, giving a global NP-SP/P-GPS scheduling.

Nevertheless, whatever the policy is, in critical systems, some method bounding the network traversal time is needed. And the bound has to be as close as possible to the real worst case to avoid over-provisioning. Network calculus [1, 2] is a method designed to compute such bounds, that have been used to certify the A380 backbone [3].

This paper presents addresses the combination NP-SP/P-GPS in network calculus.

[1]This works has been partialy funded by French ANR agency under project id ANR-09-SEGI-009.

NP-SP have already been studied [1, 4, 5] but these results appear not to carefully consider the hypothesis of non-preemption. In particular, when a non-preemptive flow is served, it benefits from the full speed of the server, even if, from long term point of view, it gets only a fractional part. This assessment was the starting point of [6] and [7]. The more accurate results are in [6], and the more general in [7]. [8] improves [7], and this paper shows that the new results on NP-SP are very accurate, since the real worst case is reached in all 2000 tested configurations. Moreover, some technical point of the contribution of [8] (the residual service is *strict*, not *simple* as in [6]) allows to consider the low priority flow itself as the aggregation of some sub-flows, and to apply another policy between these flows.

GPS, and P-GPS (also known as WFQ) is a common scheduling policy [9], designed to share a system in a "fair" way. Nevertheless, in all studies, to our knowledge, the capacity (or bandwidth) of the shared system was always assumed to be a constant. More formally, the system was assumed to be able to handle $r(s-t)$ work units on any interval $[t, s]$. Network calculus generalises this notion, considering a capacity $\beta(s-t)$, for any β function. Non constant service can be of interest when considering energy management, or when the considered service is what is left unused by some disturbing process. In this paper, it is shown how the definition of GPS and P-GPS can be generalised to any kind of service in network calculus.

At last, thanks to the compositional aspect of network calculus, the combination NP-SP/P-GPS comes for free.

After a short presentation of network calculus, in Section 2, modelling in network calculus of the on preemptive static priority policy is presented (including some accuracy evaluation in Subsection 3.5). Section 4 presents the modelling of GPS and P-GPS in network calculus. Section 5 gives an example of NP-SP/P-GPS integration. Section 6 concludes.

2. Network Calculus

The network calculus analysis focuses on worst case performances. The information about the system features

are stored in functions, such as arrival curves characterising the traffic or service curves quantifying the service guaranteed at the network nodes. These functions can be combined together thanks to special network calculus operations, in order compute bounds on buffers size or delays.

2.1. Mathematical background: $(\min, +)$ dioid

Here are presented some operators of the $(\min, +)$ dioid used by network calculus. Beyond usual operations like the minimum or the addition of functions, network calculus makes use of several classical operations which are the translations of $(+, \times)$ filtering operations into the $(\min, +)$ setting, as well as a few other transformations.

Network calculus mainly uses non-decreasing functions, and related operators. Here are those used in this article.

Set $\mathcal{F}$ $\mathcal{F}$ denote the set of wide-sense increasing functions $f : \mathbb{R} \to \mathbb{R} \cup \{+\infty\}$ such that $f(t) = 0$ for $t < 0$.

Function $[\,]^+ : x \mapsto \max(x, 0)$.

Flooring and ceiling $\lfloor x \rfloor \in \mathbb{N}, \lfloor x \rfloor \leq x < \lfloor x \rfloor + 1$; $\lceil x \rceil \in \mathbb{N}, \lceil x \rceil - 1 < x \leq \lceil x \rceil$

The Vertical deviation It is defined for two functions f and g by $v(f, g) = \sup_{t \geq 0} \{f(t) - g(t)\}$

The Horizontal deviation It is defined for two functions f and g by $h(f, g) = \sup_{t \geq 0} \{\inf \{d \geq 0 \mid f(t) \leq g(t + d)\}\}$

The Min-plus convolution It is defined for two functions f and g by $(f * g)(t) = \inf_{0 \leq s \leq t} \{f(t - s) + g(s)\}$

The Positive and non-decreasing upper closure It is defined for a functions f by $f \uparrow (t) = [\sup_{0 \leq s \leq t} f(s)]^+$

The pseudo inverse The inverse, f^{-1}, of a function $f \in \mathcal{F}$ cannot always be assumed to exist, however, two pseudo-inverses can be defined [10]:

$$f^{-1}_{\inf}(u) \stackrel{\text{def}}{=} \inf\{t | f(t) \geq u\} \quad \stackrel{\text{ppty}}{=} \sup\{t \mid f(t) < u\}$$
$$f^{-1}_{sup}(u) \stackrel{\text{def}}{=} \sup\{t | f(t) \leq u\} \quad \stackrel{\text{ppty}}{=} \inf\{t \mid f(t) > u\}$$

To model flows constraint and service guarantees, network calculus uses a set of usual parametrised curves, δ_d, λ_R, $\beta_{R,T}$, $\gamma_{r,b}$, $\nu_{T,\tau}$ defined by:

$$\delta_d(t) = \begin{cases} 0 \text{ if } t \leq d \\ \infty \text{ otherwise} \end{cases} \qquad \gamma_{r,b}(t) = \begin{cases} rt + b \text{ if } t > 0 \\ 0 \text{ otherwise} \end{cases}$$
$$\lambda_R(t) = Rt \qquad \beta_{R,T}(t) = R[t - T]^+$$
$$\nu_{T,\tau}(t) = \min\left(\delta_0, \left\lceil \frac{t + \tau}{T} \right\rceil\right)$$

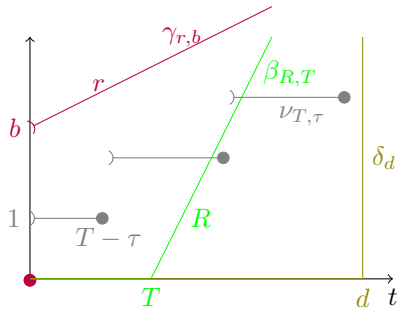

Figure 1. Common curves

2.2. Network calculus: reality modelling

A network calculus model for a communication network consists in the three following components:

1. A partition of the network into subsystems (often called nodes) which may have different scales (from elementary hardware like a processor to large subnetworks).

