Optimal routing for end-to-end guarantees using Network Calculus

Anne Bouillard, Bruno Gaujal, Sébastien Lagrange, Éric Thierry

INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Optimal routing for end-to-end guarantees using Network Calculus

Anne.Bouillard and Bruno Gaujal and Sébastien Lagrange and Éric Thierry

N° ????

Janvier 2008

Thème COM

ISSN 0249-6399 ISRN INRIA/RR--????--FR+ENG

Optimal routing for end-to-end guarantees using Network Calculus

Anne.Bouillard * and Bruno Gaujal † and Sébastien Lagrange ‡ and Éric Thierry §

Thème COM — Systèmes communicants
Projet Distribcom

Rapport de recherche n° ???? — Janvier 2008 — 20 pages

Abstract: In this paper we show how Network Calculus can be used to compute the optimal route for a flow (w.r.t. end-to-end guarantees on the delay or the backlog) in a network in the presence of cross-traffic. When cross-traffic is independent, the computation is shown to boild down to a functional shortest path problem. When cross-traffic perturbates the main flow over more than one node, then the "Pay Multiplexing Only Once" phenomenon makes the computation more involved. We provide an efficient algorithm to compute the service curve available for the main flow and show how to adapt the shortest path algorithm in this case.

Key-words: Network Calculus, multiplexing, shortest-path.

* Anne.Bouillard@bretagne.ens-cachan.fr
† Bruno.Gaujal@imag.fr
‡ Sebastien.Lagrange@istia.univ-angers.fr
§ Eric.Thierry@ens-lyon.fr

Unité de recherche INRIA Rennes
IRISA, Campus universitaire de Beaulieu, 35042 Rennes Cedex (France)
Téléphone : +33 2 99 84 71 00 — Télécopie : +33 2 99 84 71 71

Routage optimal pour le calcul de garanties de performances de bout en bout dans les réseaux

Résumé : Dans ce rapport, nous montrons comment le *Network Calculus* peut être utilisé pour calculer une route optimale pour un flot, quant à ses garanties de performances pour le délai de bout en bout ou le nombre de paquets, dans un réseau en présence de trafic transverse. Quand le trafic transverse est indépendant, on peut se ramener à un calcul de plus court chemin dans un graphe pondéré par des fonctions. Quand le trafic transverse perturbe le flot principal sur plus d'un nœud, le phonomène *pay multiplexing only once* rend les calculs plus complexes. Nous fournissons un algorithme efficace pour calculer la courbe de service disponible pour le flot principal et montrons comment adapter l'algorithme de plus court chemin dans ce cas.

Mots-clés : Network Calculus, multiplexage, plus court chemins.

1 Introduction

Optimizing the route of a flow of packets through a network has been investigated in many directions and using many approaches depending on the assumptions made on the system as well as the performance objectives. When one wants to maximize the throughput of one connection, most recent results in deterministic contexts use multi-flow or LP techniques [3, 4], or optimal control and/or game theory in a stochastic one as for example in [1].

When the maximal delay over all packets in the flow is the performance index, fewer results are available in the litterature. Under static assumptions on the flows and the network ressources, optimal bandwidth allocation has been investigated in [8]. However, when the flows and the ressources have dynamic features, most focus is on simple systems such as single nodes where the issue becomes optimal scheduling.

Here we consider the problem of computing the route of a flow that provides the best *delay guarantee* $D_{\max}$ (no packet of the flow will ever spend more than $D_{\max}$ seconds in the system) or *backlog guarantee* $B_{\max}$ (the number of packets of the flow inside the network never tops $B_{\max}$), in the presence of cross-traffic. Network Calculus [7, 11] is a framework that allows us to formulate this problem as a mathematical program.

In the first part of this paper, we show how to compute the best route for one flow from source to destination over an arbitrary network when the cross-traffic in each node is independent. Using the network calculus framework, we show that this boils down to solving a classical shortest path problem using appropriate costs at each node, as soon as the service curves are piecewise affine and convex and arrival curves are concave, which are classical assumptions in Network Calculus.

The second part of the paper considers the more realistic case where cross-traffic in each node is not independent. This happens when several flows follow the same sub-paths over more than two nodes or when the main flow crosses the same cross-traffic several times. This case is much harder to solve because of the "Pay Multiplexing Only Once" (PMOO) phenomenon, which was first identified in [10]. When the main flow merges with a cross-traffic, its service might be strongly reduced in the first node. However, in the following nodes, the interference due to the cross-traffic cannot be as severe since the competition for the ressource has already been partially resolved in the previous ones. The PMOO phenomenon can be quantified in the Network Calculus context. It does provide good bounds on performance guarantees but this comes with a price:

- In that case we only tackle efficiently networks with a strong acyclicity property (introduced in this paper). In fact, computing tight guarantees in cyclic networks is still open (the simpler problem of stability is also open [2]).
- The algorithms involved have much higher complexities.

For single paths, the approach in [13] provides an example showing how to compute the global service curve for a single path with 2 cross-traffic flows. When the service curve in each node is piecewise affine, then the algorithm provided in [13] is based on a decompostion in affine functions. The complexity grows exponentially with the number of cross-traffic flows and the number of nodes in the path. Here, we provide an explicit general formula for the PMOO phenomenon for arbitrary cross-traffic. The global service curve is written under the form of a multi-dimensional convolution which helps designing an algorithm to compute it with a sub-quadratic complexity. For routing problems, this single path computation can be applied to find the best route in an acyclic network, taking into account PMOO. Under stronger assumptions (affine functions, concentration of the cross-traffic), we show how to speed up the best route computation by reducing the problem once more to classical shortest path algorithms.

This paper is a long version of [5], providing detailed proofs of all the results as well as several extensions. The most important one is the new algorithm provided in Section 3. It allows one to compute the best route for backlog and delay minimization under concave/convex assumptions on the arrival/service curves. In [5] more direct algorithms were provided, but they worked on more restricted types of curves: backlog minimization assumed rate-latency service curves or affine arrival curves, and delay minimization assumed affine arrival curves. A detailed overview of the complexity of all these algorithms is provided in Table 3.3.

2 Performances guarantees

In this section, we recall the main definitions and the main properties of the Network Calculus functions and operations. More precise insights can be found in [7, 11].

2.1 Network Calculus functions

Network Calculus is based on the $(\min, +)$ algebra and models flows and services in a network with non-decreasing functions taking their values in the $(\min, +)$ semiring.

Formally, the $(\min, +)$ semiring, denoted by $(\mathbb{R}_{\min}, \oplus, \otimes)$ is defined on $\mathbb{R}_{\min} = \mathbb{R} \cup \{+\infty\}$, and is equipped with two internal operations: $\oplus$, the minimum, and $\otimes$, the addition. The zero element is $+\infty$, the unitary element is 0. The $\oplus$ and $\otimes$ operators are commutative and associative. Moreover $\otimes$ is distributive over $\oplus$.

Consider the set $\mathcal{F}$ of functions from $\mathbb{R}_+$ into $\mathbb{R}_{\min}$. One can define as follows two operators on $\mathcal{F}$, the minimum, denoted by $\oplus$, and the (min,+) convolution, denoted by $*$: for all f, g in $\mathcal{F}$, $\forall t \in \mathbb{R}_+$,

- $f \oplus g(t) = f(t) \oplus g(t)$ and
- $f * g(t) = \inf_{0 \le s \le t}(f(s) + g(t-s))$.

The triple $(\mathcal{F}, \oplus, *)$ is also a semiring and the convolution can be seen as an analogue to the classical $(+,\times)$ convolution of filtering theory, transposed in the (min,+) algebra. Another important operator for Network Calculus is the $(\max, +)$ deconvolution, denoted by $\oslash$: let $f, g \in \mathcal{F}$, $\forall t \ge 0$,

- $f \oslash g(t) = \sup_{u \ge 0}(f(t+u) - g(u))$.

2.2 Arrival and service curves

Arrival curves. Given a data flow traversing a system, let A be its *cumulative arrival function*, i.e. $A(t)$ is the number of packets that have arrived until time t. We say that α is an *arrival curve* for A (or that A is upper-constrained by α) if $\forall\ 0 \le s \le t \in \mathbb{R}_+$, $A(t) - A(s) \le \alpha(t-s)$. This means that the number of packets arriving between time s and t is at most $\alpha(t-s)$. An important particular case of arrival curves are the affine functions: $\alpha(t) = \sigma + \rho t$. Then σ represents the maximal number of packets that can arrive simultaneously (the maximal burst) and ρ the maximal long-term rate of arrivals.

Service curves. Consider B the *cumulative departure function* of the flow, defined similarly by the number $B(t)$ of packets that have left the system until time t. The system provides a (minimum) *service curve* β if $B \ge A * \beta$. Particular cases of service curves are the *peak rate* functions with rate r (the system can serve r packets per unit of time and $\beta(t) = rt$) and the *pure delay* service curves with delay T: $\beta(t) = 0$ if $t < T$ and $\beta(t) = +\infty$ otherwise. The combination of those two service curves gives a *rate-latency* function $\beta(t) = r(t-T)_+$ where a_+ denotes $\max(a, 0)$. A *strict service curve* β is a service curve such that for all $t \in \mathbb{R}_+$, let $u < t$ be the last instant before t when there is no packet in the system i.e. $B(u) = A(u)$, then $B(t) \ge B(u) + \beta(t-u)$. This enforcement of the service curve notion is necessary to have refined bounds (e.g. positiveness of output service curves in Lemma 2 and Theorem 2). Note that contrary to a statement of [5], if a service curve is convex, it is not necessarily a strict service curve. Consequently, all along the paper, we will detail whenever service curves are assumed to be strict or not.

We also consider that for any service curve β, $\beta(0) = 0$: there is no instantaneous service.

2.3 Performance characteristics and bounds

The worst case backlog and the delay can be easily characterized with Network Calculus.

Definition 1. *Let A be the arrival function of a flow through a system and B be its corresponding departure function. Then the* backlog *of the flow at time t is*

$$b(t) = A(t) - B(t)$$

and the delay (assuming FIFO order for serving packets of the flow) at time t is

$$d(t) = \inf\{s \ge 0 \mid A(t) \le B(t+s)\}.$$

Given an arrival curve and a service curve, it is possible to compute with the Network Calculus operations the maximal backlog and delay. Moreover, one can also compute the arrival curve of the departure process.

Theorem 1 ([7, 11])**.** *Let A be the arrival function with an arrival curve α for a flow entering a system with service curve β. Let B be the departure function. Then,*

1. *B has an arrival curve $\alpha \oslash \beta$.*

2. $b(t) \leq B_{\max}(\alpha, \beta) \stackrel{\text{def}}{=} \sup\{\alpha(t) - \beta(t) \mid t \geq 0\} = (\alpha \oslash \beta)(0)$.

