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

Anne Bouillard [a], Bruno Gaujal [b], Sébastien Lagrange [c], Éric Thierry [d]

[a] *ENS Cachan/IRISA, Campus de Beaulieu, 35000 Rennes, France*

[b] *INRIA/LIG, 51, av Kuntzmann, 38330 Montbonnot, France*

[c] *ISTIA/LISA, 62, av. Notre Dame du Lac, 49000 Angers, France*

[d] *CNRS/LIAFA & ENS Lyon/IXXI, 46, allée d'Italie, 69007 Lyon, France*

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 boil down to a functional shortest path problem. Under usual assumptions (concave arrival curves and convex service curves), a simple min-max lemma enables to solve the problem in polynomial time by a sequence of classical shortest path computations.

When cross-traffic perturbs the main flow over more than one node and under blind multiplexing, one can take into account the "Pay Multiplexing Only Once" (PMOO) phenomenon identified in [38,39]. It enables to improve bounds on delays and backlogs, but it makes the computation more involved. We provide a formula which gives a service curve for a path with PMOO conditions. It introduces a new multi-dimensional Network Calculus operator and it generalizes a formula of [38,39]. Moreover when arrival (resp. service) curves are affine (resp. convex), we describe an efficient algorithm to compute this formula. We finally show how to adapt our routing algorithms to have optimal end-to-end guarantees for these new bounds on delays and backlogs which take into account PMOO.

Key words: Network Calculus, Multiplexing, Shortest paths.

⋆ This work was supported by the INRIA Actions de Recherche Collaborative COINC, and partially supported by the ANR project MASED (06-JCJC-0069).

Email addresses: `Anne.Bouillard@bretagne.ens-cachan.fr` (Anne Bouillard), `Bruno.Gaujal@imag.fr` (Bruno Gaujal), `Sebastien.Lagrange@istia.univ-angers.fr` (Sébastien Lagrange),

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 (see [29] for a recent survey). Beyond classical shortest path algorithms, techniques range for instance from multiflow and linear programming in deterministic contexts [3,5] to optimal control or game theory in stochastic ones [1].

When the maximal delay over all packets in the routed flow is the performance index, most of the studies focus on *multi-constrained path problems* (see [20] for a nice and short survey). The optimization usually concerns m QoS measures and each arc or node in the network is characterized by a vector of m respective non-negative weights. The different QoS values of a path combine these weights over the nodes and arcs of the path, in an additive way (*e.g.* delay) or non-additive way (*e.g.* available bandwidth). As soon as several QoS values are involved in the optimization, the problems become computationally hard [42] and most of the work investigates the design of good heuristics [17,19,24–26,28,30–35]. In these models of networks, the assumptions on the flows and the network resources are rather static: the weights may express some measures involving time (like propagation delays or bandwidth) but they are fixed and do not evolve over time. When one comes to dynamic features, most studies focus on very simple systems such as single nodes where the issue becomes optimal scheduling.

In this paper, we consider the problem of computing the route for 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 and in the *Network Calculus* framework. Network Calculus [6,10,13,14] is a framework that allows us to formulate this problem as a mathematical program. This theory is based on (min,+) algebra (also called *tropical algebra*) and has been designed for deterministic network performance analysis, in particular for worst-case analysis. It enables to capture some dynamic features of a network thanks to a set of constraints on the shapes of the traffic and the services. These constraints are functions which can be combined together to provide some bounds on performances. Originally intended for Internet quality of service [15], its use seems promising in other fields: embedded systems like networks processors [41], switched Ethernet networks [16], sensor networks [37,40].

After a short introduction to Network Calculus definitions and notation (Sec-

Eric.Thierry@ens-lyon.fr (Éric Thierry).

tion 2), the first part of this paper (Section 3) is devoted to the computation of the best route for one flow from source to destination over an arbitrary network when the cross-traffic in each node is independent. Under *blind multiplexing* (no information about the scheduling policy at the nodes), we show how the problem boils down to solving a functional shortest path problem in a network with no cross-traffic. Under usual assumptions (concave arrival curves and convex service curves), a simple min-max lemma (Lemma 10) enables to solve the problem in polynomial time by a sequence of classical shortest path computations.using appropriate costs at each node (Theorem 15). This problem had been only partially solved in [7] for more restricted types of arrival and service curves such as affine arrival curves and rate-latency service curves.

The second part of the paper considers a more realistic case where cross-traffic in each node is not independent. This happens when some flows (cross-traffic flows and the flow to route) cross each other several times in the network. In this case, one can take into account the *Pay Multiplexing Only Once* (PMOO) phenomenon which was first identified in [38,39]. To give a short description of this phenomenon, let us say that when the routed flow merges with some cross-traffic flows, its service may be strongly reduced at the first node. However at the next nodes, the interference due to the cross-traffic cannot be as severe since the competition for the resource has already been partially resolved at the first node. The PMOO phenomenon can be quantified in the Network Calculus framework. For a fixed path, if each interfering cross-traffic flow follows a subpath, we give an explicit formula of an end-to-end service curve for the whole path (Theorem 20). It generalizes a formula provided in [38,39] for a specific scenario. The general formula is written under the form of a multi-dimensional convolution which helps designing an efficient algorithm to compute it when arrival curves are affine (Proposition 23 and Corollary 25). It can be compared to an algorithm of [38,39] with higher complexity.

The performance bounds on backlogs and delays are usually much better when using these PMOO end-to-end service curves (although not always as discussed in Section 4.3). For the routing problem, this single path analysis can be used in order to take into account PMOO and to find optimal routes with regard to tighter performance guarantees. In that case, we suggest a general approach for acyclic networks. However it has a high computational complexity. Under stronger assumptions (affine curves, 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.

Several algorithms of the paper have been implemented thanks to our Network Calculus library COINC [11] and have led to a few experiments (Section 5). Note that this paper corrects some errors that were left in its former version [7].

2 The Network Calculus framework

2.1 Network Calculus functions

Network Calculus primal objective is the performance analysis of communication networks. It can be presented as a deterministic queuing theory and since it is based on the $(\min, +)$ semi-ring, it is sometimes presented as a $(\min, +)$ analogue of the classical $(+, \times)$ filtering theory. Formally, the $(\min, +)$ semi-ring consists in the set $\mathbb{R}_{\min} = \mathbb{R} \cup \{+\infty\}$ equipped with operations min and $+$ which provide a semi-ring structure.

Flows and services in the network are modelled by non-decreasing functions $t \mapsto f(t)$ where t is *time* and $f(t)$ an amount of *data*. There are different models depending on whether t (resp. $f(t)$) takes discrete or continuous values, *e.g.* in $\mathbb{N}$ or $\mathbb{R}_+$. Concerning data, one can even consider three models of granularity:

- *infinitesimal* (sometimes called *fluid*): data can be divided into arbitrarily small pieces (data measures are in $\mathbb{R}_+$);
- *unitary* (sometimes called *discrete*): data is composed of indivisible packets of the same length (data measures are in $\mathbb{N}$),
- *multi-scaled*: data may be divided into packets of variable lengths (possibly in $\mathbb{N}$, $\mathbb{R}_+$ or even infinitesimal).