2. A description of data flows, where each flow follows a path through a specified sequence of subsystems and where each flow is shaped by some arrival curve just before entering the network.

3. A description of the behaviour of each subsystem, that is service curves bounding the performances of each subsystem, as well as service policies in case of multiplexing (several flows entering the same subsystem and thus sharing its service).

In network calculus, the real *flows* are modelled by cumulative functions $R \in \mathcal{F}$: $R(t)$ counts the total amount of data produced by the flow up to time t.

The *servers* are just relations between some input and output flow ($S \in \mathcal{F} \times \mathcal{F}$). Then $(R, R') \in S$, denoted $R \stackrel{S}{\to} R'$, means that a server S receives an input flow, $R(t)$, and delivers the data after a variable delay. We have relation $R' \leq R$, meaning that data goes out after being entered. System S might be, for example, a single buffer served at a constant rate, a complex communication node, or even a complete network. Figure 3 shows input and output functions for a single server queue.

The *backlog* is the amount of bits that are held inside the system; if the system is a single buffer, it is the queue length. In contrast, if the system is more complex, then the backlog is the number of bits "in transit", assuming that we can observe input and output simultaneously [1]. For a system where R is the input and R' the output, *the backlog at time t is* $b(t) = R(t) - R'(t)$. Obviously, $b(t) \leq v(R, R')$.

A *backlogged period* is a period during which the backlog is not zero. Let t, a moment in a backlogged period, this backlogged period has started at $StBl(t) = \sup\{u \leq t \mid R'(u) = R(u)\}$.

The *virtual delay* at a time t is the delay that a bit entered at time t will wait until going out, defined

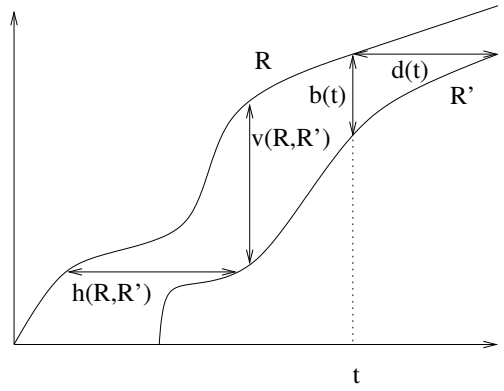

Figure 2. Backlog and delay

by $d(t) = \inf\{\tau \geq 0 \mid R(t) \leq R'(t+\tau)\}$. Obviously $d(t) \leq h(R, R')$.

These notions are illustrated in Figure 2.

2.3. Network calculus: contract modelling

To provide guarantees to data flows, some traffic contract on the traffics and the services in the network are needed. For this purpose, network calculus provides the concepts of arrival curve and service curve.

Arrival curve A flow $R \in \mathcal{F}$ *is constrained by* $\alpha \in \mathcal{F}$ if and only if for all $s \leq t$: $R(t) - R(s) \leq \alpha(t-s)$. We say also that R *has* α *as an arrival curve*, or also that R *is* α*-smooth*. This condition is equivalent to $R \leq R * \alpha$.

The function $\gamma_{r,b}$ models the token-bucket contract: the flow can send a burst of size b, and has a long-term rate r. The function $s\nu_{T,\tau}$ models a sporadic flow with packets of maximal size s, a pseudo-period T and a jitter τ.

Service curve The behaviour of a server is modelled by the concept of service curve, modelling some guarantees on the service provided to flows.

The literature offers several definitions for different flavours of service. [4] proposes a comparative study. Consider a system $R \xrightarrow{S} R'$, *i.e.* a server S with input R and output R' (Figure 3).

The server S offers to the flow a *simple service of curve* β if and only if, for all pair $R \xrightarrow{S} R$, $R' \geq R * \beta$.

We say that a system S offers a *strict service of curve* β if, f, for all pair $R \xrightarrow{S} R$, during any backlogged period $[t, s)$, we have $R'(s) - R'(t) \geq \beta(s-t)$.

There is a hierarchy between these service notions. A strict service is also a weak service. As discussed in Section 2.4, the need to have these different definitions is the decomposition of the residual service.

Let us now present the main network calculus results:

Theorem 1 (Backlog and delay bound)**.** *Assume a flow, constrained by an arrival curve* α*, traverses a system that offers a service curve* β*, the backlog* $b(t)$ *for all* t *satisfies:* $b(t) \leq v(\alpha, \beta)$.
The virtual delay $d(t)$ *for all* t *satisfies:* $d(t) \leq h(\alpha, \beta)$.

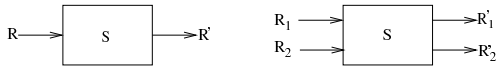

Figure 3. Servers.

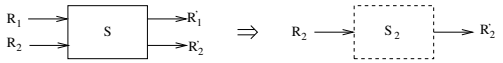

Figure 4. Residual server and residual service

2.4. Aggregation and residual service

In general, servers are not used to transfer one single flow, but a set of flows. The definition of server must be generalised to multiple-input/multiple output servers: $S \in \mathcal{F}^n \times \mathcal{F}^n$, $(R_1, \ldots, R_n) \xrightarrow{S} (R'_1, \ldots, R'_n)$. And the capacity of the server is shared by several flows.