3. $d(t) \leq D_{\max}(\alpha, \beta) \stackrel{\text{def}}{=} \inf\{d \geq 0 \mid \forall t \geq 0,\ \alpha(t) \leq \beta(t+d)\}$
$= \inf\{d \geq 0 \mid (-\beta) \oslash (-\alpha)(d) \leq 0\}$.

The maximal backlog is the maximal vertical distance between α and β while the maximal delay is given by the maximal horizontal distance between those two functions. Figure 1 illustrates this fact.

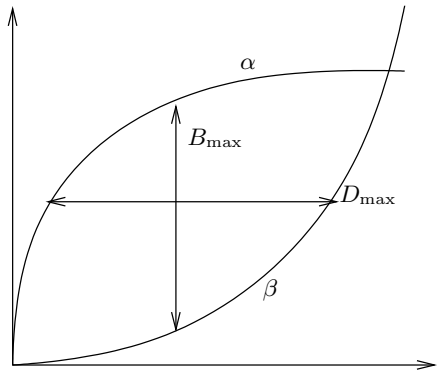

Figure 1: Guarantee bounds on backlog and delay.

In this paper, we are interested in computing bounds for end-to-end guarantees in networks of servers, where several flows can interfere. A network can be modeled, with no loss of generality, by a directed graph where the flows must follow the arcs and the servers (commuters, transmission links, routers...) are represented by the vertices.

The two following lemmas are very useful for computing service curves for concatenation of servers and for blind multiplexing of flows.

Lemma 1 ([7, 11])**.** *Consider two servers in tandem with respective service curves β_1 and β_2. Then the concatenation of the two servers offers a minimum service curve $\beta_1 * \beta_2$ to the flow.*

Lemma 2 ([7, 11])**.** *Consider a server offering a strict service curve β and two flows entering that server, with respective arrival curves α_1 and α_2. Then a service curve for flow 1 is $\beta_1 = (\beta - \alpha_2)_+$.*

The next example illustrates some Network Calculus computations for usual input functions. We will make an intensive use of those elementary results throughout the paper.

Example 1 ([11])**.** *Let $\alpha(t) = \sigma + \rho t$, $\beta(t) = R(t-T)_+$ where $\sigma, \rho, R, T \geq 0$. Then $(\alpha \oslash \beta)(t) = (\sigma + \rho T) + \rho t$ if $\rho \leq R$ and $= +\infty$ everywhere if $\rho > R$. And $(\alpha - \beta)_+(t) = (R-\rho)(t-T')_+$ with T' is $\frac{\sigma+\rho T}{R-\rho}$ if $\rho < R$ and $= 0$ everywhere if $\rho \geq R$. Let $\beta_i(t) = R_i(t-T_i)_+$, $i \in \{1,2\}$. Then $(\beta_1 * \beta_2)(t) = \min(R_1, R_2)(t - (T_1 + T_2))_+$.*

2.4 Representation of the functions

Our main objective is to find algorithms that enable to do some computations in Network Calculus. Then, we have to consider the implementability of the Network Calculus operations and functions. This question is addressed in [6] where it is shown that Network Calculus operations can be performed efficiently on a good class of functions with a finite representation: the piecewise affine functions that are ultimately pseudo-periodic.

Here, we will actually work on a little more restricted class of functions: piecewise affine functions with a finite number of segments and which may have $+\infty$ value from a point. Such a function can be represented for instance by a linked list of triples, each triple representing a segment. A triple (x, y, ρ) can represent the coordinates (x, y) of the beginning of the segment, and ρ its slope. The end of the segment is given by the next triple and the last triple represents either a last segment which is of infinite length or a range from which the function is $+\infty$ (it can be denoted by the triple $(x, +\infty, +\infty)$). Let f be such a piecewise affine function, $|f|$ denotes the *size* of f, i.e. its number of segments. The complexity of the algorithms will strongly depend on the size of the functions.

We will also always suppose that the networks are stable, that is the total number of packets in the servers never grows to infinite. For the class of functions we use as arrival and service curves, checking the stability of a network is easy if the directed graph is acyclic [11]: at each vertex, the long-term rate of arrivals must be less than the long-term service rate, i.e. the sum over the flows entering the vertex of the ultimate slopes of their arrival curves must be less than the ultimate slope of the service curve. For general digraphs, the complexity of this decision problem is open [2].

3 Optimal routing with independent cross-traffic

In this section, we wish to route one flow over an arbitrary network. Each vertex may or may not be subject to interference due to independent cross-traffic: the cross-traffic in any two vertices are not correlated. We want to find a path from the source v_{in} of the flow to its destination v_{out}, that optimizes the end-to-end performance guarantees for that flow, given the service curves at each vertex and the arrival curves of that flow and of the cross-traffic.

Here is a survey of the main assumptions we will consider. All functions are non-decreasing. We suppose that arrival curves are *concave*. It is an usual assumption. Even when it is not fullfilled, an arrival curve α can always be replaced by its subadditive closure α^* [11] for which $\alpha^*(t)/t$ converges to a finite value when $t \to +\infty$ (as any subadditive function). Then one can choose to use the concave envelope of α^* as arrival curve for the flow. This is asymptotically tight with regard to α^*.

We suppose that the service curves are *convex*. It is also an usual assumption. Since we allow infinite values, note that a function which is convex over $[0, T]$ and equal to $+\infty$ over $]T, +\infty[$ is convex over $\mathbb{R}_+$. If the service curves are also *strict*, we can take into account the cross-traffic with blind multiplexing: if vertex v offers a service curve β_v^0 and the cross-traffic in vertex v has arrival curve α_v, then we can use $\beta_v = (\beta_v^0 - \alpha_v)_+$ as a service curve for the main flow, as stated in Lemma 2, and if β_v^0 is convex and α_v is concave, then β_v is convex. Such a reduction is totally appropriate for independent cross-traffic. Once this is done, no mention of the cross-traffic is necessary any longer in this section.

The general routing problem we consider in this section is:

> *Given a directed graph $G = (V, A)$ with a service curve β_v for each $v \in V$ and some flow specifications, namely its source $v_{in} \in V$, its destination $v_{out} \in V$ and an arrival curve α, compute a path from v_{in} to v_{out} such that the worst case delay (or backlog) for the flow is minimal.*

In graph theory, one can mention two classical versions of optimal routing. With arcs and/or vertices weighted by numbers, the first one consists in finding a classical shortest path from one source to one destination (minimizing the sum of the weights of the path), and the latter one is to find a path with maximum bottleneck capacity (maximizing the smallest weight of the path). Those two problems can be seen as special cases of our problem, when respectively the service curves β_v are all pure delays or are all peak rates (see Lemma 6 explaining how these functions behave with regard to the convolution).

We first state some general lemmas about computing maximal backlog and delay for concave arrival curves and convex service curves. Then we describe a polynomial algorithm to solve the optimal routing problem for delays and backlogs

3.1 Concave arrival curves and convex service curves

Some specific results apply when arrival/service curves are concave/convex. The first one is a simple min-max theorem concerning the maximal delay or backlog.

We first define a notion of *stripe*. Let $\lambda \in \mathbb{R}_+ \cup \{+\infty\}$, a *$\lambda$-stripe* S is a subset of $\mathbb{R}^2$ of the form $\{(x, y) \in \mathbb{R}^2 \mid y_{\min} + \lambda x \leq y \leq y_{\max} + \lambda x\}$ for $y_{\min} \leq y_{\max}$ (and of the form $\{(x, y) \in \mathbb{R}^2 \mid x_{\min} \leq x \leq x_{\max} + \lambda x\}$ for $x_{\min} \leq x_{\max}$ if $\lambda = +\infty$). In other words, a λ-stripe S is the stripe between two lines of slope λ. We denote $hsize(S)$ (resp. $vsize(S)$, and $width(S)$) the horizontal (resp. vertical, and euclidian) distance between these two border lines. We set $hsize(S) = +\infty$ (resp. $vsize(S) = +\infty$) if $\lambda = 0$ (resp. $\lambda = +\infty$), and we clearly have $vsize(S) = width(S)\sqrt{1 + \lambda^2}$ and $hsize(S) = width(S)\frac{\sqrt{1+\lambda^2}}{\lambda}$.

Lemma 3. *Let α be a non-decreasing concave function, β a non-decreasing convex function and let $\mathcal{I}(\alpha, \beta) \stackrel{\text{def}}{=} \{(x, y) \in \mathbb{R}_+^2 \mid \beta(x) \leq y \leq \alpha(x)\}$ the area between the two curves. Then for a flow with arrival curve α entering*

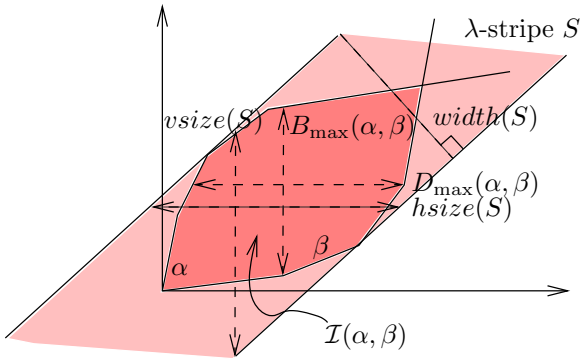

Figure 2: An arrival curve α, a sevice curve β and a stripe S containing the area $\mathcal{I}(\alpha,\beta)$. The quantities $width(S)$, $hsize(S)$, $D_{\max}(\alpha,\beta)$ and $vsize(S)$, $D_{\max}(\alpha,\beta)$ are given in the figure.

a node with service curve β, the maximal delay and maximal backlog satisfy:

$$D_{\max}(\alpha,\beta) = \min\{hsize(S) \mid S \text{ is a } \lambda\text{−stripe, } \lambda \in \mathbb{R}_+ \cup \{+\infty\}, \ S \supseteq \mathcal{I}(\alpha,\beta)\}$$

$$B_{\max}(\alpha,\beta) = \min\{vsize(S) \mid S \text{ is a } \lambda\text{−stripe, } \lambda \in \mathbb{R}_+ \cup \{+\infty\}, \ S \supseteq \mathcal{I}(\alpha,\beta)\}$$