Note that we do not exactly use the term *fluid* as in [6] where the *fluid model* adds the condition that the manipulated functions are continuous.

In this paper, we consider continuous time and data which can be arbitrarily divided or data which is composed of indivisible packets of the same length. In both cases, we call *bit* the data unit (divisible in the first case, or indivisible and equal to the fixed packet length in the second case). To cover both cases, we work with functions from $\mathbb{R}_+$ into $\mathbb{R}_{\min}$ which are not necessarily continuous. All our results apply to both models for data: all the classical Network Calculus formulas, like the ones providing bounds on worst-case backlogs and delays, are valid in both cases and this is also true for our proofs (see [6] for discussions about the discrete vs. continuous models). In the paper, we do not tackle networks and traffic manipulating data with variable length packets. As explained in [6], classical formulas require then some adjustments, *e.g.* using a *packetization* operator, and our work does not study these aspects. One may also wonder whether our results apply to the discrete time model. In fact it depends and we will discuss it for the main theorems of this paper.

Consider the set $\mathcal{F}$ of functions from $\mathbb{R}_+$ into $\mathbb{R}_{\min}$. Beyond usual operations like the minimum or the addition of functions, Network Calculus makes use of several classical operations [4] which are the translations of $(+,\times)$ filtering operations into the $(\min,+)$ setting. The *convolution*, denoted $*$, and the *deconvolution*, denoted $\oslash$, are defined as: for all f,g in $\mathcal{F}$, $\forall t\in\mathbb{R}_+$,

- Convolution: $(f*g)(t)=\inf_{0\le s\le t}(f(s)+g(t-s))$.
- Deconvolution: $(f\oslash g)(t)=\sup_{u\ge 0}(f(t+u)-g(u))$.

The convolution is associative, commutative and distributive w.r.t. the minimum. The deconvolution also have interesting algebraic properties (see [6] for a survey on such properties). Using such operations, Network Calculus formulas combine the constraints on the traffic and the services in the network in order to output worst-case performance bounds.

2.2 Arrival and service curves

Real cumulative functions vs. constraint functions. In Network Calculus, one must distinguish two kinds of objects: the real movements of data and the constraints that these movements satisfy. The real movements of data are mainly modeled by *cumulative functions*: a cumulative function $f(t)$ counts the total amount of data that has achieved some condition up to time t (*e.g.* the total amount of data which has gone through a given place in the network). In all the paper, we make the usual assumption that *cumulative functions are left-continuous*. This is not a huge restriction for the modeler: *e.g.* let $f(t)$ be the total amount of data that has entered a system until time t, in case an instantaneous burst of b bits occur at time t_0 while a bits had already arrived, one only has to set $f(t_0)=a$ and $f(t_0+)=a+b$. This assumption has nevertheless a technical importance in the Network Calculus edifice (*e.g.* when using *strict service curves* in Theorem 20 or below). On the contrary, no assumption of (left- or right-)continuity is imposed to the constraint functions.

Arrival curves. Given a data flow traversing a system, let $A\in\mathcal{F}$ be its *cumulative arrival function*, *i.e.* $A(t)$ is the number of bits that have arrived in the system until time t, with $A(0)=0$. A function $\alpha\in\mathcal{F}$ is an *arrival curve* for A if $\forall\, s,t\in\mathbb{R}_+$, $0\le s\le t$, we have $A(t)-A(s)\le\alpha(t-s)$. It means that the number of bits arriving between time s and t is at most $\alpha(t-s)$. An example of arrival curve is the affine function $\alpha(t)=\sigma+\rho t$, $\sigma,\rho\in\mathbb{R}_+$ (sometimes called *leaky-bucket* arrival curve). In this case, σ can be interpreted as the maximal number of bits that can arrive simultaneously (maximal burst) and ρ the maximal long-term rate of arrivals.

Service curves. Now consider B the *cumulative departure function* of the flow, *i.e.* the number $B(t)$ of bits that have left the system until time t, with $B(0) = 0$. The system is said to provide a (minimum) *service curve* $\beta \in \mathcal{F}$ if $B \geq A * \beta$. Particular cases of service curves are called:

- *Peak rate* with rate $R \in \mathbb{R}_+$: $\lambda_R(t) = Rt$ for all $t \in \mathbb{R}_+$;
- *Burst delay* with delay $T \in \mathbb{R}_+$: $\delta_T(t) = 0$ if $t \leq T$ and $= +\infty$ otherwise;
- *Rate-latency* with delay $T \in \mathbb{R}_+$ and rate $R \in \mathbb{R}_+$: $\beta_{R,T} = R(t-T)_+$ for all $t \in \mathbb{R}_+$;

with the notation $a_+ = \max(a, 0)$ for $a \in \mathbb{R}_+$. Note that if β is a service curve, then $B(0) \geq A(0) + \beta(0)$. Since $\forall t \in \mathbb{R}_+$, $B(t) \leq A(t)$, we have $B(0) \leq A(0)$ and thus $\beta(0) \leq 0$. If β is non-negative, $\beta(0) = 0$.

A *backlogged period* of the system is an interval of time during which the backlog is non-null. Let $t \in \mathbb{R}_+$, the *start of the backlogged period* of t is $s = \sup\{u \leq t | A(u) = B(u)\}$. Since the cumulative functions A and B are assumed left-continuous, we also have $A(s) = B(s)$. If $A(t) = B(t)$, then $s = t$.

A *strict service curve* β for the system is a non-negative function in $\mathcal{F}$ such that during any backlogged period of duration u, at least $\beta(u)$ data is served. More formally, for any $s, t \in \mathbb{R}_+$, $s \leq t$, if $\forall s < v < t$, $A(v) - B(v) > 0$, then $B(t) \geq B(s) + \beta(t - s)$. A strict service curve is clearly always a service curve for the system (by considering starts of backlogged periods), but the converse is not true: *e.g.* there exist systems with δ_T or $\beta_{R,T}$ as a service curve but not as a strict service curve (see [6], pages 22 and 198). Note that these counterexamples also disprove the following unfortunate statement of [7]: "convex service curves are always strict service curves". Strict service curves are an enforcement of the service curve notion. There are both useful to model the behaviour of some servers and necessary to get refined bounds in some situations (*e.g.* positiveness of output service curves in Lemma 5 and Theorem 20).

Convex/concave/piecewise affine assumptions. In this paper, we will often assume that the input curves are piecewise affine with a finite number of pieces, and convex or concave. These are recurrent assumptions in Network Calculus [6,10,38,41]. Their relevance depends of course on the type of systems which are modeled. Here is a simple way to generate such functions with an arbitrary number of pieces in the Internet context. Integrated Services Networks (Intserv [18]) introduce traffic specifications (T-SPEC) with arrival curves of the form $\alpha(t) = \min(M + pt, rt + b)$. Adding T-SPEC arrival curves when aggregating flows leads to concave arrival curves with any number of pieces. Note also that applying Lemma 5 of the next subsection, with a T-SPEC arrival curve and a rate-latency strict service curve may yield an convex strict service curve with three affine pieces. The convolution of functions with three pieces can provide service curves with any number of pieces.