Modelling aggregation and residual service is an important issue in network calculus. Aggregation means that the service is shared by different flows: for example, if a server S offers an aggregated simple service of curve β to two flows R_1 and R_2 , it means that it offers this service to the flow $R = R_1 + R_2$ (*i.e.* $R'_1 + R'_2 \geq (R_1 + R_2) * \beta$), but the repartition of the service between the flows depends on priority flows and server policy (common policies are FIFO [11], static priorities [5, 4], P-GPS/WFQ [9]).

The global idea, in network calculus, is to consider the residual server S_i associated to each flow ($R_i \xrightarrow{S_i} R'_i$), and to derive a residual service (simple or strict) of curve β_i offered by this server.

An important issue is the tightness: in network calculus, a residual service is said to be *tight* iff the residual service allows to compute a delay that is a reachable case, and not an over approximation.

In aggregation and residual service, the flavour of service is crucial: some results need the assumption of strict service, others of simple[2]. And the residual service can also be simple or strict. And this is of importance in the case of more than two flows, since a residual service can be shared by aggregated flows.

3. Static priority in Network calculus

This part focuses on non-preemptive static priorities, and the residual service of the low priority flows.

First, related works are presented (Section 3.1), then, the contribution on NP-SP is presented (informally in Section 3.2, formally in Section 3.3, and illustrated in Section 3.4. The accuracy of the method is evaluated in more than 2000 cases in Section 3.5.

3.1. Related works

Several results have been published on this subject, each one refining the previous results. A detailed com-

[2] In fact, up to now, most results need a *strict* service, and the FIFO policy is the only one requiring only simple service.

parison can be found in [8]. Here is just a quick overview.

All works assume a server S with a *strict* service of curve β, shared by several flows R_i of arrival curve α_i.

The first works on static priority can be found in [1, Cor. 6.2.1], but it have some technical limitations[3], and the residual service is simple.

The two limitations described above have been independently studied by [4, 5] with exactly the same result. They can then be applied to any pair of arrival and service curve, and can derive a simple and a strict service[4].

But these approaches only model the negative impact of non-preemption (the high-priority can be delayed by one low priority packet), but not a positive one: when a non-preemptive flow is served it benefits of the full speed from the server.

The positive impact of non-preemption on the low priority flow described above has been studied in [6], which gives an algorithm (without proof) to compute a residual service with better delay bound than the one of [4, 5], assuming fixed size packets for all flows, and a constant service rate[5]. The residual service is simple[6].

Out of the network calculus area, the non-preemptive static priority policy have been used, for example in [12] for the CAN network. It assumes periodic messages of bounded size, and computes the exact worst case.

3.2. Contribution overview

The same positive effect of non-preemption is the subject of [7, 8]. It generalises [6]: it need less hypotheses (it does not assume a constant rate server, and fixed packet size only for the considered flow), an analytical expression for the residual service is given (with its proof), and the residual service is strict.

This work also generalises [12] in all except one point: it assumes constant packet size for the flow R_i to compute its residual service β_i, as [12] only need bounded size. But on all other aspects, it generalises it: the flows are not assumed to be periodic (only one arrival curve is needed), the server is not assumed to be a constant rate (it can be a variable speed system, or the residual service left by other flow, etc.) and we do not only compute a delay but a residual service.

An important question is the tightness of the result: does the residual service allow to compute the exact worst case, or only an over-approximation? There no proof of the tightness, up to now, but in this paper, the results are compared with the ones of [12], on a very large set of experiments (Section 3.5) and the exact worst case is always reached, giving us strong confidence in the fact that this work generalises [12], with the same exact accuracy.

[3] It can not be applied on any pair kind of service – arrival curve, since the term $[\beta - \alpha_1 - l_2^{\max}]^+$ must be non decreasing.

[4] Note that [5] handle real-time calculus, not network calculus, but, they are equivalent on the considered part, as shown in [4, § 3].

[5] The assumption is implicit in all the paper and the algorithm.

[6] The flavour of service is not given in [6], but [7] shows that it is a simple one.

	Period	Size	α_i	delay
R_1	3	1	$\lceil \frac{t}{3} \rceil$	4
R_2	9	3	$3\lceil \frac{t}{9} \rceil$	5
R_3	4	1	$\lceil \frac{t}{4} \rceil$	6

Table 1. First example

i	χ'_i	$\chi"_i$	χ_i
1	2	-4	2
2	7	1	7
3	11	5	11
4	16	10	16

Table 2. Calculation of β_2

3.3. Theorem

Theorem 2 (NP-SP residual service)**.** *Consider a server that offers to three flows, R_1, R_2 and R_3, a strict service curve represented by a non-decreasing function β. Suppose that the flow R_2 (resp. R_3) emits its data into packets of fixed size l_2 (resp. of maximal size l_3). If the flow R_1 (resp. R_2) is α_1 (resp. α_2) upper-constrained and R_1 has a non-preemptive priority over the flow R_2 and R_2 has a non-preemptive priority over the flow R_3, then the server guarantees to R_2 a strict service curve β_2^{np} defined in eq (1).*

$$\beta_2^{np}(t) = \min \begin{cases} i \times l_2 \\ \beta(t) + (i-1) \times l_2 - \beta(\chi'_i) \\ \beta(t) + (i-1) \times l_2 \\ \quad - \beta(\chi"_i + \psi_2) + \beta(\Delta + \psi_2) \end{cases} \quad (1)$$

with $i = \max\{j : \chi_j \leq t\}$ and the definitions:

$$\psi_1 = (\beta - \alpha) \uparrow^{-1}_{\sup} (0) \qquad \psi_i = \beta^{-1}_{\sup}(l_i) \text{ for } i \in \{2, 3\}$$
$$\chi'_i = ((\beta - \alpha_1) \uparrow)^{-1}_{\inf}(l_3 + (i-1) \times l_2)$$
$$\chi"_i = ((\beta - \alpha_1) \oslash \delta_{\psi_2})^{-1}_{\inf}(i \times l_2)$$
$$\chi_i = \max\{\chi'_i, \chi"_i\} \qquad \Delta = (\alpha_2)^{-1}_{\inf}(2 \times l_2) - \psi_2$$