Proof. Since α is concave and β is convex, then $\alpha - \beta$ is concave with value 0 at 0.
If $\alpha-\beta$ remains non-negative then $\mathcal{I}(\alpha,\beta)$ is infinite and no stripe of finite width can contain it. By considering that $\min\emptyset = +\infty$, one gets $D_{\max}(\alpha,\beta) = B_{\max}(\alpha,\beta) = \infty$. This ends the proof in that case.
If α is always smaller than β, then $\mathcal{I}(\alpha,\beta) = \{(0,0)\}$ and all stripes containing $(0,0)$ will do. In particular, a stripe reduced to a line with an arbitrary slope and containing $(0,0)$ has a null width and provides the equalities $D_{\max}(\alpha,\beta) = B_{\max}(\alpha,\beta) = 0$.
The only interesting case is when $\alpha-\beta$ is larger than 0 up to some point t_0 and is non-positive from that point on. From the definitions of $D_{\max}(\alpha,\beta)$ and $B_{\max}(\alpha,\beta)$, the inequalities

$$\begin{aligned} D_{\max}(\alpha,\beta) &\leq \min\{hsize(S) \mid S \text{ is a } \lambda\text{−stripe,, } S \supseteq \mathcal{I}(\alpha,\beta)\} \\ B_{\max}(\alpha,\beta) &\leq \min\{vsize(S) \mid S \text{ is a } \lambda\text{−stripe,, } S \supseteq \mathcal{I}(\alpha,\beta)\}. \end{aligned}$$

are clearly true as seen on Figure 2.
For the other inequality, let us first carry the proof for B_{max}. By concavity, there exists a finite time t_1 such that $B_{\max}(\alpha,\beta) = \alpha(t_1) - \beta(t_1)$. Since the function $\alpha-\beta$ is piecewise affine, let us call by $d_+(\alpha-\beta)(t_1)$ (resp. $d_-(\alpha-\beta)(t_1)$) its slope just after t_1 (resp. just before t_1). The function $\alpha-\beta$ being maximal at t_1, then $d_+(\alpha-\beta)(t_1) = d_+(\alpha)(t_1) - d_+(\beta)(t_1) \leq 0$ and $d_-(\alpha-\beta)(t_1) = d_-(\alpha)(t_1) - d_-(\beta)(t_1) \geq 0$. Moreover by convexity and concavity of β and α, $d_+(\beta)(t_1) \geq d_-(\beta)(t_1)$ and $d_-(\alpha)(t_1) \geq d_+(\alpha)(t_1)$. Combining all those inequalities says that $d_+(\beta)(t_1)$ and $d_-(\alpha)(t_1)$ are both larger than $d_-(\beta)(t_1)$ and $d_+(\alpha)(t_1)$. Consider any slope λ such that

$$\min(d_+(\beta)(t_1), d_-(\alpha)(t_1)) \geq \lambda \geq \max(d_-(\beta)(t_1), d_+(\alpha)(t_1)).$$

Then, the stripe of slope λ containing the points $(t_1,\alpha(t_1))$ and $(t_1,\beta(t_1))$ is of vertical size $\alpha(t_1)-\beta(t_1)$, equal to $B_{\max}(\alpha,\beta)$ by definition of t_1 and contains $\mathcal{I}(\alpha,\beta)$, because from point $(t_1,\alpha(t_1))$ all the slopes of α and β compare well with λ. This ends the proof for $B_{\max}$.
As for horizontal distance $D_{\max}$, the proof is essentially the same by considering the vertical distance between α^{-1} and β^{-1}. □

Lemma 4. *With the assumptions of Lemma 3, suppose that α (resp. β) is piecewise affine with a finite number of segments of slopes $r_0 \geq r_1 \geq \cdots \geq r_p$ (resp. $\rho_1 \leq \cdots \leq \rho_m$) with $r_0 \stackrel{\text{def}}{=} +\infty$ corresponding to the segment from $(0,0)$ to $(0, f(0+))$ and with $\rho_m \stackrel{\text{def}}{=} +\infty$ if $\beta = +\infty$ from a point. Then*

$$D_{\max}(\alpha,\beta) = \min_{\lambda \in \{r_0,\ldots,r_p\}\cup\{\rho_1,\ldots,\rho_m\}} \{hsize(S) \mid S \text{ is a } \lambda\text{−stripe, } S \supseteq \mathcal{I}(\alpha,\beta)\}$$

$$B_{\max}(\alpha,\beta) = \min_{\lambda \in \{r_0,\ldots,r_p\}\cup\{\rho_1,\ldots,\rho_m\}} \{vsize(S) \mid S \text{ is a } \lambda\text{−stripe, } S \supseteq \mathcal{I}(\alpha,\beta)\}$$

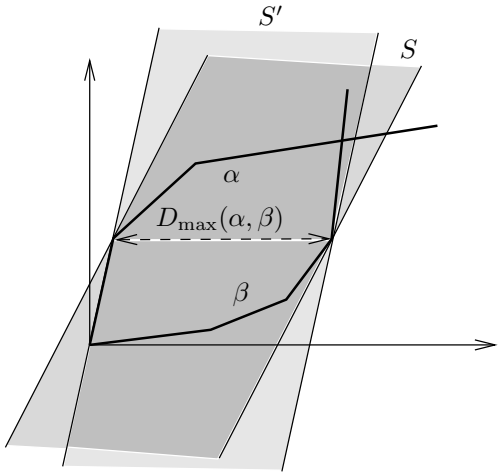

Figure 3: A stripe S with horizontal size equal to $D_{\max}(\alpha,\beta)$. By tilting S, one gets S' a stripe with the same size as S whose slope is commum with α (or β).

Proof. We only provide the proof for $D_{\max}(\alpha,\beta)$. The same argument holds for $B_{\max}(\alpha,\beta)$. Assume that $D_{\max}(\alpha,\beta)$ is the horizontal size of a stripe S is $D_{\max}(\alpha,\beta)$. It should be clear that the stripe touches both α and β (otherwise, a stripe with the same slope and a smaller horizontal size can be constructed). Figure 3 displays such an example. From S let us construct a new stripe S' with the same horizontal size as S by fixing the two contact points with α and β and letting λ increase as long as the stripe contains $\mathcal{I}(\alpha,\beta)$. The slope of the new tilted stripe S' must be the slope of one of the segments of α and β (see Figure 3). □

If both α and β have a finite number of segments, and if the slope λ is fixed, then the minimum value of $width(S)$, $hsize(S)$ and $vsize(S)$ over λ-stripes is given by a simple formula involving projections along the direction λ. Let $\lambda \in \mathbb{R}_+ \cup \{+\infty\}$, we denote

$$\begin{aligned} W_\lambda(\alpha,\beta) &\stackrel{\text{def}}{=} \min\{width(S) \mid S \text{ is a } \lambda\text{−stripe},\ S \supseteq \mathcal{I}(\alpha,\beta)\} \\ H_\lambda(\alpha,\beta) &\stackrel{\text{def}}{=} \min\{hsize(S) \mid S \text{ is a } \lambda\text{−stripe},\ S \supseteq \mathcal{I}(\alpha,\beta)\} \\ V_\lambda(\alpha,\beta) &\stackrel{\text{def}}{=} \min\{vsize(S) \mid S \text{ is a } \lambda\text{−stripe},\ S \supseteq \mathcal{I}(\alpha,\beta)\}. \end{aligned}$$

We also define the orthogonal projection $proj_\lambda$ parallel to the line $y = \lambda x$ (the line $x = 0$ if $\lambda = +\infty$) which associates with each segment $\mathfrak{s}$ of $\mathbb{R}^2$ of slope $r \in \mathbb{R}_+ \cup \{+\infty\}$ and length $\ell \in \mathbb{R}_+ \cup \{+\infty\}$ the value[1]:

$$proj_\lambda(\mathfrak{s}) = \frac{\lambda - r}{\sqrt{1+\lambda^2}\sqrt{1+r^2}}\ell.$$

Lemma 5. *With the assumptions of Lemma 4 and the notation above, let α_i, $0 \le i \le p$ (resp. β_j, $1 \le j \le m$), be the segments composing α (resp. β). Let $\lambda \in \mathbb{R}_+ \cup \{+\infty\}$, then:*

$$H_\lambda = W_\lambda \frac{\sqrt{1+\lambda^2}}{\lambda}\ ,\ V_\lambda = W_\lambda\sqrt{1+\lambda^2},$$

$$W_\lambda = \sum_{i=0}^{p}(-proj_\lambda(\alpha_i))_+ + \sum_{j=1}^{m}(proj_\lambda(\beta_j))_+$$

where x_+ denotes $\max(x,0)$.

Proof. The equalities linking H_λ, V_λ and W_λ, are direct consequences of the same equalities mentionned before and linking $hsize(S)$, $vsize(S)$ and $width(S)$ for any λ-stripe S.

The formula for W_λ is illustrated by Figure 4. It can be easily checked thatW_λ is the width of the orthogonal projection along direction λ of the set $\mathcal{I}(\alpha,\beta)$ which is convex since α is concave and β convex. The first sum

[1] with the convention that $0 \times \infty = 0$.

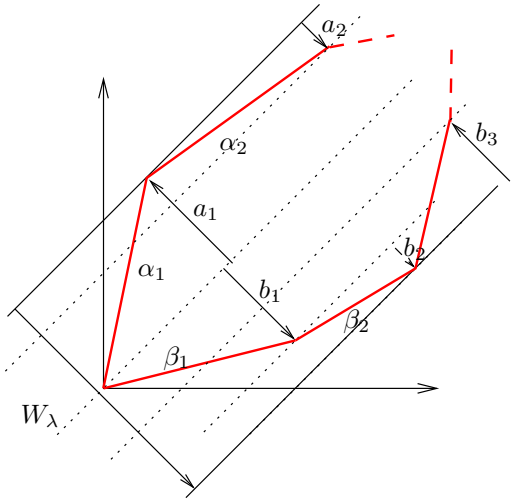

Figure 4: The projection along direction λ of the segments of α_1 and α_2 give the quantities $a_1(<0)$ and $a_2(>0)$ respectively. All the other segments of α have slopes smaller than λ so that they are projected on positive quantities. The projection of $\beta_1, \beta_2, \beta_3$ are $b_1(>0), b_2(>0), b_3(<0)$ respectively. All the other segments of β have slopes larger than λ so that they are projected on negative quantities. One can verify on the figure that $W_\lambda = (-a_1)_+ + (-a_2)_+ + \sum_{i>2}(-a_i)_+ + b_{1+} + b_{2+} + b_{3+} + \sum_{i>3}(b_i)_+ = -a_1 + b_1 + b_2$.

corresponds to the concave curve α. Due to its shape, the projection of α only takes into account the first part of the curve where the segments have a slope at least λ (the maximum with 0 is used to retain only the contribution of those segments). It is clearly additive with regard to the concatenation of the segments α_i. Similarly the second sum represents the projection of the first part of the convex curve β where the segments have a slope at most λ. □

Note that the formula of Lemma 5 is a sum. Due to commutativity, it means that it is not necessary to give the list of the segments α_i (resp. β_j) ordered by non-increasing (resp. non-decreasing) slopes.