Note that since we consider functions from $\mathbb{R}_+$ into $\mathbb{R}_{\min}$, our definition of a *convex* (resp. *concave*) function f is that $\mathcal{U}(f) = \{(x, y) \in \mathbb{R}_+ \times \mathbb{R} | y \geq f(x)\}$ (resp. $\mathcal{D}(f) = \{(x, y) \in \mathbb{R}_+ \times \mathbb{R} | y \leq f(x)\}$) is a convex subset of $\mathbb{R}_+ \times \mathbb{R}$. In the same way, a function f from $\mathbb{R}_+$ into $\mathbb{R}_{\min}$ is *piecewise affine* if its support (the set where it is finite) is $\mathbb{R}_+$ or an interval $[0, T]$, $T \in \mathbb{R}_+$, and f is piecewise affine on its support. The number of affine pieces of f is denoted $|f|$.

Remark 1 *In this paper, we will sometimes use arrival and service curves which are continuous over $\mathbb{R}_+$ and take non-integer values. One may then question our statement in Subsection 2.1 assuring that our results also apply for the discrete data model where any data manipulated in the system is a integer number of bits. This is not an issue: although it is true that in this case cumulative functions, such as $A(t)$ and $B(t)$ above, will be defined from $\mathbb{R}_+$ into $\mathbb{N}$, nothing prevents us from using continuous functions from $\mathbb{R}_+$ into $\mathbb{R}_{\min}$ that fulfill the definitions of arrival and service curves.*

Note that during the modeling of a network, one must be cautious about such distinctions about the data granularity. For instance, consider a node which is said to serve at least r bits per second, $r \in \mathbb{N}$. If data can be divided into infinitesimal pieces, then $\lambda_r(t) = rt$ is a service curve. If data is always an integer number of bits, then λ_r is not any longer a service curve, but $\beta(t) = \lceil rt - r \rceil$ for $t > 0$ and $\beta(0) = 0$ is one, as well as the rate-latency $\beta_{1/r,r}$ which is a little less accurate.

2.3 Performance bounds

Bounds on worst-case backlogs and delays can be easily derived from the Network Calculus constraints.

Definition 2 *Let A be the cumulative arrival function of a flow entering a system and let B be its corresponding cumulative departure function. Then the* backlog *of the flow at time t is*

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

and the delay (assuming FIFO order when serving bits of the flow) at time t is

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

Given an arrival curve and a service curve, it is possible to compute with the Network Calculus operations an upper bound on the maximal backlog (resp.

delay). Moreover, one can also compute the arrival curve of the departure flow.

Theorem 3 ([6,10]) *Let $A \in \mathcal{F}$ be the cumulative arrival function with an arrival curve α for a flow entering a system with service curve β. Let $B \in \mathcal{F}$ be the cumulative departure function. Then,*

(1) B *has an arrival curve* $\alpha \oslash \beta$.
(2) $b(t) \leq B_{\max}(\alpha,\beta) \stackrel{\text{def}}{=} \sup\{\alpha(t) - \beta(t) \mid t \geq 0\} = (\alpha \oslash \beta)(0)$. [1]

(3) $d(t) \leq D_{\max}(\alpha,\beta) \stackrel{\text{def}}{=} \inf\{d \geq 0 \mid \forall t \geq 0,\ \alpha(t) \leq \beta(t+d)\}$ [2]

$$= \inf\{d \geq 0 \mid (-\beta) \oslash (-\alpha)(d) \leq 0\}.$$

As illustrated on Fig. 1, the backlog bound $B_{\max}(\alpha,\beta)$ is the maximal vertical distance between α and β while the delay bound $D_{\max}(\alpha,\beta)$ is given by the maximal horizontal distance between those two functions.

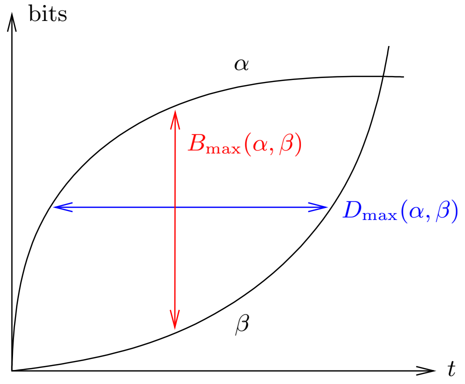

Fig. 1. Bounds on worst-case backlog and worst-case delay.

Given a network of server nodes with some cross-traffic, the main problem of this article is to route flows along paths with good end-to-end performance guarantees. It requires to quantify the performances over a path and the next two lemmas are very useful for computing service curves of concatenated servers and of servers with blind multiplexing of flows.

Lemma 4 (Tandem [6,10]) *Consider a flow crossing two nodes in tandem with respective service curves β_1 and β_2. Then the concatenation of the two nodes offers a minimum service curve $\beta_1 * \beta_2$ to the flow.*

Lemma 5 (Residual service [6,10]) *Consider a node 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 $\beta_1 = (\beta - \alpha_2)_+$.*

[1] with the convention $(+\infty) - (+\infty) = -\infty$
[2] with the convention $\inf \emptyset = +\infty$

This lemma makes no assumption on the scheduling policy which is applied to the input flows. Stating results which apply to any set of policies is equivalent to working with unknown policies. This situation is often called *blind multiplexing*. Note that a careful look at the proof of Lemma 5 in [6] shows that it is not required that β_1 is non-decreasing.

The next example illustrates some Network Calculus computations for usual input functions.

Example 6 ([6]) *Let $\alpha(t) = \sigma + \rho t$, $\beta(t) = R(t - T)_+$ where $\sigma, \rho, R, T \in \mathbb{R}_+$. Then*

$$
\begin{aligned}
(\alpha \oslash \beta)(t) &= \begin{cases} (\sigma + \rho T) + \rho t, & \forall t \in \mathbb{R}_+ \text{ if } \rho \leq R \\ +\infty, & \forall t \in \mathbb{R}_+ \text{ if } \rho > R. \end{cases} \\
(\alpha - \beta)_+(t) &= \begin{cases} (R - \rho)\left(t - \frac{\sigma + \rho T}{R - \rho}\right)_+, & \forall t \in \mathbb{R}_+ \text{ if } \rho < R \\ 0, & \forall t \in \mathbb{R}_+ \text{ if } \rho \geq R. \end{cases}
\end{aligned}
$$

Let $\beta_i(t) = R_i(t - T_i)_+$, $i \in \{1, 2\}$. Then

$$(\beta_1 * \beta_2)(t) = \min(R_1, R_2)(t - (T_1 + T_2))_+$$

Concerning the deconvolution, it is also easy to see that if $\alpha(t) = \sigma + \rho t$, with $\sigma, \rho \in \mathbb{R}_+$, then for all $\beta \in \mathcal{F}$, $\alpha \oslash \beta$ is also affine: for all $t \in \mathbb{R}_+$, $(\alpha \oslash \beta)(t) = \sigma + \rho t + \sup_{s \geq 0}(\rho s - \beta(s))$.