3.4. Example

As an illustration, consider a server with a strict service curve $\beta(t) = t$ for three flows R_1, R_2 and R_3. Each flow R_i is α_i upper-constrained, and has fixed packet size l_i, given in Table 1.

Let us evaluate β_2^{np}, the residual service offered to flow R_2. $\Delta = 6$ and Table 2 shows the calculated values of χ_i.

Figure 5 shows the residual service, β_2^{np}, offered to flow R_2 using the results of [8]. The function β_2, computed using [4, 5] is also drawn: $\beta_2^{\text{np}} \geq \beta_2$, meaning that our residual service is more accurate.

3.5. Accuracy

Theorem 2 gives a way to compute upper bounds on delays, but the question of its accuracy arises: is the result tight or not? Did it computes the exact worst case?

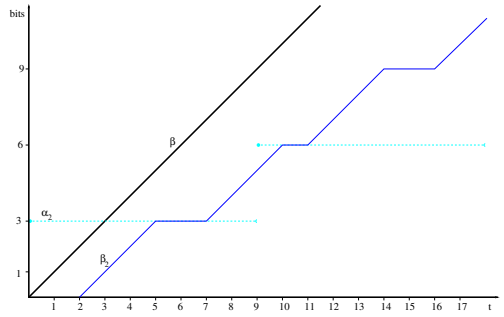

Figure 5. Residual service β_2^{np} on first example (Table 2)

Test Size	Nb Tests	Mean Load
2	100	0.986988
3	217	0.984841
4	325	0.983228
5	452	0.986097
6	433	0.987534
7	342	0.988197
8	233	0.987539
9	118	0.98833
10	105	0.960005

Table 3. Configurations numbers and loads

We do not have, up to now, any tightness proof, neither any counter example.

Since [12] is known to compute the exact worst case, (assuming periodic flows and bounded frame size) the two approaches have been compared on a significant number of cases. Of course, the famous counter example of [12, Table 3] have been one of the first test of our method, and, as presented in [7, Table 2], the exact worst case is reached using Theorem 2. This was a first test of the method, but insufficient.

We have consider systems with n flows ($n \in [2, 10]$), shared by periodic flows R_i, each flow having its period $T_i \in [1, 40]$, and messages of fixed size s_i.

In network calculus, each flow R_i has an arrival curve $\alpha_i = s_i \nu_{T_i,0}$. The systems offers a strict service of curve $\beta(t) = t$.

The number of flow is uniformly randomly chosen in $[2, 10]$[7]. Then, the period of each flow is randomly chosen in $[2, 40]$. The size of the flows is chosen to have a high system load: the first flow randomly chose to use a part of the bandwidth ρ_i, and the message size is then $s_i = \lfloor \rho_i T_i \rfloor$. The second flow randomly chose a part of the remaining bandwidth, and so on.

The (min,plus) computations have been done with the (min,plus) library of the RT@W-PEGASE tool [13].

More than 2000 configurations have been generated, all with a heavy load (an overview is presented in Table 3). In all configurations, both Theorem 2 and [12] gives the same result, *i.e.* the exact worst case.

Such experiment does not gives a tightness proof, but provides a strong confidence that the result is tight, at least when considering periodic messages and constant server.

4. GPS and P-GPS in network calculus

GPS and P-GPS (often known as WFQ [9]) are two commons scheduling policies.

Nevertheless, up to our knowledge, all papers on such subject, when looking into details of proofs, assume that the server has a fixed constant capacity (also known as constant rate server), *i.e.* the system was assumed to be able to handle $r(s-t)$ work units on any interval $[t, s]$. [1, § 2.1] integrates the result of [9] into network calculus but assumes also constant rate server. Some papers on real-time calculus make some reference to GPS [14, Fig. 8], but we did not find any formal proof for the general case of any β service function. And, when trying to integrate SP-NP and P-GSP, we can not restrict ourselves to the constant rate server[8].

So, this section formally generalises the GPS and P-GPS scheduling policies to any strict service curve, in the network calculus framework.

4.1. GPS policy

GPS, Generalized Processor Sharing, is presented in [9] as a flow-based multiplexing discipline that is efficient, flexible, and analyzable. However, it is a theoretical scheduler, since it assume some fluid behaviour. Nevertheless, it is of interest, even if, in implementation, approximations of GPS will be used.

Let assume that a server is shared by n flows $R_1, \ldots, R_n$. A Generalized Processor Sharing (GPS) policy, is often presented as a policy that shares is capacity such that, each flow R_i receives a fraction $\phi_i / \sum_j \phi_j$. GPS have been defined in [9] assuming a constant rate service, but this definition can be generalised.