3.2 Transforming service curves into elementary service curves

Before presenting an algorithm which solves the optimal routing problem when the arrival curve of the routed flow is concave and the services curves are convex, we show how to transform the input network so that the service curves we manipulate all have the same elementary shape, that is a single segment.

An *elementary segment* β is a finite or semi-infinite segment over $\mathbb{R}_+$ such that $\beta(0) = 0$. More precisely, either $\beta(t) = rt$ when $0 \leq t \leq T$ and $= +\infty$ when $t > T$ for some $r, T \in \mathbb{R}_+$ (*finite elementary segment*) or $\beta(t) = rt$ on $\mathbb{R}_+$ for some $r \in \mathbb{R}_+$ (*semi-infinite elementary segment*).

Any non-decreasing, convex, piecewise affine function β over $\mathbb{R}_+$ such that $\beta(0) = 0$ and with a finite number of segments, is equal to the convolution of these segments all translated to point $(0, 0)$ to be elementary segments. This result is a direct consequence of the following well-known lemma describing the convolution of two convex piecewise affine functions.

Lemma 6 ([11])**.** *Let β_1 and β_2 be two convex and piecewise affine functions over $\mathbb{R}_+$. Then, the convolution $\beta_1 * \beta_2$ consists in concatenating the segments of the service curves in the increasing order of the slopes, starting from $\beta_1(0) + \beta_2(0)$ (if β_1 and β_2 both end by a semi-infinite segment, $\beta_1 * \beta_2$ keeps only the semi-infinite segment with minimum slope).*

It can be exploited to transform the input network so that the service curve at each vertex is an elementary segment. Let v be a vertex with service curve β_v of size k: it is a piecewise affine convex function consisting in k segments with respective origin (x_i, y_i) and slope ρ_i, $i \in \{1, \ldots, k\}$, with $0 = x_1 < x_2 < \cdots < x_k < +\infty$. The last segment starting at x_k is either finite and ends at $x_{k+1} < +\infty$, or semi-infinite and ends at $x_{k+1} = +\infty$. Let us transform the vertex v into k vertices $v_1, \ldots, v_k$. The service function of vertex v_i is defined by $\beta_{v_i}(t) = \rho_i t$ if $0 \leq t \leq x_{i+1} - x_i$ and $+\infty$ otherwise. The input arcs of v_1 are the input arcs of v, the output arcs of v_k are the ouput arcs of v, and there is a single arc from v_i to v_{i+1} for all $i \in \{1, \ldots, k-1\}$. Then on the path from v_1 to v_k, the overall service curve is the convolution of the service curves on that path, that is β_v. Figure 5 shows

an example of the transformation. The service curve of the central vertex is composed of three segments. That vertex is replaced by three vertices, each of which has an elementary service curve corresponding to the three segments (the last segment is of infinite length, so its associate service curve is an affine function).

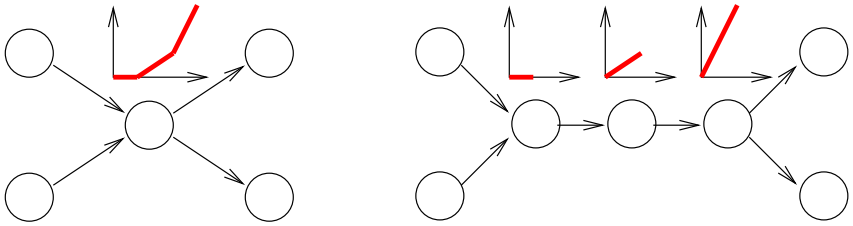

Figure 5: Transformation of the graph.

Transforming all the vertices in this way gives a graph such that there is a one-to-one correspondance between the paths of the initial graph from v to w and the paths of transformed graph from v_1 to $w_{|\beta_w|}$, for all $v, w \in V$, and such that for each path in this one-to-one correspondance, the overall service curve is the same. The transformation is linear in the size of the initial data: the number of vertices in the new graph is $\sum_{v\in V} |\beta_v|$ and the number of arcs is $|A| + \sum_{v\in V}(|\beta_v| - 1)$.

All these considerations can be summed up in the following proposition.

Proposition 1. *Any instance of the initial routing problem where the graph is $G = (V, A)$ and the services curves are $(\beta_v)_{v\in V}$ can be transformed into an* equivalent *instance where the graph $G' = (V', A')$ satifisfies $|V'| = \sum_{v\in V} |\beta_v|$, $|A'| = |A| + \sum_{v\in V}(|\beta_v| - 1)$, and the service curve at each vertex is an elementary segment.*

In the next subsection, we describe algorithms solving the optimal routing problem for inputs where the services curves are elementary segments. Thanks to the transformation, the algorithms apply to inputs where services curves have several segments. We could have directly presented our algorithms for this latter case. However the description and proof of the algorithms is more convenient when working with elementary segments.

3.3 Routing algorithms for concave arrival curves and convex service curves

Let $G = (V, A)$ be the directed graph where one wish to route a flow from v_{in} to v_{out}. The flow has an arrival curve α non-decreasing, concave and piecewise affine with a finite number of concatenated segments α_i of respective slope $r_i \in \mathbb{R}_+ \cup \{+\infty\}$ and length $l_i \in \mathbb{R}_+ \cup \{+\infty\}$, $0 \le i \le p$. Note that α_0 corresponds to the segment from $(0, 0)$ to $(0, \alpha(0+))$, thus $r_0 = +\infty$. Each vertex $v \in V$ has a service curve β_v which is an elementary segment of slope $\rho_v \in \mathbb{R}_+$ and length $\ell_v \in \mathbb{R}_+ \cup \{+\infty\}$.

Let $\mathfrak{p}$ be a path from v_{in} to v_{out}, we denote $\beta(\mathfrak{p})$ the service curve of the whole path. From Lemma 3, finding a path which minimizes the maximum delay with regard to α is equivalent to finding a path achieving the minimum

$$D_{\max}(G) = \min_{\mathfrak{p}_{v_{in}} \text{path to } v_{out}} \min_{\lambda\in\mathbb{R}_+\cup\{+\infty\}} H_\lambda(\alpha, \beta(\mathfrak{p})).$$

One only needs to replace H_λ by V_λ to minimize the maximum backlog. The service curve $\beta(\mathfrak{p})$ of path $\mathfrak{p}$ is the convolution of the elementary segments associated to its vertices. Due to Lemma 6, it is their concatenation by non-decreasing slopes. Thus applying Lemma 4, we know that for any path $\mathfrak{p}$, the minimum over λ is always reached for a slope $\lambda \in \{r_0, r_1, \dots, r_p\} \cup \{\rho_v \mid v \in \mathfrak{p}\}$ which is included in the set $\{r_0, r_1, \dots, r_p\} \cup \{\rho_v \mid v \in V\}$.

Interverting the two minima yields

$$D_{\max}(G) = \min_{\lambda\in\{r_0,r_1,\dots,r_p\}\cup\{\rho_v|v\in V\}} \min_{\mathfrak{p}_{v_{in}} \text{path to } v_{out}} H_\lambda(\alpha, \beta(\mathfrak{p})).$$

For each $\lambda \in \{r_0, r_1, \dots, r_p\} \cup \{\rho_v \mid v \in V\}$, from Lemma 5, $H_\lambda(\alpha, \beta(\mathfrak{p})) = \frac{\sqrt{1+\lambda^2}}{\lambda}\big(\sum_{i=0}^{p}(-proj_\lambda(\alpha_i))_+ + \sum_{v\in\mathfrak{p}}(proj_\lambda(\beta_v))_+\big)$. The first sum does not depend on the path $\mathfrak{p}$. Only the second term is subject to optimization. Minimizing this value over paths from v_{in} to v_{out} is equivalent to computing a *classical shortest path* from v_{in} to v_{out} in the graph $G = (V, A)$ where each vertex $v \in V$ is weighted by the orthogonal projection $proj_\lambda(\beta_v)_+ = \max(\frac{\lambda-\rho_v}{\sqrt{1+\lambda^2}\sqrt{1+\rho_v^2}}\ell_v, 0)$. Since the weights are non-negative, one can use for instance Dijkstra's algorithm. These arguments lead to Algorithm 1 and ensure its correction.

Algorithm 1 Optimal routing minimizing the maximum delay (resp. backlog).

Require: $G = (V, A)$ a directed graph and $v_{in}, v_{out} \in V$, $(\beta_v)_{v \in V}$ elementary segments of slopes ρ_v, α concave and concatenation of the segments $(\alpha_i)_{0 \leq i \leq p}$ of slopes r_i.

Ensure: A path $\mathfrak{p}$ from v_{in} to v_{out} minimizing $D_{\max}(\alpha, \beta(\mathfrak{p}))$ (resp. $B_{\max}(\alpha, \beta(\mathfrak{p}))$).

for $\lambda \in \{r_i \mid 0 \leq i \leq p\} \cup \{\rho_v \mid v \in V\}$ **do**

Replace each segment β_v by its orthogonal projection along the direction λ (if positive), i.e. by $\max(proj_\lambda(\beta_v), 0)$;

Compute a classical shortest path $\mathfrak{p}_\lambda$ from v_{in} to v_{out} with minimum weight b_λ in the graph $G = (V, A)$ with these weights;

Compute the sum $a_\lambda = \sum_{i=0}^{p} \max(-proj_\lambda(\alpha_i), 0)$;

Set the coefficient $c_\lambda \leftarrow \frac{\sqrt{1+\lambda^2}}{\lambda}$ to deal with delays (resp. $\sqrt{1+\lambda^2}$ for backlogs);

The minimum worst-case delay (resp. backlog) is $\min\{c_\lambda(a_\lambda + b_\lambda) \mid \lambda \in \{r_i \mid 0 \leq i \leq p\} \cup \{\rho_v \mid v \in V\}\}$ and an optimal path is $\mathfrak{p}_\lambda$ for λ achieving this minimum.