Remark 7 *The* strict service curve *notion must be used cautiously. Results for service curves do not automatically admit a strict service curve version.*

For example, unlike Lemma 4, if two nodes in tandem have strict *service curves β_1 and β_2, then $\beta_1 * \beta_2$ is not necessarily a* strict *service curve for their concatenation. Here is a counterexample with burst-delay strict service curves. As a direct application of definitions, a node admits δ_T as a strict service curve if and only if the duration of any backlogged period is at most T. Consider the system on Fig 2 composed of two nodes in tandem. Suppose that Node 1 (resp. Node 2) works as the following server: as soon as a positive amount of data arrives, it waits a time $T_1 = 2$ (resp. $T_2 = 3$) while its queue fills up, and then it serves instantaneously all the data in its queuing. Node 1 (resp. Node 2) clearly has δ_{T_1} (resp. δ_{T_2}) as a strict service curve. Fig 2 shows what happens to a flow crossing Node 1 and 2 with the cumulative arrival function $A(t)$. The duration of the backlogged period is 9, and thus $\delta_{T_1} * \delta_{T_2} = \delta_{T_1+T_2}$ is not a strict service curve since $T_1 + T_2 = 5$.*

The strict service curve property can be also lost in Lemma 5: under blind multiplexing, a node with strict *service β crossed by a flow of arrival curve α_2 does not necessarily offer $(\beta - \alpha_2)_+$ as a* strict *service curve to other flows.*

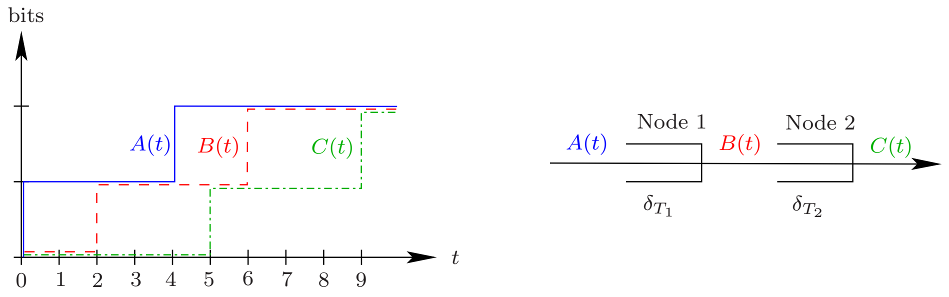

Fig. 2. Tandem: strict service curve * strict service curve $\neq$ strict service curve.

Suppose for instance that we are in the infinitesimal data model and consider a node which serves data at a constant rate 2. Such a node has $\beta(t) = 2t$ as a strict service curve. Let two flows cross this node with respective cumulative arrival functions $A_1(t) = t + 1$ and $A_2(t) = t$. An arrival curve of Flow 2 is $\alpha_2(t) = t$. Fig 3 shows the trajectory of the system for the following policy: at first, for $0 \leq t \leq 1$, Flow 1 is given top priority, then for $t > 1$, Flow 2 is given top priority. Since the sum $A_1(t)+A_2(t) = 2t+1$, the node is always backlogged and the sum of the cumulative departure functions is $B_1(t) + B_2(t) = 2t$. It can be checked that $\beta_1(t) = (\beta - \alpha_2)_+(t) = t$ is a service curve for Flow 1 ($B_1 \geq A_1 * \beta_1$) but it is not a strict *service curve: during the period $1 \leq t \leq 2$, Flow 1 has some data backlogged in the node but this data is not served at all.*

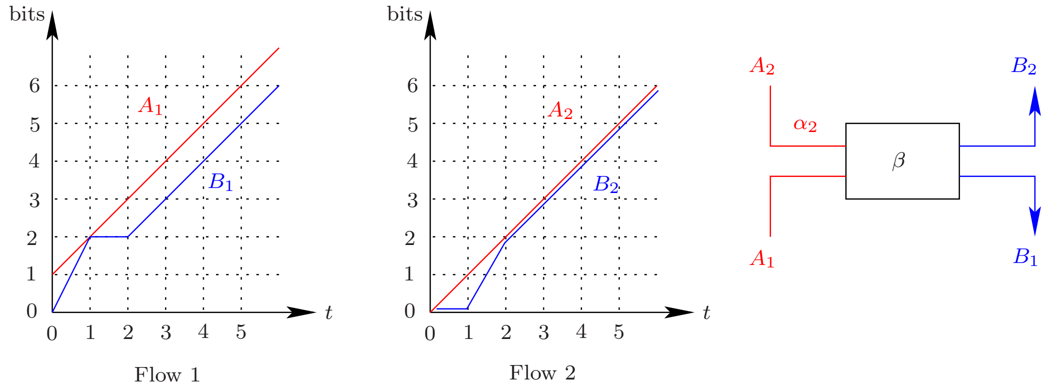

Fig. 3. Residual service curves are not necessarily strict.

Due to all those traps, we will warn all along the paper whenever service curves are required to be strict or not.

3 Optimal routing with independent cross-traffic

3.1 The routing problem

We first specify the general model of network and cross-traffic that we consider, including the scheduling policy:

- **Network**: the network is represented by a directed graph $G = (V, A)$ with $|V| = n$ nodes and $|A| = m$ arcs. Each node v is a server for which a non-negative service curve β_v is known. Those service curves can be assumed to be either strict or not.
- **Cross-traffic**: the cross-traffic is composed of k flows. Each flow F_i follows a path $\mathfrak{p}_i$ in the directed graph G. The first (resp. last) node of the path is where data enter (resp. leave) the network. At its entry in the network, each flow F_i is constrained by an arrival curve α_i bounding the injection of data.
- **Scheduling policy**: blind multiplexing, *i.e.* nothing is known about the scheduling policies at the nodes when flows aggregate (nonetheless if one only considers data belonging one of the flows, it is served in a FIFO way).

In such a model, arcs instantaneously transmit data between the nodes. If one wish to introduce some constraints on an arc, like a propagation delay or a minimum throughput, it can add a new node in the middle of the arc with a service curve (strict or not) corresponding to those features.