Definition 1 (GPS)**.** *Let S be a server shared by n flows, $R_1, \ldots, R_n$. This server apply a Generalized Processor Sharing policy of (non null) parameters $\phi_1, \ldots, \phi_n$ iff, for all interval $[t, s)$ backlogged for a flow R_i, it holds, for all j*

$$R'_i(s) - R'_i(t) \geq \frac{\phi_i}{\phi_j}(R'_j(s) - R'_j(t)) \qquad (2)$$

This relation is often written $\frac{R'_i(s)-R'_i(t)}{R'_j(s)-R'_j(t)} \geq \frac{\phi_i}{\phi_j}$ but is does not have the same good mathematical properties when $R'_j(s) - R'_j(t) = 0$.

Theorem 3 (GPS residual service)**.** *Let S be a server, with a strict service of curve β, shared by n flows, $R_1, \ldots, R_n$, with a GPS policy of (non null) parameters $\phi_1, \ldots, \phi_n$.*

[7] But, as can be seen in Table 3, the random generator seems not so uniform as can be expected.

[8] In fact, the β_i^{np} function of Theorem 2 can be under-approximated by a rate-latency function, but it would be pessimistic.

Then, this servers offers to each flow R_i the strict service of curve β_i^{gps} defined by:

$$\beta_i^{gps} = \frac{\phi_i}{\sum_{j=1}^{n} \phi_j} \beta \tag{3}$$

Proof. The proof mimics the one of [9], except that considers any curve β, not only a linear $\beta(t) = rt$.

Let $[t, s)$ be a backlogged period for for R_i, then

$$(R'_i(s) - R'_i(t)) \frac{\phi_j}{\phi_i} \geq R'_j(s) - R'_j(t)$$

$$\implies \sum_{j=1}^{n} \frac{(R'_i(s) - R'_i(t))\phi_j}{\phi_i} \geq \sum_{j=1}^{n} R'_j(s) - R'_j(t)$$

and since S offers a strict service β, and $[t, s)$ is a backlog period, $\sum_{j=1}^{n} R'_j(s) - R'_j(t) \geq \beta(t)$

$$\implies (R'_i(s) - R'_i(t)) \frac{\sum_{j=1}^{n} \phi_j}{\phi_i} \geq \beta(t)$$

□

4.2. P-GPS policy

As mentioned in [9]: "a problem with GPS is that it is an idealized discipline that does not transmit packets as entities. It assumes that the server can serve multiple sessions simultaneously and that the traffic is infinitely divisible". Then, [9] defines P-GPS as an "an excellent approximation to GPS even when the packets are of variable length." This definition make no assumption of constant rate service, and does not need be generalised.

Definition 2 (P-GPS)**.** *Let S be a server shared by n flows, $R_1, \ldots, R_n$. This server apply a Generalized Processor Sharing policy of (non null) parameters $\phi_1, \ldots, \phi_n$ iff is has the following behaviour (from [9]): "Let F_p be the time at which packet p will depart (finish service) under Generalized Processor Sharing. Then, a very good approximation of GPS would be a work-conserving scheme that serves packets in increasing order of F_p. Now, suppose that the server becomes free at time τ. The next packet to depart under GPS may not have arrived at time τ and, since the server has no knowledge of when this packet will arrive, there is no way for the server to be both work conserving and serve the packets in increasing order of F_p. The server picks the first packet that would complete service in the GPS simulation if no additional packets were to arrive after time τ".*

In the following, like in [9], for a flow R_i (R_i the input and R'_i the output), which crosses the GPS server, we will note $\hat{R}_i$ ($\hat{R}_i$ the input and $\hat{R}'_i$ the output), the equivalent flow on the P-GPS server ($R_i = \hat{R}_i$ but, to be consistant with [9], the two notations are used in this paper).

Then, an important theorem of [9] can be also generalised to any service curve.

Theorem 4. *Consider $\hat{S}$ a server that offers to n aggregated flows, $R_1, \ldots, R_n$, a strict service curve β, with a $P-GPS$ policy of parameters $\phi_1, \ldots, \phi_n$. Let S^{gps} be the reference GPS server used to compute the P-GPS scheduling.*

$$(R_1, \ldots, R_n) \xrightarrow{\hat{S}} (\hat{R'}_1, \ldots, \hat{R'}_n)$$
$$(R_1, \ldots, R_n) \xrightarrow{S^{gps}} (R'_1, \ldots, R'_n)$$

For all times τ and sessions i:

$$R'_i(\tau) - \hat{R}'_i(\tau) \leq L_{\max} \tag{4}$$

where $L_{\max}$ is the maximum packet length.

Is simply means that the P-GPS can not delay output of more that one packet size, for each flow.

It can also be written $R'_i(\tau) - \hat{R}'_i(\tau) \leq L_{\max}$, but the presentation of [9] have been kept. In [9], the relation is written $(R'_i(\tau) - R'_i(0)) - (\hat{R}'_i(\tau) - \hat{R}'_i(0)) \leq L_{\max}$, which is equivalent. The proof is given in appendix A.

Theorem 5. *Let $S, \hat{S}$ be two servers. Assume that S offers a simple service β, and that, for all $R, \hat{R}'$ such that $R \xrightarrow{\hat{S}} \hat{R}'$, it exist R' such that $R \xrightarrow{S} R'$, verifying*

$$\forall t : R'(t) - \hat{R}'(t) \leq M \tag{5}$$

then, $\hat{S}$ offers a simple service $\beta - M$.

Proof. Let be $R, \hat{R}' \in \hat{S}$, and some $t \geq 0$. It exist $R, R' \in S$ such that eq. 5 holds. And S has minimal service β *i.e.*

$$R'(t) \geq \inf_{0 \leq s \leq t} \{R(t-s) + \beta(s)\}$$
$$\iff R'(t) - M \geq \inf_{0 \leq s \leq t} \{R(t-s) + \beta(s) - M\}$$

$$\implies \hat{R}'(t) \geq (R * (\beta - M))(t)$$

□

From this theorem, we can deduce the following result.