Computational complexity. We analyze the complexity for the initial problem where the functions β_v can have several segments. Let $G = (V, A)$ be the initial graph and denote $n = |V|$ and $m = |A|$. Up to the transformation of Section 3.2, we apply algorithm 1 to $G' = (V', A')$ where $|V'| = \sum_{v \in V} |\beta_v|$, $|A'| = m + \sum_{v \in V}(|\beta_v| - 1)$ and the service curves have been decomposed into elementary segments. The algorithm performs $|V'| + |\alpha|$ shortest path computations for the different projections of the segments. For each slope λ, computing the projections at all vertices is $\mathcal{O}(|V'|)$ and a shortest path can be computed using e.g. Dijskstra'algorithm in $\mathcal{O}(|A'| + |V'| \log |V'|)$ (with Fibonacci heaps). For each λ, one also has to compute the sum a_λ. It can be done in $\mathcal{O}(|\alpha|)$ by summing the projections over all the segments α_i. However one can speed up this step. The sum a_λ can be also described as the opposite of the orthogonal projection along direction λ of the point on curve α where the slope changes and becomes lower or equal to λ. One way to find efficiently this point is to precompute the coordinates of all the points at angles of α where the slope changes. Then store these points in a dictionary data structure using slopes as keys for the searches. With a binary search tree or simply a sorted array, if the input data for $|\alpha|$ is not already well formatted, the data structure is precomputed in $\mathcal{O}(|\alpha| \log |\alpha|)$ which turns out to be negligible. The search of the right point to project for a given λ is then performed in $\mathcal{O}(\log |\alpha|)$. The overall time complexity is thus $\mathcal{O}((|V'| + |\alpha|)(|A'| + |V'| \log |V'| + \log |\alpha|))$, that is $\mathcal{O}((\sum_{v \in V} |\beta_v| + |\alpha|)(m + (\sum_{v \in V} |\beta_v|) \log(\sum_{v \in V} |\beta_v|) + \log |\alpha|))$.

For some shapes of functions, there exist more simple and efficient algorithms to compute the best worst-case delay and backlog [5]. Table 3.3 sums up the complexities of the different algorithms. The parameters are:

- n = the number of vertices of the initial graph,
- m = the number of arcs of the initial graph,
- $|\alpha|$ = the total number of segments in the arrival curve α,
- $|V'| = \sum_{v \in V} |\beta_v|$ = the total number of segments of all the service curves.

The different complexities of our algorithms and their origins are:

- [1] = classical shortest path, e.g. $\mathcal{O}(m + n \log n)$ (see [5]),
- [2] = n shortest paths, e.g. $\mathcal{O}(mn + n^2 \log n)$ (see [5]),
- [3] = $(|V'| + |\alpha|)$ shortest paths on $|V'|$ vertices and m arcs including $\mathcal{O}(\log |\alpha|)$ additional computations. The overall complexity: $\mathcal{O}((|V'| + |\alpha|)(m + (|V'| \log |V'|) + \log |\alpha|))$.

	Affine arrival	Concave arrival
Rate-latency service	Backlog[1]/Delay[2]	Backlog[2]/Delay[2]
Convex service	Backlog[1]/Delay[3]	Backlog[3]/Delay[3]

Figure 6: Complexity to route optimally with regard to backlog or delay.

4 Optimal routing with general cross-traffic

When the cross-traffic is not independent, the previous approach collapses. The first issue comes from the computation of the service curve over a single path which is addressed in the next subsection while the problem of optimization is addressed in Section 4.4.

The algorithms described in the previous section only deal with independent flows. In the general case, there are several flows interfering. In that case, the Pay Multiplexing Only Once (PMOO) phenomenon has to be taken into account to have tighter bounds. In all the section, we make a strong assumption: we focus on a single flow that crosses several interfering traffic flows over sets of *consecutive vertices*.

As in [12], we assume here that we have blind multiplexing of the flows (there are no priority / FIFO policy) and make a worst case analysis. We generalize the bounds in [12] and give an efficient algorithm to compute the minimal service curve for one flow interfering with several other flows under blind multiplexing.

4.1 PMOO for one interfering flow

Let A_1 and A_2 be two arrival processes with respective arrival curves α_1 and α_2, that cross two concatenated servers with strict service curves β_1 and β_2. Let us compute the overall service curve for A_1 under blind multiplexing. Lemma 2 and Theorem 1 ensure that the service curve for A_1 is $(\beta_1 - \alpha_2)_+$ at server 1 and $(\beta_2 - \alpha_2 \oslash (\beta_1 - \alpha_1)_+)_+$ at server 2. Lemma 1 then states that the service curve for the two concatenated servers is $\beta = (\beta_1 - \alpha_2)_+ * (\beta_2 - \alpha_2 \oslash (\beta_1 - \alpha_1)_+)_+$. On the other hand, if one sees the two servers as one server (their concatenation) and then compute the service for A_1 under blind multiplexing, then the service curve for A_1 is $\beta' = ((\beta_1 * \beta_2) - \alpha_2)_+$.

Unfortunately, one cannot compare the two curves β and β'. If $\alpha_i(t) = \sigma_i + \rho_i t$, and $\beta_i = R_i(t - T_i)_+$ $i \in \{1, 2\}$, then we have $\forall t \in \mathbb{R}$,

$$\begin{aligned}
\beta(t) &= (\min(R_1, R_2) - \rho_2) \left(t - \frac{R_2 T_2 + \sigma_2 + \frac{R_1 T_1 + \sigma_1}{R_1 - \rho_1}}{R_2 - \rho_2} - \frac{R_1 T_1 + \sigma_2}{R_1 - \rho_2} \right)_+ \\
\beta'(t) &= (\min(R_1, R_2) - \rho_2) \left(t - \frac{\min(R_1, R_2)(T_1 + T_2) + \sigma_2}{\min(R_1, R_2) - \rho_2} \right)_+ .
\end{aligned}$$

If R_2 is large and σ_2 small, then β can be larger than β'. But, a major drawback of β is that it depends on α_1, whereas β' is an universal service curve (that is, it does not depend on the arrival curve). And one can always find σ_1 and ρ_1 such that $\beta' \geq \beta$. Morever, β' illustrates the PMOO phenomenon: considering the multiplexing only once with the concatenation of the servers possibly gives better results. Things become more complex when there are several interfering flows. An example of overlapping flows is given in Figure 7, where PMOO cannot be analysed using only the simple convolution and multiplexing operations described in Lemmas 1 and 2.

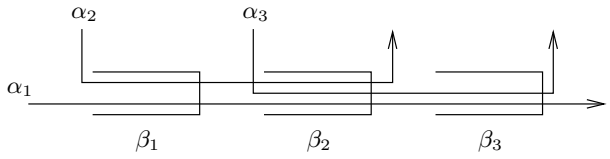

Figure 7: Overlapping flows.

4.2 PMOO with several interfering flows

Now, consider a flow F_1 with an arrival curve α_1, crossing servers $S_1, \ldots, S_n$ in that order. A strict service curve for S_j, $j \in \{1, \ldots, n\}$ is β_j. Let $(F_i)_{i \in \{2,\ldots,k\}}$ be the flows that interfere with F_1, with respective arrival curves α_i. Suppose that flow F_i interfere with F_1 only on a connected subpath (consecutive servers in the same order). Let us denote by S_{s_i} the server where the interference between F_1 and F_i starts and by S_{e_i} the server where it ends (in particular, we have $S_{s_1} = S_1$ and $S_{e_1} = S_n$). We denote by $A_i^{(j)}(t)$ the number of packets of flow F_i served by server S_j at time t and by $A_i^{(s_i - 1)}$ the number of packets for flow F_i arrived at time t.

Lemma 7. *With the notations and assumptions above,* $\forall t \in \mathbb{R}_+$, $\exists u_1, \dots, u_n \in \mathbb{R}_+$ *such that*

$$A_1^{(n)}(t) - A_1^{(0)}(t - \sum_{j=1}^{n} u_j) \geq 0, \text{ and}$$

$$A_1^{(n)}(t) - A_1^{(0)}(t - \sum_{j=1}^{n} u_j) \geq \sum_{j=1}^{n} \beta_j(u_j) - \Big(\sum_{i=2}^{k} A_i^{(e_i)}(t - \sum_{j=e_i+1}^{n} u_j) - A_i^{(s_i-1)}(t - \sum_{j=s_i}^{n} u_j) \Big).$$

Proof. The proof is done by induction on the number of servers.

For $n = 0$, nothing needs to be done : $A_1^{(0)}(t) - A_1^{(0)}(t) \geq 0 = e(0)$, where e is the unit element of $\mathcal{F}$ ($e(0) = 0$ and $e(t) = +\infty$ otherwise). Now, suppose that the lemma holds for $n-1$ servers. In particular, it holds for the $n-1$ first servers of a system of n servers, with the restriction of the interfering flows to $S_1, \dots, S_{n-1}$.

Consider the n-th server and denote by B the set of flows beginning their interaction with F_1 at server S_n and C the flows that have an interaction continuing to server S_n.

For every $t \in \mathbb{R}_+$, there exists u_n such that

$$A_1^{(n)}(t) + \sum_{i \in B \cup C} A_i^{(n)}(t) \geq \beta_n(u_n) + A_1^{(n-1)}(t - u_n) + \sum_{i \in B \cup C} A_i^{(n-1)}(t - u_n),$$

and $t - u_n$ is the start of the last backlog period at server n. This gives

$$A_1^{(n)}(t) - A_1^{(n-1)}(t - u_n) \geq 0 \text{ and}$$

$$\geq \beta_n(u_n) - \sum_{i \in B \cup C} \Big(A_i^{(n)}(t) - A_i^{(n-1)}(t - u_n) \Big), \quad (1)$$

Note that for every flow i in B, $s_i = n$ and for every flow in $B \cup C$, $e_i = n$.