The general routing problem that we study can be stated as follows:

Problem 8 (Optimal routing) *The inputs are:*

- *A network* $(G = (V, A), (\beta_v)_{v \in V})$ *and its cross-traffic* $((\mathfrak{p}_i)_{1 \le i \le k}, (\alpha_i)_{1 \le i \le k})$ *modeled as above.*
- *A flow* F *with arrival curve* α*, to route from a node* v_{in} *to a node* v_{out}*.*
- *An end-to-end performance measure (like worst-case end-to-end delay or backlog for the flow* F*) and a cost function function which associates with* α *and any path* $\mathfrak{p}$ *from* v_{in} *to* v_{out} *an upper bound on this performance measure.*

Find a path $\mathfrak{p}$ *from* v_{in} *to* v_{out} *for the flow* F *which minimizes the bound provided by the cost function.*

The significance of the optimal path which is searched is of course strongly linked to the quality of the cost function providing the performance bounds. A loose cost function may generate easier computations but if the performance bounds are really far from the worst-case behaviour of the system, then the relevance of the computations may be questioned. Finding such good cost functions, *i.e.* tight bounds on end-to-end performance measures, is a major issue of Network Calculus and some important problems remain open. Consider for instance the model of network and cross traffic exposed above, if the graph G admits some circuits, checking the *stability* (*i.e.* that the amount of data backlogged in the network never grows to infinite) is open. It is not even known if it is decidable or not [2,6]. Fortunately in many interesting cases, *e.g.* when the graph is acyclic, one can lead some performance analyses with Network Calculus.

The way cross-traffic flows may interfere plays an important role in the difficulty to analyze the performances and consequently in the complexity of the routing problem. From now on, this section will study the case of *independent cross-traffic*, that is when each cross-traffic flow crosses only one node of the network. Section 4 will be devoted to more general situations where some cross-traffic flows and the routed flow may interfere at more that one node.

The **Optimal routing with independent cross-traffic** problem is a particular case of the general routing problem where we assume that the path $\mathfrak{p}_i$ associated with each cross-traffic flow F_i is reduced to a *single node* and where all the service curves $(\beta_v)_{v\in V}$ are *strict*. Note that the graph G is not required to be acyclic. Fig. 4 illustrates an instance of this problem as well as a possible path to route a flow from v_{in} to v_{out}.

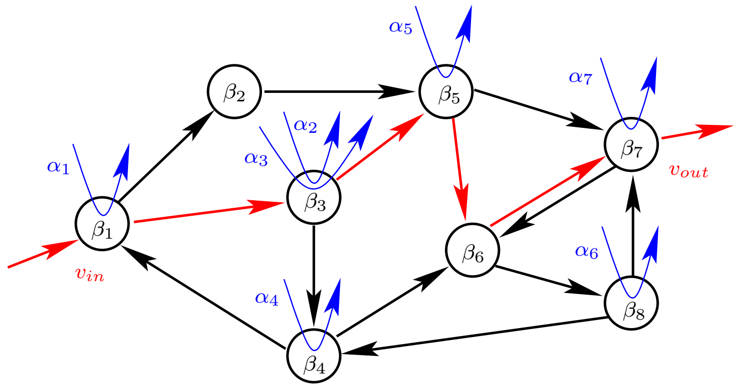

Fig. 4. An instance of network with independent cross-traffic (blue curved arrows) and a possible path from v_{in} to v_{out} (in red).

We focus on worst-case end-to-end delays and backlogs for the routed flow and we need a cost function which is a good upper bound for those measures. A way to lead the analysis is to get rid of the cross-traffic. The straightest way is the use of Lemma 5: for each node v, replace the strict service curve β_v by the residual service curve $\tilde{\beta}_v = (\beta_v - \sum_{i\in I_v} \alpha_i)_+$ (it uses the *strict* hypothesis on β_v). Let $\mathfrak{p} = (v_0, v_1, \ldots, v_\ell)$, $v_0 = v_{in}$ and $v_\ell = v_{out}$, be a possible path with no loop for the routed flow, then due to Lemma 4 and 5, this path offers an end-to-end service curve $\beta(\mathfrak{p}) = \tilde{\beta}_{v_0} * \tilde{\beta}_{v_1} * \cdots * \tilde{\beta}_{v_\ell}$ which can be used to compute bounds. If we retain this analysis, the optimal routing problem boils down to solving the optimal routing problem with no cross-traffic.

Problem 9 (Optimal routing with no cross-traffic) *The inputs are:*

- *A directed graph $G = (V, A)$ (not necessarily acyclic) with non-negative service curves $(\beta_v)_{v\in V}$ at the nodes.*
- *A flow F with arrival curve α, to route from a node v_{in} to a node v_{out}.*

*Find a path $\mathfrak{p} = (v_0, v_1, \ldots, v_\ell)$ from $v_0 = v_{in}$ to $v_\ell = v_{out}$ for the flow F which minimizes $D_{max}(\alpha, \beta(\mathfrak{p}))$ (*guaranteed delay optimization*) or $B_{max}(\alpha, \beta(\mathfrak{p}))$*

*(*guaranteed backlog optimization*) where* $\beta(\mathfrak{p}) = \beta_0 * \beta_1 * \cdots * \beta_\ell$.

A few words must be said about the presence of loops or not in the path used to route the flow. In our model of network and cross-traffic, it is clear that for the routed flow, adding a loop to a path increases the worst-case end-to-end delays and backlogs. Thus if we were optimizing with regard to the worst-case delay (or backlog) under the constraints of the model, paths with loops would be directly eliminated. However we are optimizing with regard to cost functions which are just bounds on the worst-case delay (or backlog), and, what is even more tricky, the cost functions we use do not necessarily give valid bounds if there is a loop on the path. Thus one may wonder whether the cost functions we use will result in eliminating paths with loops. First such a direct elimination would be satisfying with regard to what happens to the exact worst-case delay or backlog. Moreover it would save ourselves discussions about problematic cases: an algorithm might return a path with loops, optimal with regard to the cost function (even though the cost function has no meaning because there is a loop) and which could not be simplified by removing loops (to regain the meaning of the cost) without increasing strictly its cost. The good news is that for the *optimal routing without cross-traffic*, the cost function $D_{max}(\alpha, \beta_0 * \beta_1 * \cdots * \beta_\ell)$ (resp. B_{max}) for a path $\mathfrak{p} = (v_0, v_1, \ldots, v_\ell)$ increases when adding a loop, since the service curves satisfy $\beta_v(0) = 0$: the extended path will have a smaller service curve due to the property $\beta * \gamma \leq \beta$ if $\gamma(0) = 0$.

For the routed flow F, studying the worst-case end-to-end delay seems natural. The worst-case end-to-end backlog which only concerns data from the flow F, distributed among the nodes of the path, may seem a less relevant measure. However it is an indicator about the global workload of the network and thus one may wish to use it as an optimization criteria.

Since the optimal routing problem with independent cross-traffic can be studied thanks to a reduction to the routing problem with no cross-traffic, we investigate solutions to this latter problem in the next subsections. In order to perform computations and design algorithms, we need to make further assumptions on the shapes of the inputs functions α and $(\beta_v)_{v \in V}$. In the next section, we consider sets of piecewise affine functions but not as general as the one studied in [8]. We make some usual Network Calculus assumptions: convex service curves and concave arrival curves. Subsection 3.2 introduces a simple min-max lemma about the Network Calculus bounds on worst-case delay (resp. backlog) for these particular shapes. Subsection 3.3 simplifies the presentation of the problem which is finally solved in Subsection 3.4 by a polynomial algorithm.