Theorem 6. *Consider a server that offers to n aggregated flows, $R_1, \ldots, R_n$, a strict service curve β. Suppose that these flows transmit their data in the form of packet, with maximum size $L_{\max}$. If these flows are served with P-GPS policy of respective weight $\phi_1, \ldots \phi_N$, then the server guarantees to each flow R_i a simple service curve β_i^{pgps} defined in eq (6).*

$$\beta_i^{pgps} = \beta_i^{gps} - L_{\max} = \frac{\phi_i}{\sum_{j=1}^{n} \phi_j} \beta - L_{\max} \tag{6}$$

Proof. Direct application of Theorem 5 and Theorem 4. □

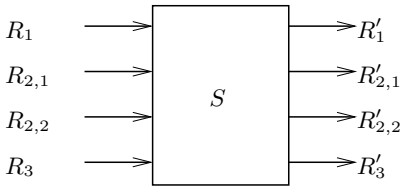

Figure 6. Serveur with NP-SP/P-GPS scheduling

The Theorems 5, 4 and 6 could have been merged into a single one, but P-GPS is not the only approximation of GPS, and, for any policy, if the difference between the ideal GPS and this policy can be bounded, a similar results holds.

Moreover, the theorem need a strict service and only gives a simple service. To have a strict service, one need also a lower bound to the difference $m \leq R' - \hat{R}' \leq M$ and the resulting strict service will be $\beta - (M - m)$ (smaller than $\beta - M$ if $m < 0$).

5. NP-SP/P-GPS integration example

Let us consider a server combining the NP-SP and P-GPS policy. Flows are first grouped by priority, and, up to the non-preemptive effect, no low priority flow is served if there is one higher priority one waiting. Inside a same priority level, flows are served with a P-GPS policy. To apply our result, we also require all packets of the same priority level to have the same size.

The flow $R_{i,j}$ will be a flow of priority level i, and have a P-GPS parameter $\phi_{i,j}$. Each flow $R_{i,j}$ have an arrival curve $\alpha_{i,j}$.

The analyse with network calculus first considers the static priority scheduling: for each priority level i, a "virtual" flow $R_i = \sum_j R_{i,j}$ is build, with an arrival curve $\alpha_i = \sum_j \alpha_{i,j}$. Then, using Theorem 2, the *strict* service β_i^{np} of the residual server S_i can be computed. Then, for each individual flow $R_{i,j}$, using Theorem 6, the residual service $\beta_{i,j}$ can be computed, and the delay of the flow $R_{i,j}$ is S can be bounded by $h(\alpha_{i,j}, \beta_{i,j})$.

A simple example is given there (cf Figure 6). They are three priority levels, with a single flow in the highest and lowest priority levels, and two flows inside the second priority level, scheduled with a P-GPS policy of parameters $\phi_1 = 1, \phi_2 = 2$.

Let us consider quite the same parameters as in Table 1, where the flow R_2 is decomposed into two sub-flows, each one having period 9 and packet size $\frac{3}{2}$, *i.e.* $\alpha_{2,1}(t) = \alpha_{2,2}(t) = \frac{3}{2}\left\lceil \frac{t}{9} \right\rceil$.

The resulting curve β_2^{np} associated to the aggregated flow $R_{2,1} + R_{2,2}$ is presented in Figure 7.

The resulting curves $\beta_{2,i}^{\text{np/gps}}$ assuming a GPS scheduler, and $\beta_{2,i}^{\text{np/p-gps}}$ assuming a P-GPS are presented in Figure 8.

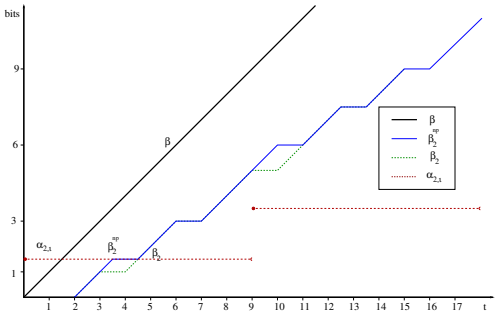

Figure 7. Residual service for $R_{1,2} + R_{2,2}$

Flow	NP-SP/GPS delay	NP-SP/P-GPS Delay
$R_{2,1}$	8	10
$R_{2,1}$	5	7.66

Table 4. Delay for each flow $R_{1,2}$ and $R_{2,2}$

6. Conclusion

Network calculus [1, 2] is a very general theory to compute bounds on worst case performance of real-time distributed systems, but one common criticism was the lack of accuracy, compared to other specific methods.

[8] propose a generalisation of a lot of previous works done on non preemptive static priority [4, 5, 6, 12]: it can deal with any kind of traffic contract, not only periodic or sporadic one, it handles any kind of service curve, not only constant rate output, and provides a residual strict service, an important technical detail in network calculus.

In this paper, these new network calculus results on non-preemptive static priority (NP-SP) scheduling are evaluated, on more than 2000 configurations, In all configurations, its computes the exact worst case. Network calculus then demonstrates it ability to be as good as other methods.

This strict service property allows to combine the residual service with others. In this paper, the P-GPS (also known as WFQ [9]) is chosen, since it is a very common policy in networks. One contribution of this paper is to propose a formal integration of P-GPS into network calculus and also to generalise P-GPS scheduling to the case of non constant rate service. We then prove the possibility to compute an upper bound on worst case performance of a system combining NP-SP and P-GPS policies.