Now, we are ready to combine Eq. (1) and the induction hypothesis applied to $t - u_n$: there exists of $u_1, \dots, u_{n-1} \in \mathbb{R}_+$ such that

$$\begin{aligned} A_1^{(n)}(t) - A_1^{(0)}(t - \sum_{j=1}^{n} u_j) \geq \sum_{j=1}^{n} \beta_j(u_j) \\ - \Big(\sum_{i \notin B \cup C} A_i^{(e_i)}(t - \sum_{j=e_i+1}^{n} u_j) - A_i^{(s_i-1)}(t - \sum_{j=s_i}^{n} u_j) \Big) \\ - \Big(\sum_{i \in C} A_i^{(n-1)}(t - u_n) - A_i^{(s_i-1)}(t - \sum_{j=s_i}^{n} u_j) \Big) \\ - \Big(\sum_{i \in C} A_i^{(n)}(t) - A_i^{(n-1)}(t - u_n) \Big) \\ - \Big(\sum_{i \in B} A_i^{(n)}(t) - A_i^{(n-1)}(t - u_n) \Big). \end{aligned}$$

The above remarks and straightforward simplifications for flows in C lead to the result for n servers, and in the same way this difference is proved to be non-negative. □

Theorem 2. *With the same assumptions and notations as above, then a service curve for* F_1 *of the servers* $S_1, \dots, S_n$ *under blind multiplexing is*

$$\phi(t) = \Big(\inf_{\substack{u_1, \dots, u_n \geq 0 \\ u_1 + \dots + u_n = t}} \sum_{j=1}^{n} \beta_j(u_j) - \sum_{i=1}^{k} \alpha_i(\sum_{j=s_i}^{e_i} u_i) \Big)_+.$$

Proof. Take the formula of the previous lemma. By causality of the system, we have $\forall i \in \{1 \dots, k\}$, $\forall j \in \{s_i, \dots, e_i\}$, $\forall t \in \mathbb{R}_+$ $A_i^{(j)}(t) \leq A_i^{(s_i-1)}(t)$. Then,

$$A_i^{(e_i)}(t - \sum_{j=e_i+1}^{n} u_j) - A_i^{(s_i-1)}(t - \sum_{j=s_i}^{n} u_j) \leq$$
$$A_i^{(s_i-1)}(t - \sum_{j=e_i+1}^{n} u_j) - A_i^{(s_i-1)}(t - \sum_{j=s_i}^{n} u_j) \leq \alpha_i(\sum_{j=s_i}^{e_i} u_j)$$

and

$$A_1^{(n)}(t) - A_1^{(0)}(t - \sum_{j=1}^{n} u_j) \geq \sum_{j=1}^{n} \beta_j(u_j) - \sum_{i=1}^{k} \alpha_i(\sum_{j=s_i}^{e_i} u_j).$$

Moreover $A_1^{(n)}(t) - A_1^{(0)}(t - \sum_{j=1}^{n} u_j) \geq 0$. □

This result introduces a new multi-dimensional operator for network calculus. It can be seen as a general formulation for the service curve on a path in presence of cross-traffic, while all cross-traffic flows intersect the path on connected subpaths. It naturally generalizes Lemma 2. This formula is also coherent with the formula presented in [12] with 2 cross-traffic flows and 3 nodes. In the following we will show how to make it effective.

Example 2. *To illustrate the formula, consider the system of Figure 7. The service curve given by Theorem 2 is ϕ with*

$$\phi(t) = \Big(\min_{\substack{u_1, u_2, u_3 \geq 0 \\ u_1 + u_2 + u_3 = t}} \beta_1(u_1) + \beta_2(u_2) + \beta_3(u_3) - \alpha_2(u_1 + u_2) - \alpha_3(u_2 + u_3) \Big)_+.$$

The formula for ϕ is not easy to simplify, using only the network calculus operators. Here is one possible simplification, bounding ϕ by a convolution.

Consider server S_j and let $B_j = \{i \in \{2, \dots, k\} \mid s_i = j\}$ be the set of flows that begin their interaction with F_1 at server j and $C_j = \{i \in \{2, \dots, k\} \mid s_i < j \leq e_i\}$ be the set of flows that continue their interaction with F_1 at server j.

For every $i \in \{2, \dots, k\}$, since $\alpha_i \oslash \alpha_i$ is a subadditive arrival curve for α_i [11],

$$\alpha_i\big(\sum_{j=s_i}^{e_i} u_j\big) = \alpha_i(u_{s_i}) + \Big(\alpha_i\big(\sum_{j=s_i}^{e_i} u_j\big) - \alpha_i(u_{s_i})\Big)$$
$$\leq \alpha_i(u_{s_i}) + \alpha_i \oslash \alpha_i\big(\sum_{j=s_i+1}^{e_i} u_j\big) \leq \alpha_i(u_{s_i}) + \sum_{j=s_i+1}^{e_i} \alpha_i \oslash \alpha_i(u_j).$$

As a consequence,

$$\phi(t) \leq \Big(\inf_{\substack{u_1, \dots, u_n \geq 0 \\ u_1 + \dots + u_n = t}} \sum_{j=1}^{n} \beta'_j(u_j) \Big)_+ = (\beta'_1 * \dots * \beta'_n)_+(t)$$

with $\beta'_j = \beta_j - \sum_{i \in B_j} \alpha_i - \sum_{i \in C_j} \alpha_i \oslash \alpha_i$.

Unfortunately, this formula is not very tight as soon as the functions α_i are composed of many affine pieces with different values at time 0. Another method de compute ϕ has been suggested in [12] to deal with the system of Figure 7 when arrival curves are concave and service curves are convex: the idea is to decompose each α_i as a minimum of affine functions, then use Theorem 2 to compute a service curve of the path for each of these affine arrival curves and finally recompose ϕ by taking the maximum of all those service curve. But the main drawback of this method is that it leads to very long computations, as one has to compute the maximum of many piecewise affine functions. If one decomposes the arrival curves and the service curves as a minimum and maximum of affine functions, one has to compute at the end the maximum of $N = |\alpha_1| \cdots |\alpha_k| . |\beta_1| \cdots |\beta_n|$ affine functions. The complexity of this is at least in $O(N \log N)$, which becomes huge very fast as one increases the number of servers or of interfering flows.

Theorem 2 applies for general arrival curves α_i and strict service curves β_i. In case all α_i are concave and all β_j are convex, there is another way to compute the service curve ϕ by taking advantage of the convexity and the concavity of the curves. It directly uses an algorithmic approach which is is detailed in the next section, and it outperforms the algorithm in [12].

4.3 Computation of the service curve of a path

Set $J = \{1, \dots, n\}$ and $I = \{1, \dots, k\}$. Here we state a more general problem: let $\{f_i\}_{i \in I}$ be a finite set of convex, continuous and piecewise affine functions on $\mathbb{R}_+$ and for each $i \in I$, define $J_i \subseteq J$. One wants to compute ϕ defined on $\mathbb{R}_+$ as

$$\phi(t) = \min_{\substack{u_1, \dots, u_n \geq 0 \\ u_1 + \dots + u_n = t}} \sum_{i \in I} f_i\Big(\sum_{j \in J_i} u_j\Big).$$

Lemma 8. *The function ϕ is convex, continuous and piecewise affine.*

Proof. For all $i \in I$, the function $f_i(\sum_{j \in J_i} u_j)$ is convex from $\mathbb{R}^n$ to $\mathbb{R}$ because it is the composition of f_i, convex from $\mathbb{R}$ to $\mathbb{R}$ and $\sum_{j \in J_i} u_j$ convex from $\mathbb{R}^n$ to $\mathbb{R}$.

Therefore, $g(u_1, \dots, u_n) \stackrel{\text{def}}{=} \sum_{i \in I} f_i(\sum_{j \in J_i} u_j)$is also convex from $\mathbb{R}^n$ to $\mathbb{R}$. Next, the domain $\mathcal{D}(t) \stackrel{\text{def}}{=} \{u_1, \dots, u_n \geq 0, \sum_{j \in J} u_j = t\}$ is convex in $\mathbb{R}^n$, therefore, g is convex from $\mathcal{D}(t)$ to $\mathbb{R}$.

Finally, consider the function ϕ from $\mathbb{R}$ to $\mathbb{R}$ and two distinct real numbers t_1 and t_2. Since the domain $\mathcal{D}(t_1)$ (resp. $\mathcal{D}(t_1)$) is compact, g reaches its minimum over $\mathcal{D}(t_1)$ (resp. $\mathcal{D}(t_1)$) at some point $v_1, \dots, v_n$: $g(v_1, \dots, v_n) = \phi(t_1)$ (resp. $w_1, \dots, w_n$, $g(w_1, \dots, w_n) = \phi(t_2)$). By convexity of g,

$$\alpha g(v_1, \dots, v_n) + (1-\alpha) g(w_1, \dots, w_n) \geq g(\alpha(v_1, \dots, v_n) + (1-\alpha)(w_1, \dots, w_n)).$$

Since the point $\alpha(v_1, \dots, v_n) + (1-\alpha)(w_1, \dots, w_n)$ belongs to $\mathcal{D}(\alpha t_1 + (1-\alpha)t_2)$, then

$$g(\alpha(v_1, \dots, v_n) + (1-\alpha)(w_1, \dots, w_n)) \geq \min_{u_1 + \dots + u_n = \alpha t_1 + (1-\alpha)t_2)} g(u_1, \dots, u_n).$$

Combining these two inequalities yields $\phi(\alpha t_1 + (1-\alpha)t_2) \leq \alpha\phi(t_1) + (1-\alpha)\phi(t_2)$.

Secondly, the functions $f_i(\sum_{j \in J_i} u_j)$ are piecewise affine so that $\sum_{i \in I} f_i(\sum_{j \in J_i} u_j)$ is piecewise affine as well as

$$\min_{\substack{u_1, \dots, u_n \geq 0 \\ u_1 + \dots + u_n = t}} \sum_{i \in I} f_i\Big(\sum_{j \in J_i} u_j\Big).$$

□

Now, let compute ϕ on an interval $[0, a]$, $a > 0$, and a small enough so that $\phi_{|[0,a]}$ is affine. Pose

$$F(u_1, \dots, u_n) = \sum_{i \in I} f_i\Big(\sum_{j \in J_i} u_j\Big).$$

For every $j \in J$, let $I_j = \{i \in I \mid j \in J_i\}$ be the set of functions where u_j appears in the expression of ϕ and let $\rho_j = \sum_{i \in I_j} r_{f_i}(0)$ be the slope of F when only u_j varies around 0. Let $\rho = \min_{j \in \{1, \dots, n\}}(\rho_j)$ be the minimal slope and suppose, without loss of generality that $\rho_1 = \rho$. Then, on an interval $[0, a]$, we have $\phi(t) = F(t, 0, \dots, 0)$. As every function f_i is convex, the slopes are increasing and that equality holds for a being the first point of change of slope of a function f_i, $i \in I_1$.

Let $t \geq a$. Suppose that $\phi(t) = F(u_1, \dots, u_n)$ with $u_1 < a$. We show that there exists another decomposition $\phi(t) = F(u_1', \dots, u_n')$ where $u_1' = a$.