3.2 Concave arrival curves and convex service curves

In Network Calculus, some specific results apply when arrival/service curves are concave/convex. Lemma 10 is a simple min-max lemma concerning the worst-case, *i.e.* maximal, delay (resp. backlog) bound.

We first define a notion of *stripe*. Let $\lambda \in \mathbb{R}_+ \cup \{+\infty\}$, a λ-*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 $\{(x,y) \in \mathbb{R}^2 \mid x_{\min} \leq x \leq x_{\max}\}$ 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 $width(S)$ (resp. $vsize(S)$, and $hsize(S)$) the euclidean (resp. vertical, and horizontal) 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 $hsize(S) = width(S)\frac{\sqrt{1+\lambda^2}}{\lambda}$ (with the convention that $0 \times \infty = \infty$).

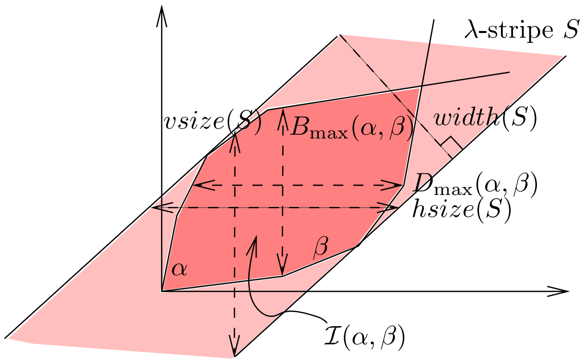

Fig. 5. An arrival curve α, a service curve β and a stripe S containing the area $\mathcal{I}(\alpha,\beta)$. The quantities $width(S)$, $hsize(S)$, $D_{\max}(\alpha,\beta)$ and $vsize(S)$, $D_{\max}(\alpha,\beta)$ are given in the figure.

Lemma 10 *Let α be a non-negative non-decreasing concave function, β a non-decreasing convex function s.t. $\beta(0) = 0$ 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 a node with service curve β, the maximal delay (resp. backlog) bound satisfies:*

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

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

PROOF. From the definitions of $D_{\max}(\alpha,\beta)$ and $B_{\max}(\alpha,\beta)$, the inequalities

$$D_{\max}(\alpha,\beta) \leq \inf\{hsize(S) \mid S \text{ is a } \lambda\text{−stripe}, , \ S \supseteq \mathcal{I}(\alpha,\beta)\} \quad (1)$$
$$B_{\max}(\alpha,\beta) \leq \inf\{vsize(S) \mid S \text{ is a } \lambda\text{−stripe}, , \ S \supseteq \mathcal{I}(\alpha,\beta)\}. \quad (2)$$

are clearly true (even if α is not concave and β is not convex). Fig. 5 illustrates the way the different measures relate to each other.

Now we wish to prove the inequalities in the other direction. We first consider $B_{\max}(\alpha, \beta)$. Let us first dismiss the cases when there exists $t \in \mathbb{R}_+$ such that $\alpha(t) = +\infty$. Due to the hypotheses on α, it implies that $\alpha = +\infty$ over $\mathbb{R}_+ \setminus \{0\}$. Then $B_{\max}(\alpha, \beta) = +\infty$ (which of course implies the other direction of Inequality (2)) unless $\beta = +\infty$ over $\mathbb{R}_+ \setminus \{0\}$ and $\alpha(0) \in \mathbb{R}_+$. In this latter case, $B_{\max}(\alpha, \beta) = \alpha(0)$ and $\mathcal{I}(\alpha, \beta) = \{0\} \times [0, \alpha(0)]$, which also implies the other direction of Inequality (2)

Now we can consider the main case when, for all $t \in \mathbb{R}_+$, $\alpha(t) \in \mathbb{R}_+$. Since α (resp. β) is concave (resp. convex), at any point $t \in \mathbb{R}_+$ of the function, it admits a right derivative and this slope is denoted $\partial_+\alpha(t)$ (resp. $\partial_+\beta(t)$), which may be $+\infty$ (possibly at $t = 0$ for α or when $\beta(t) = +\infty$). Moreover the function $\partial_+\alpha$ (resp. $\partial_+\beta$) is non-increasing (resp. non-decreasing) and non-negative (both α and β are non-decreasing). It implies that $\partial_+\alpha$ (resp. $\partial_+\beta$) has a limit (possibly $+\infty$) when t tends to $+\infty$, denoted $\partial_+\alpha_\infty$ (resp. $\partial_+\beta_\infty$). Moreover $\partial_+\alpha(t) < +\infty$ and $\partial_+\alpha_\infty < +\infty$ since $\alpha(t) \in \mathbb{R}_+$ for all $t \in \mathbb{R}_+$.

Suppose that $\partial_+\alpha_\infty > \partial_+\beta_\infty$. Due to the concavity (resp. convexity) of α (resp. β), we have $\alpha(t) \geq \partial_+\alpha_\infty \times t$ (resp. $\beta(t) \leq \partial_+\beta_\infty \times t$). Thus $\mathcal{I}(\alpha, \beta)$ contains the cone $\{(x, y) \in \mathbb{R}_+^2 | \partial_+\beta_\infty \times x \leq y \leq \partial_+\alpha_\infty \times x\}$, and $B_{\max}(\alpha, \beta) = \sup_{t \in \mathbb{R}_+}(\beta - \alpha)(t) = +\infty$. Since $B_{\max}(\alpha, \beta) = +\infty$, we have of course the other direction of Inequality (2).

Suppose that $\partial_+\alpha_\infty \leq \partial_+\beta_\infty$. Consider the function $\alpha - \beta$. It is concave, its right derivative tends to $\partial_+\alpha_\infty - \partial_+\beta_\infty$ when t tends to $+\infty$, and $(\alpha - \beta)(0) \geq 0$. Its supremum $B_{\max}(\alpha, \beta)$ is non-negative. Suppose that $B_{\max}(\alpha, \beta) = +\infty$, then of course the other direction of Inequality (2) is satisfied. Now suppose that $B_{\max}(\alpha, \beta) \in \mathbb{R}_+$. Since $\alpha - \beta$ is concave and $\partial_+\alpha_\infty - \partial_+\beta_\infty \geq 0$, either this supremum is reached for some $t_0 \in \mathbb{R}_+$, or it is the limit of $\alpha - \beta$ when t tends to $+\infty$ (this case may occur only if $\partial_+\alpha_\infty - \partial_+\beta_\infty = 0$).