The approach currently still have one drawback: it assume fixed packet size for the considered flow. It seems possible to solve it using [15].

The approach also open very interesting perspective. Its very general assumption can be use in the case of scheduling with voltage scaling for example.

References

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

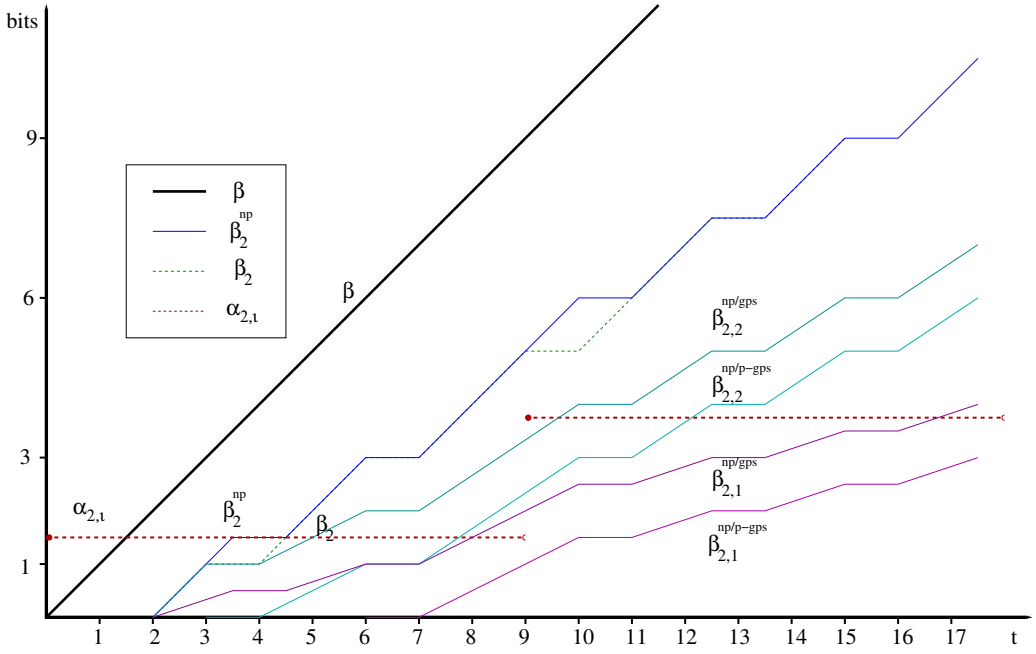

Figure 8. Residual services for $R_{1,2}$ and $R_{2,2}$

[2] C. Cheng-Shang, *Performance Guarantees in Communication Networks*, volume ISBN : 1-85233-226-3, Springer-Verlag, 2000.

[3] J. Grieu, *Analyse et évaluation de techniques de commutation Ethernet pour l'interconnexion des systèmes avioniques*, Thèse de Doctorat, Institut National Polytechnique de Toulouse, Toulouse, 2004.

[4] A. Bouillard, L. Jouhet, and E. Thierry, "Service curves in Network Calculus: dos and don'ts", Rapport de recherche INRIA 7094, INRIA, Novembre 2009.

[5] W. Haid and L. Thiele, "Complex task activation schemes in system level performance analysis", in *ESWeek'07: Proc. of the 5th IEEE/ACM int. conf. on Hardware/Software Codesign and System Synthesis (Salzburg, Austria, September 30 - October 03, 2007)*, 2007, pp. 173–178, New York, NY, USA. ACM.

[6] D. B. Chokshi and P. Bhaduri, "Modeling Fixed Priority Non-Preemptive Scheduling with Real-Time Calculus", in *RTCSA '08: Proc. of the 2008 14th IEEE int. conf. on Embedded and Real-Time Computing Systems and Applications*, 2008, Washington, DC, USA. IEEE Computer Society.

[7] W. Mangoua Sofack and M. Boyer, "Non preemptive static priority with network calculus", in *Proc. of the 16th IEEE int. conf. on Emerging Technologies and Factory Automation (ETFA'11)*, September 2011.

[8] W. Mangoua Sofack and M. Boyer, "Non preemptive static priority with network calculus: Enhanced", in *Proc. of the 16th International GI/ITG Conference on Measurement, Modelling and Evaluation of Computing Systems and Dependability and Fault Tolerance (MMB & DFT 2012) – Workshop on Network Calculus (WoNeCa)*, March 2012.

[9] A. K. Parekh and R. G. Gallager, "A generalized processor sharing approach to flow control in integrated services networks: the single-node case", *IEEE/ACM Transactions on Networking (TON)*, vol. 1, 1993.

[10] V. Pollex, H. Lipskoch, F. Slomka, and S. Kollmann, "Runtime improved computation of path latencies with the real-time calculus", in *Proc. of the 1st International Workshop on Worst-Case Traversal Time*, WCTT '11, 2011, pp. 58–65. ACM.

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

[12] R. I. Davis, A. Burns, R. J. Bril, and J. J. Lukkien, "Controller Area Network (CAN) Schedulability Analysis: Refuted, Revisited and Revised", 2007.

[13] M. Boyer, J. Migge, and M. Fumey, "PEGASE, A Robust and Efficient Tool for Worst Case Network Traversal Time", in *Proc. of the SAE 2011 AeroTech Congress & Exhibition*, 2011, Toulouse, France. SAE International.

[14] E. Wandeler, *Modular Performance Analysis and Interface-Based Design for Embedded Real-Time Systems*, PhD thesis, PhD Thesis ETH Zurich, 2006.

[15] A. Bouillard, N. Farhi, and B. Gaujal, "Packetization and Aggregate Scheduling", Technical Report 7685, INRIA, 2011.