Set $b = a - u_1$, $v_1 = a$ and consider a decomposition of $t - a$ in $t - a = \sum_{j \in J - \{1\}} v_j$ with $v_j \leq u_j \ \forall j \in J - \{1\}$. We have

$$\begin{aligned}
F(u_1, \dots, u_n) - F(v_1, \dots, v_n) &= \sum_{i \in I} f_i\Big(\sum_{j \in J_i} u_i\Big) - f_i\Big(\sum_{j \in J_i} v_j\Big) \\
&= \sum_{i \in I - I_1} \Big[f_i(\sum_{j \in J_i} u_j) - f_i(\sum_{j \in J_i} v_j)\Big] + \sum_{i \in I_1} \Big[f_i(\sum_{j \in J_i} u_j) - f_i(\sum_{j \in J_i} v_j)\Big] \\
&= \sum_{i \in I - I_1} \Big[f_i\Big(\sum_{j \in J_i} u_j\Big) - f_i\Big(\sum_{j \in J_i} v_i\Big)\Big] + \sum_{i \in I_1} \Big[f_i(u_1) - f_i(a)\Big] + \\
&\qquad \sum_{i \in I_1} \Big[(f_i\Big(\sum_{j \in J_i} u_i\Big) - f_i(u_1)) - (f_i\Big(\sum_{j \in J_i} v_j\Big) - f_i(v_1))\Big].
\end{aligned}$$

For $i \in I - I_1$, let us define h_i by $f_i(\sum_{j\in J_i} u_j) - f_i(\sum_{j\in J_i} v_j) = h_i \sum_{j\in J_i}(u_j - v_j)$, the average slope for f_i being h_i over $I - I_1$ and for $i \in I_1$, define h_i by $(f_i(\sum_{j\in J_i} u_j) - f_i(u_1)) - (f_i(\sum_{j\in J_i} v_j) - f_i(v_1)) = h_i \sum_{j\in J_i - \{1\}}(u_j - v_j)$.

The equation above can be rewritten as

$$\sum_{i\in I} h_i \sum_{j\in J_i - \{1\}} (u_j - v_j) - \rho b = \sum_{j\in J - \{1\}} \big(\sum_{i\in J_i} h_i\big)(u_j - v_j) - \rho b.$$

But, because of the convexity of the functions and because ρ is the minimum of the slopes around 0, we have $\sum_{i\in I_j} h_i \geq \rho$. Then $F(u_1, \ldots, u_n) - F(v_1, \ldots, v_n) \geq 0$ and a decomposition for $\phi(t)$ can be found where $u_1 \geq a$.

Set $f_i'(t) = f_i(t + a) - f_i(a)$ if $i \in I_1$ and $f_i' = f_1$ for $i \notin I_1$. For every $t \geq a$, there exists $u_1, \ldots, u_n \geq 0$ with $u_1 \geq a$ such that

$$\phi(t) = \sum_{i=1}^{k} f_i(\sum_{j\in J_i} u_j) = \sum_{i\in I_1} f_1(a) + \sum_{i=1}^{k} f_i'(\sum_{j\in J_i} u_j'),$$

with $u_1' = u_1 - a$ and $u_j' = u_j$ for $j \geq 2$. The functions f_i' are still convex, continuous and piecewise affine. So one can compute ϕ on an interval $[a, b]$ using the f_i'. Remark also that the total size of the functions f_i' is strictly less than that of the f_i, because a corresponds to a change of slope of a function (so the first segment of that function disappears in at least on of the $f_i', i \in I_1$). The function ϕ can then be computed in finite time, repeating the computations above at most as many times as the total number of segments of the functions f_i.

Algorithm 2 gives the computation of ϕ. The functions are represented as described in Paragraph 2.4. Operator **Next**.f points on the next triple of f and **AddSegment** construct ϕ adding the last three parameters as the last segment of ϕ. Moreover $\phi.x$, $\phi.y$ and $\phi.\rho$ represent the triple of the last constructed segment. In the outside loop, ρ can be found in time n, ℓ_0 in time at most k. The inside loop has a constant execution time if the total length remains the same, and has a complexity at most n if the total size of the f_i's decreased by one. The overall complexity is then in $O((\sum_{i=1}^{k} |f_i|)(k + n))$.

Algorithm 2 Computation of ϕ.

Require: I, J two finite sets, $f_i, i \in I$ convex continuous piecewise affine functions, $J_i \subseteq J$, $I_j \subseteq I$ such that $i \in I_j \Leftrightarrow j \in J_i$.

Ensure: $\phi : t \mapsto \min_{(u_j)_{j\in J} \geq 0, \sum_{j\in J} u_j = t} \sum_{i\in I} f_i(\sum_{j\in J_i} u_j)$.

```
φ ← nil; x ← Σ_{i∈I} f_i.x; y ← Σ_{i∈I} f_i.y;
for j ∈ J do
    ρ[j] ← Σ_{i∈I_j} f_i.ρ;
for i ∈ I do
    ℓ_i ← Next.f_i.x − f_i.x;
repeat
    Find j_0 such that ρ[j_0] = min{ρ[j], j ∈ J};
    ρ ← ρ[j_0]; AddSegment(φ, x, y, ρ);
    ℓ_0 ← min{ℓ_i | i ∈ I_{j_0}};
    x ← φ.x + ℓ_0; y ← φ.y + ρ_0.ℓ_0;
    for i ∈ I_{j_0} do
        ℓ_i ← ℓ_i − ℓ_0;
        if ℓ_i = 0 then
            ρ' ← f_i.ρ; f_i ← Next.f_i; ℓ_i ← Next.f_i.x − f_i.x;
            for j ∈ J_i do
                ρ[j] ← ρ[j] − ρ' + f_i.ρ;
until ℓ_0 = +∞
```

Applying Algorithm 2 to the functions β_j and $-\alpha_i$ outputs a function ϕ, and then removing the negative part by taking ϕ_+ gives the computation of the service curve of Theorem 2.

When all β_j are rate-latency functions and all α_i are affine, ϕ_+ is also a rate-latency function and its computation can be done in linear time due to simplifications as shown in the next section.

4.4 Optimal routing under PMOO

This section combines the optimal routing problem presented in Section 3 with the PMOO bound given in Theorem 2.

4.4.1 General acyclic graphs

First, let us consider the routing problem in a directed graph $G = (V, A)$ which is acyclic (containing no circuit). Each node v has a strict service curve β_v and the cross traffic is made of k flows F_i, $i \in \{1, \dots, k\} = I$ which respectively follow paths p_i and have respective arrival curves α_i.

Consider that the main flow (called F_0) follows a *fixed* path p in the graph from its source v_{in} to its destination v_{out}. Since the network is acyclic, the vertices are sorted according to the topological order in the graph. If vertex $v \in p_i$, $\phi_v^i(t)$ denotes the overall service curve of the cross-traffic F_i just after node v. Its arrival curve in the following node will then be $\alpha_i \oslash \phi_v^i$. Using Theorem 2, the service curves $\phi_v^i(t)$ are defined by the following inductive formula, where $v_1, \dots v_m$ are the vertices over path p_i up to node v (included) and $F_{h_1}, \dots, F_{h_r}$ are all the flows interfering with flow F_i up to vertex v. $v_h(\ell)$ is the vertex on p_h just before flows F_i and F_h meet for the ℓ^{th} time (they meet w times in total) and $p(\ell)$ is the ℓ^{th} commun sub-path for the flows F_i and F_h after node $v_h(\ell)$:

$$\phi_v^i(t) = \big(\min_{\sum_{j=1}^m u_j = t} \sum_{j=1}^{m} \beta_{v_j}(u_j) - \sum_{h=1}^{r} \sum_{\ell=1}^{w} \alpha_h \oslash \phi^h_{v_h(\ell)}\big(\sum_{v_j \in p(\ell)} u_j \big)\big)_+.$$

Using these notations, the end-to-end service curve of the main flow over path p is $\phi^0_{v_{out}}(t)$.

Note that all the functions ϕ_v^i depend on p, the path chosen for the main flow (F_0) from v_{in} to v_{out}.

Finally, Algorithm 3 provides the best route optimizing the service curve.

Algorithm 3 Computation of the best route in acyclic graphs

Require: an oriented acyclic graph $G = (V, A)$, The service curves β_v and the paths p_i, , $i \in \{1, \dots, k\}$, for each traffic flow.

Ensure: a route p and a service curve ϕ providing the best worst-case delay (resp. backlog) for the main flow;

for all path p in G from v_{in} to v_{out} **do**

Compute $\phi^0_{v_{out}}$;

keep the route p such that $D_{max}(\alpha_0, \phi^0_{v_{out}})$ is minimal (resp. $B_{max}(\alpha_0, \phi^0_{v_{out}})$) is minimal.

4.4.2 Strongly acyclic graphs

The main drawback of Algorithm 3 is that one needs to compute the service curve ϕ for each path p from v_{in} to v_{out} (there can be exponentially many). This is because the arrival curve of the cross-traffic in each node depends on the path p chosen for the main flow.

Here, we characterize networks for which this is not the case and where the arrival curve of the cross-traffic at each node can be precomputed by the classic use of the deconvolution formula as explained e.g. in [11]. Moreover the computation of the best route for the main flow can be carried without computing the service curve on each path.

An acyclic network with cross-traffic is *strongly acyclic* if for any pair of vertices in a connected component of the subgraph obtained by keeping only the arcs used by the cross-traffic, they are connected by at most one path in the initial graph which necessarily belongs to the subgraph.

Assuming that all α_i are affine with rate-latency service curves on each node, and using the formula of ϕ in Theorem 2, the service curve on a path $p = v_1, \dots, v_m$ can be written as

$$\phi(t) = \Big(- \sum_{i \in I} \sigma_i + \min_{u_1 + \dots + u_m = t} \sum_{j=1}^{m} (\beta_{v_j}(u_j) - (\sum_{i \in I_{v_j}} \rho_i) u_j) \Big)_+$$

where I is the set of the flows crossing path p.

Then, the key idea to model this is to weight the graph with service curves on the vertices and on the arcs:

- Vertex v is weighted with $\beta'_v : t \mapsto \beta_v(t) - (\sum_{i \in I_v} \rho_i)t$.
- Arc $e = (u, v)$ is weighted with β'_e with $\beta'_e(0) = \sum_{i \in I_v - I_u} \sigma_i$ and $\forall t \in \mathbb{R}_+ - \{0\}$, $\beta'_e = +\infty$.

On path $p = v_1, \dots, v_m$, the service curve is $\phi = (\beta'_{v_1} * \beta'_{(v_1,v_2)} * \beta_{v_2} * \dots * \beta'_{(v_{n-1},v_n)} * \beta'_{v_m})_+$. With the hypothesis on the arrival and service curves, such a service curve is the composition of a pure delay and a conservative link $\phi : t \mapsto R(t - T)_+$.

For a flow F on that path, with an affine arrival curve $\alpha(t) = \sigma + \rho t$, the maximal backlog is $\alpha(T) = \sigma + \rho T$ and the maximal delay is $\phi^{-1}(\sigma) = T + \sigma/R$. Due to the special shape of input functions, both R and T can be easily computed. The rate R is the smallest number among the rates of the β'_v: $R = \min_{v \in p}(R_v - (\sum_{i \in I_v} \rho_i))$. Then T can be deduced:

$$T = \sum_{v \in p} T_v \Big(1 + \sum_{i \in I_v} \frac{\rho_i}{R}\Big) + \sum_{e \in p} \frac{\beta'_e(0)}{R}.$$

The maximal delay and backlog strongly depend on R, and if R is fixed, following algorithm can be applied. In order to compute the maximal delay, for fixed R, the weight of a vertex v is $T_v(1 + \sum_{i \in I_v} \frac{\rho_i}{R})$ and the weight of an arc $e = (u, v)$ is $\sum_{i \in I_v - I_u} \sigma_i / R$. The maximal delay on a path is the sum of the weights on that path plus σ/R. Any shortest-path algorithm can be applied (restricted to the vertices whose asymptotic service rate is greater than R).