In the first case, let us denote $\partial_-\alpha(t_0)$ (resp. $\partial_-\beta(t_0)$) the left derivative of α (resp. β) which exists at t_0. In case $t_0 = 0$, the convention $\partial_-\alpha(0) = +\infty$ (resp. $\partial_-\beta(0) = 0$) will work with the following arguments. At its supremum at t_0, $\alpha - \beta$ admits a left derivative $\partial_-(\alpha - \beta)(t_0) = \partial_-\alpha(t_0) - \partial_-\beta(t_0) \geq 0$ and a right derivative $\partial_+(\alpha - \beta)(t_0) = \partial_+\alpha(t_0) - \partial_+\beta(t_0) \leq 0$. Moreover by concavity and convexity of α and β, we also have $\partial_-\alpha(t_0) \geq \partial_+\alpha(t_0)$ and $\partial_-\beta(t_0) \leq \partial_+\beta(t_0)$. Combining all those inequalities, $\partial_+\beta(t_0)$ and $\partial_-\alpha(t_0)$ are both larger than $\partial_-\beta(t_0)$ and $\partial_+\alpha(t_0)$. Consider then any slope $\lambda < +\infty$ such that

$$\max(\partial_-\beta(t_0), \partial_+\alpha(t_0)) \leq \lambda \leq \min(\partial_+\beta(t_0), \partial_-\alpha(t_0)).$$

Then, the stripe of slope λ whose borders contain respectively the points

$(t_0, \alpha(t_0)$ and $(t_0, \beta(t_0))$ is of vertical size $\alpha(t_0)-\beta(t_0)$, and equal to $B_{\max}(\alpha,\beta)$ by definition of t_0. Moreover this stripe contains $\mathcal{I}(\alpha,\beta)$ because on both sides of t_0 the derivatives of α (resp. β) compare well to λ, placing the curve α below the top border and β above the bottom border. The other direction of Inequality (2) is then proved.

In the second case, the supremum $B_{\max}(\alpha,\beta)$ is the limit of $\alpha-\beta$ when t tends to $+\infty$, which occurs only if $\partial_+\alpha_\infty = \partial_+\beta_\infty$. Consider $\lambda = \partial_+\alpha_\infty < +\infty$. Since α is concave and $\partial_+\alpha_\infty \geq 0$, there exists $y_{\max} \in \mathbb{R}$ such that for all $t \in \mathbb{R}_+$, $y_{\max} + \lambda t \geq \alpha(t)$ and $\lim_{t\to+\infty}(y_{\max} + \lambda t - \alpha(t)) = 0$. Since β is convex and $\partial_+\beta_\infty = \lambda < +\infty$, there exists $y_{\min} \in \mathbb{R}$ such that for all $t \in \mathbb{R}_+$, $y_{\min} + \lambda t \leq \beta(t)$ and $\lim_{t\to+\infty}(\beta(t) - (y_{\min} + \lambda t)) = 0$. The λ-stripe between the lines $y_{\min} + \lambda t$ and $y_{\max} + \lambda t$ contains $\mathcal{I}(\alpha,\beta)$. Moreover its vertical size is $y_{\max} - y_{\min} = \lim_{t\to+\infty}((y_{\max} + \lambda t) - (y_{\min} + \lambda t)) = \lim_{t\to+\infty}(\alpha(t) - \beta(t)) = B_{\max}(\alpha,\beta)$. It proves the other direction of Inequality (2).

As for horizontal distance $D_{\max}$, the proof is essentially the same by considering the vertical distance between the pseudo-inverses $\alpha^{-1}(x) = \inf\{t \in \mathbb{R}_+ | \alpha(t) \geq x\}$ and $\beta^{-1}(x) = \inf\{t \in \mathbb{R}_+ | \beta(t) \geq x\}$ (with the convention $\inf\emptyset = +\infty$). ■

When α and β are also piecewise affine, one can refine this min-max lemma by considering only a selected set of stripes, as shown below.

Lemma 11 *With the assumptions of Lemma 10, 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 $(0, \alpha(0+))$ and with $\rho_m \stackrel{\text{def}}{=} +\infty$ if $\beta = +\infty$ from a point. Then*

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

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

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. It should be clear that the borders of this stripe touch $\mathcal{I}(\alpha,\beta)$ on both sides, that is respectively α and β (otherwise, a stripe with the same slope and a smaller horizontal size could be constructed). Fig. 6 displays such an example. If one border touches $\mathcal{I}(\alpha,\beta)$ at more than one point of α or β, it means that it touches a whole segment of α or β, and thus the slope of the stripe is the slope of a segment.

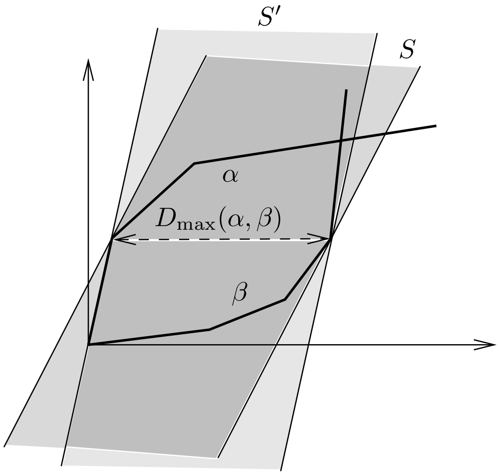

Fig. 6. A stripe S with horizontal size equal to $D_{\max}(\alpha,\beta)$. By tilting S, one gets S' a stripe with the same size as S whose slope is common with α (or β).

Otherwise, 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 $\mathcal{I}(\alpha,\beta)$. Then the slope of the new tilted stripe S' is necessarily the slope of one segment of α or β (see Fig. 6). ■

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[3]:

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

Lemma 12 *With the assumptions of Lemma 11 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).*

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

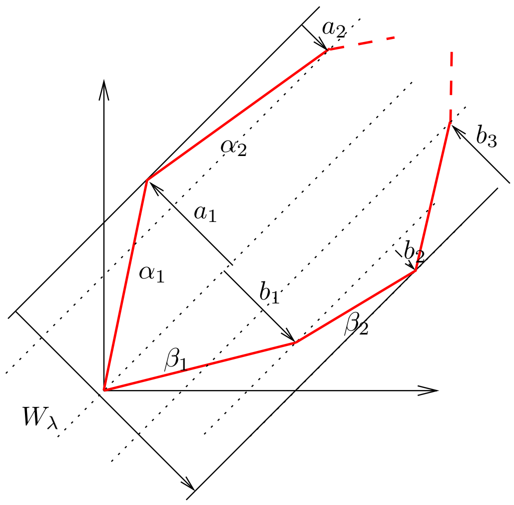

Fig. 7. The projection along direction λ of the segments α_1 and α_2 gives the quantities $a_1 < 0$ and $a_2 > 0$ respectively. All the other segments of α have slopes smaller than λ so that they are projected on positive quantities. The projections of $\beta_1, \beta_2, \beta_3$ are respectively $b_1 > 0$,$b_2 > 0$,$b_3 < 0$. 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 $W_\lambda = (-a_1)_+ + (-a_2)_+ + \sum_{i>2}(-a_i)_+ + b_{1+} + b_{2+} + b_{3+} + \sum_{i>3}(b_i)_+ = -a_1 + b_1 + b_2$.

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 mentioned before and linking $hsize(S)$, $vsize(S)$ and $width(S)$ for any λ-stripe S.