A. Proof of Theorem 4

We demonstrate here the first lemma, the first and second theorem of [9] in a broader context ($\beta \in \mathcal{F}$).

Lemma 7. *Let p and p' be packets in a GPS system at time τ, and suppose that packet p completes service before packet p' if there are no arrivals after time τ. Then, packet p will also complete service before packet p' for any pattern of arrivals after time τ.*

Proof. Idem as in [9].
The sessions to which packets p and p belong are both backlogged from time r until one completes transmission. By (2), the ratio of the service received by these sessions is independent of future arrivals. □

Lemma 8. *Now let F_p (resp. $\hat{F}_p$) be the time at which packet p departs under GPS (resp. P-GPS). For all packets p, $F_p - \hat{F}_p \leq \bar{P}$ where $\bar{P}$ is an upper bound on the processing time of a packet of size $L_{\max}$*

Proof. Since both GPS and P-GPS are work-conserving disciplines, their busy periods coincide, i.e. the GPS server is in a busy period iff the PGPS server is in a busy period. Hence, it suffices to prove the result for each busy period. Consider any busy period and let the time that it begins be time zero. Let p_k be the k^{th} packet in the busy period to depart under P-GPS, and let its length be L_k. Also, let t_k be the time that p_k departs under P-GPS and u_k be the time that p_k departs under GPS. Finally, let a_k be the time that p_k arrives. Let us show that

$$t_k \leq u_k + \bar{P}$$

for $k = 1, 2, \ldots$ Let m be the largest integer that satisfies both $0 < m \leq k-1$ and $u_m > u_k$. Thus,

$$u_m > u_k \geq u_i \text{ for } m < i < k$$

Then, packet p_m is transmitted before packets $p_{m+1}, \ldots, p_k$ under P-GPS but after all these packets under GPS. If no such integer m exists, then set $m = 0$. Now, for the case $m > 0$, packet p_m begins transmission at $t_m - trans(m)$, ($trans(m)$ is the transmission time of p_m); so, from Lemma 7,

$$\min\{a_{m+1}, \ldots, a_k\} > t_m - trans(m)$$

Since $p_{m+1}, \ldots, p_{k-1}$ arrive after $t_m - trans(m)$ and depart before p_k does under GPS,

$$u_k \geq (trans(m+1) + \cdots + trans(k-1) + trans(k)) + t_m - trans(m)$$

$$\begin{aligned} \implies u_k \geq & t_k - trans(m) \\ \geq & t_k - \bar{P} \end{aligned}$$

If $m = 0$, then $p_{k-1}, \ldots, p_1$ all leave the GPS server before p_k does, and so

$$u_k \geq t_k$$

□

Now comes the proof of the main theorem.

Proof of Theorem 4. The slope of $\hat{R}'_i$ alternates between the maximum capacity of the server when a session i packet is being transmitted, and 0 when session i is not being served. Since the slope of R'_i also obeys these limits, the difference $(R'_i(\tau) - R'_i(0)) - (\hat{R}'_i(\tau) - \hat{R}'_i(0))$ reaches its maximal value when session i packets begin transmission under P-GPS. Let t be some such time, and let L be the length of the packet p going into service. Then, the packet completes transmission at time $t + trans(p)$. Let ρ be the time at which the given packet completes transmission under GPS. Then, since session i packets are served in the same order under both schemes,

$$(R'_i(\rho) - R'_i(0)) = (\hat{R}'_i(t + trans(p)) - \hat{R}'_i(0)) \quad (7)$$

From Theorem 8 (with $F_p = \rho$ and $\hat{F}_p = t + trans(p)$),

$$\begin{aligned} & t + trans(p) - \tau \leq \bar{P} \\ \implies & \rho \geq t + trans(p) - \bar{P} \end{aligned} \quad (8)$$

Then,

$$\begin{aligned} & (R'_i(t + trans(p) - \bar{P}) - R'_i(0)) \\ & \leq (R'_i(\rho) - R'_i(0)) \text{ (from eq. 8)} \\ & = (\hat{R}'_i(t + trans(p)) - \hat{R}'_i(0)) \text{ (from eq. 7)} \\ & = (\hat{R}'_i(t) - \hat{R}'_i(0)) + L \text{ (from the definition of } trans(p)) \end{aligned}$$

Then,

$$(R'_i(t + trans(p) - \bar{P}) - R'_i(0)) \leq (\hat{R}'_i(t) - \hat{R}'_i(0)) + L \quad (9)$$

Let $\tilde{L}$ the length of the packet $\tilde{p}$ that requires the largest transmission time. With $\bar{P} = trans(\tilde{p})$, eq.9 becomes

$$\begin{aligned} (R'_i(t) - R'_i(0)) \leq & (\hat{R}'_i(t) - \hat{R}'_i(0)) + \tilde{L} \\ \leq & (\hat{R}'_i(t) - \hat{R}'_i(0)) + L_{\max} \end{aligned}$$

Then,

$$(R'_i(t) - R'_i(0)) - (\hat{R}'_i(t) - \hat{R}'_i(0)) \leq L_{\max} \quad (10)$$

Since t is a time where the difference $(R'_i(\tau) - R'_i(0)) - (\hat{R}'_i(\tau) - \hat{R}'_i(0))$ reaches its maximal value, with the eq. 10, we can write $\forall$ times τ and sessions i:

$$(R'_i(\tau) - R'_i(0)) - (\hat{R}'_i(\tau) - \hat{R}'_i(0)) \leq L_{\max}$$

□

Note for reviewers

The reference [8] have been accepted, but it not published up to now. We can of course send a copy of this paper.