In order to compute the maximal backlog, for fixed R, the weight of a vertex v is ρT_v and the weight of an arc $e = (u, v)$ is $\rho \sum_{i \in I_v - I_u} \sigma_i / R$. The maximal backlog on a path is the sum of the weights on that path plus σ.

Following that scheme, in the worst case, one has to execute one shortest path algorithm per vertex (it covers all the possible rates R). All this results in the following proposition.

Proposition 2. *For a strongly acyclic network (V, A), with affine arrival curves and rate-latency service curves, the optimal end-to-end service curve is a rate-latency service curve that can be computed using a shortest path algorithm with an overall complexity in $O(|V|(|V| + |A|))$.*

4.5 Implementation work

Following the algorithmic framework of [6], a software for worst case performance evaluation with Network Calculus is currently under development. The main Network Calculus operations have been implemented for a large class of piecewise affine functions including the classical arrival and service curves of network calculus. A first version should be released soon for downloads (COINC Project [9]).

The new multi-dimensional operator, described in Algorithm 2 and corresponding to the service curve ϕ on a path with cross-traffic has been incorporated to the software. Routing algorithms presented in this paper have also been implemented.

These implementations are now tested on the example of Figure 8, which a strongly acyclic graph.

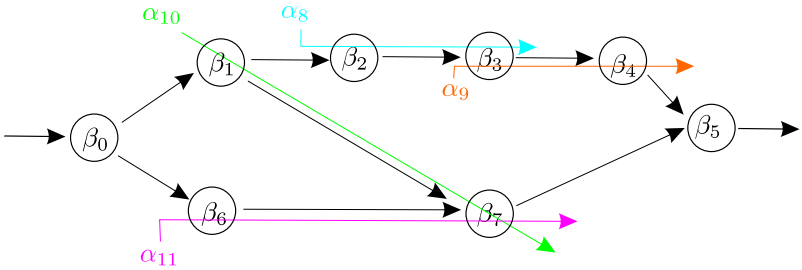

Figure 8: An example of a strongly acyclic network.

The routing problem has been solved for a main flow with arrival curve α that enters the network at vertex β_0 and leaves at vertex β_5. The arrival curves are affine: $\alpha(t) = \sigma + \rho t$ and for cross traffic $\alpha_i(t) = \sigma_i + \rho_i t$, $i \in \{8, 9, 10, 11\}$. The service curves are rate-latency functions in all nodes: $\beta_i(t) = R_i(t - T_i)_+$, $i \in \{0, 1, \ldots, 7\}$. The numerical values used for the example are given below.

	β_0	β_1	β_2	β_3	β_4	β_5	β_6	β_7		α	α_8	α_9	α_{10}	α_{11}
T	1	1	3	3	2	1	2	8	σ	25	8	1	10	8
R	21	22	20	18	22	24	26	22	ρ	3	2	4	2	4

In the example, the main flow may take three different paths: Path 1 is $\beta_0, \beta_1, \beta_2, \beta_3, \beta_4, \beta_5$, Path 2 is $\beta_0, \beta_6, \beta_7, \beta_5$ and Path 3 is $\beta_0, \beta_1, \beta_7, \beta_5$. It can be easily checked that this network with its cross-traffic is strongly acyclic.

By applying the routing algorithm sketched in Section 4.4.2, the worst case backlog $B_{\max}$ is minimal over Path 1 and it guarantees a worst case backlog $B_{\max} = 70.51$. As for the delay, the best maximal delay is reached over Path 3: $D_{\max} = 17.1875$.

In order to check the results and to point out the benefits of taking into account PMOO compared to the classical approach [11] treating the cross-traffic as independent in each node, we have also computed for each path:
- the service curve ϕ^i_{PMOO} for the path i when taking into account PMOO, thanks to Algorithm 2,
- and the service curve $\phi^i_{classic}$ for the path i computed in the classical way.
These two service curves are (*e.g.* on Path 1):

$$\phi^1_{PMOO}(t) = \big(\min_{\sum u_i = t} \beta_0(u_0) + \beta_1(u_1) + \beta_2(u_2) + \beta_3(u_3) + \beta_4(u_4) + \beta_5(u_5) - \alpha_8(u_2+u_3) - \alpha_9(u_3+u_4) - \alpha_{10}(u_1) \big)_+$$

$$\phi^1_{classic}(t) = \beta_0 * (\beta_1 - \alpha_{10})_+ * (\beta_2 - \alpha_8)_+ * (\beta_3 - \alpha_8 \oslash \beta_2 - \alpha_9)_+ * (\beta_4 - \alpha_9 \oslash \beta_3)_+ * \beta_5$$

The six curves are pictured on Figure 9 and their parameters are listed below. One can measure the gain of ϕ_{PMOO} over $\phi_{classic}$ and note that with $\phi_{classic}$, Path 3 would have been wrongly chosen as the best for the backlog.

	ϕ^1_{PMOO}	$\phi^1_{classic}$	ϕ^2_{PMOO}	$\phi^2_{classic}$	ϕ^3_{PMOO}	$\phi^3_{classic}$
T	15.17	17.17	16.75	17.75	15.625	16.5
R	12	12	16	16	16	16

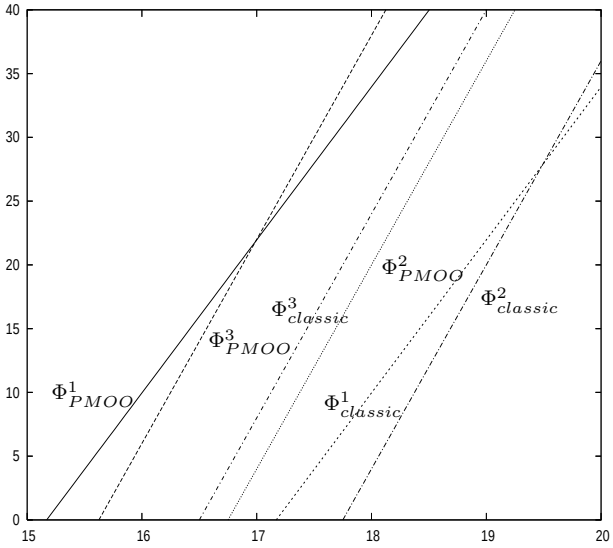

Figure 9: Service curves of paths.

5 Conclusion

Network Calculus does lend itself well to routing problems optimizing performance guarantees. We have shown how to solve them efficiently in usual cases for an independent cross-traffic. Then we have provided a new multi-dimensional operator quantifying the PMOO phenomenon for interfering flows, generalizing results of [12], and we have shown how to use it in routing problems for more general cross-traffic. Our results give insights into the benefit, but also the cost, of taking into account PMOO. The algorithms presented here are new bricks in a more global construction effort of an efficient software tool developped within the Coinc project [9], for the analysis and the control of complex networks, based on Network Calculus.

References

[1] E. Altman, T. Basar, and R. Srikant. Nash equilibria for combined flow control and routing in networks: Asymptotic behavior for a large number of users. *IEEE Transctions on Automatic Control*, 47(6):917–930, 2002.

[2] M. Andrews. Instability of FIFO in the permanent sessions model at arbitrarily small network loads. In *Proceedings of SODA'07*, 2007.

[3] B. Awerbuch and T. Leighton. A simple local-control approximation algorithm for multicommodity flow. In *Proceedings of FOCS'93*, pages 459–468, 1993.

[4] D. Bertsimas and D. Gamarnik. Asymptotically optimal algorithm for job shop scheduling and packet routing. *Journal of Algorithms*, 33(2):296–318, 1999.

[5] A. Bouillard, B. Gaujal, S. Lagrange, and E. Thierry. Optimal routing for end-to-end guarantees: the price of multiplexing. In *VALUETOOLS*, 2007.

[6] A. Bouillard and E. Thierry. An algorithmic toolbox for network calculus. *Journal of Discrete Event Dynamic Systems*, 2007. To appear.

[7] C. S. Chang. *Performance Guarantees in Communication Networks*. TNCS, 2000.

[8] J. Cohen, E. Jeannot, N. Padoy, and F. Wagner. Messages scheduling for parallel data redistribution between clusters. *IEEE Trans. Parallel Distrib. Syst.*, 17(10):1163–1175, 2006.

[9] COINC. Computational issues in network calculus. http://perso.bretagne.ens-cachan.fr/∼bouillar/coinc.

[10] M. Fidler. Extending the network calculus pay bursts only once principle to aggregate scheduling. In *QoS-IP*, pages 19–34, 2003.

[11] J.-Y. Le Boudec and P. Thiran. *Network Calculus: A Theory of Deterministic Queuing Systems for the Internet*, volume LNCS 2050. Springer-Verlag, 2001.

[12] J. B. Schmitt and F. A. Zdarsky. The disco network calculator: a toolbox for worst case analysis. In *Valuetools '06: Proceedings of the 1st international conference on Performance evaluation methodolgies and tools*. ACM Press, 2006.

[13] J. B. Schmitt, F. A. Zdarsky, and I. Martinovic. Performance Bounds in Feed-Forward Networks under Blind Multiplexing. Technical Report 349/06, University of Kaiserslautern, Germany, April 2006.

Unité de recherche INRIA Rennes
IRISA, Campus universitaire de Beaulieu - 35042 Rennes Cedex (France)

Unité de recherche INRIA Futurs : Parc Club Orsay Université - ZAC des Vignes
4, rue Jacques Monod - 91893 ORSAY Cedex (France)
Unité de recherche INRIA Lorraine : LORIA, Technopôle de Nancy-Brabois - Campus scientifique
615, rue du Jardin Botanique - BP 101 - 54602 Villers-lès-Nancy Cedex (France)
Unité de recherche INRIA Rhône-Alpes : 655, avenue de l'Europe - 38334 Montbonnot Saint-Ismier (France)
Unité de recherche INRIA Rocquencourt : Domaine de Voluceau - Rocquencourt - BP 105 - 78153 Le Chesnay Cedex (France)
Unité de recherche INRIA Sophia Antipolis : 2004, route des Lucioles - BP 93 - 06902 Sophia Antipolis Cedex (France)

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt, BP 105 - 78153 Le Chesnay Cedex (France)
http://www.inria.fr
ISSN 0249-6399