The formula for W_λ is illustrated by Figure 7. 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 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 12 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.

3.3 Transforming service curves into elementary service curves

Our study of the *optimal routing problem with no cross-traffic* assume that input functions are piecewise affine with a finite number of pieces. Let $f \in \mathcal{F}$ such a function, we will denote $|f|$ its number of affine pieces.

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 (illustrated on Fig. 8).

Lemma 13 ([6]) *Let β_1 and β_2 be two convex and piecewise affine functions over $\mathbb{R}_+$ with a finite number pieces. Let ρ_1 (resp. ρ_2) be the slope of the last semi-infinite segment of β_1 (resp. β_2) if it exists, or $\rho_1 = +\infty$ (resp. $\rho_2 = +\infty$) if β_1 (resp. β_2) is equal to $+\infty$ from a point. Then the convolution $\beta_1 * \beta_2$ first consists in concatenating, in the increasing order of the slopes and starting from $\beta_1(0) + \beta_2(0)$, all the segments of slope $< \min(\rho_1, \rho_2)$ of β_1 and β_2. It ends by adding a semi-infinite segments of slope $\min(\rho_1, \rho_2)$ if this value is finite.*

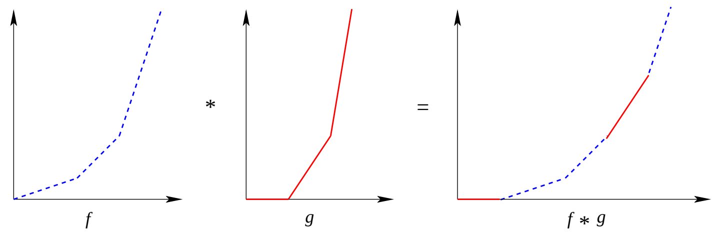

Fig. 8. Convolution of convex piecewise affine functions.

It can be exploited to transform the input network so that the service curve at each node is an elementary segment. Let v be a node 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 node v into k nodes $v_1, \ldots, v_k$. The service function of node 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 output 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 9 shows an example of the transformation. The service curve of the central node is composed of three segments. That node is replaced by three nodes, 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).

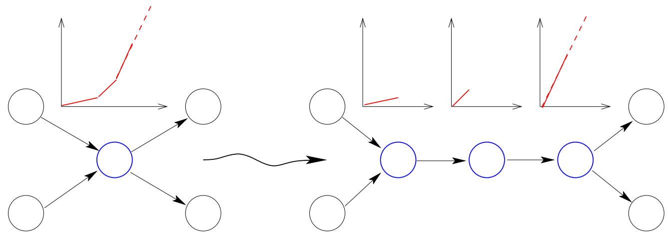

Fig. 9. Transformation of the graph.

Transforming all the nodes in this way gives a graph such that there is a one-to-one correspondence 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 correspondence, the overall service curve is the same. The transformation is linear in the size of the initial data: the number of nodes 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 14 *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 $G' = (V', A')$ satisfies $|V'| = \sum_{v \in V} |\beta_v|$, $|A'| = |A| + \sum_{v \in V}(|\beta_v| - 1)$, and the service curve at each node is an elementary segment.*

In the next subsection, we describe an algorithm 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 algorithm is more convenient when working with elementary segments. Once the algorithm is presented, we will nevertheless show how to adjust it to work directly with the initial service curves, not using the transformation, and consequently improve the overall complexity.

3.4 Routing algorithms for concave arrival curves and convex service curves

The next theorem states that the *optimal routing problem with no cross-traffic* with regard to delay or backlog is tractable, and as explained in Subsection 3.1 it can be used for the *optimal routing problem with independent cross-traffic*.

Theorem 15 *In the* optimal routing problem with no cross-traffic*, when the service curves in the network are convex and the arrival curve of the flow to route is concave, if all these functions are piecewise affine with finite numbers of pieces, then a path minimizing the end-to-end delay (resp. backlog) bound can be found in polynomial time thanks to Algorithm 1. The computational complexity for the general case, as well as for particular cases, is summed up in the next table:*

	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}

where the complexities are:

- ^{1} *= classical shortest path, e.g.* $\mathcal{O}(m + n \log n)$ *(see [7]),*
- ^{2} $= n$ *shortest paths, e.g.* $\mathcal{O}(mn + n^2 \log n)$ *(see [7]),*
- ^{3} $= \mathcal{O}((|\alpha| + \sum_{v \in V} |\beta_v|)(\log |\alpha| + \sum_{v \in V} \log |\beta_v| + m + n \log n))$

with the parameters:

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

PROOF. Here is an instance of the *optimal routing problem with no cross-traffic*:

- $G = (V, A)$ is the directed graph where one wish to route a flow on a path from v_{in} to v_{out}.
- The flow to route 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 $l_i \in \mathbb{R}_+ \cup \{+\infty\}$, $0 \leq i \leq p$. Note that α_0 corresponds to the segment from $(0,0)$ to $(0, \alpha(0+))$, thus $r_0 = +\infty$.
- Each node $v \in V$ has a service curve β_v which is non-decreasing, convex and piecewise affine with a finite number of concatenated segments.

An algorithm when service curves are elementary segments. The transformation of the graph exposed in Section 3.3 enables to decompose the service curves into elementary segments, and Proposition 14 ensures that it is sufficient to solve the problem for such instances with elementary segments as service curves. Thus we wish to design an algorithm for instances such that:

- Each node $v \in V$ has a service curve β_v which is an elementary segment of slope $\rho_v \in \mathbb{R}_+$ and length $\ell_v \in \mathbb{R}_+ \cup \{+\infty\}$.

We now discuss the optimization with regards to the end-to-end delay bound, but all the arguments also apply to the backlog version. 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 10, finding a path which minimizes the maximum delay with regard to α is equivalent to finding a path achieving the minimum

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

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 nodes. Due to Lemma 13, it is their concatenation by non-decreasing slopes. Thus applying Lemma 11, we know that for any path $\mathfrak{p}$, the minimum over λ is always reached for a slope $\lambda \in \{r_0, r_1, \ldots, r_p\} \cup \{\rho_v \mid v \in \mathfrak{p}\}$ which is included in the set $\{r_0, r_1, \ldots, r_p\} \cup \{\rho_v \mid v \in V\}$.

Inverting the two minima yields

$$D_{\max}(G) = \min_{\lambda \in \{r_0, r_1, \ldots, r_p\} \cup \{\rho_v \mid v \in V\}} \min_{\substack{\mathfrak{p}\ path\ from \\ v_{in}\ to\ v_{out}}} H_\lambda(\alpha, \beta(\mathfrak{p})).$$

For each $\lambda \in \{r_0, r_1, \ldots, r_p\} \cup \{\rho_v \mid v \in V\}$, from Lemma 12, $H_\lambda(\alpha, \beta(\mathfrak{p})) = \frac{\sqrt{1+\lambda^2}}{\lambda}\Big(\sum_{i=0}^{p}(-proj_\lambda(\alpha_i))_+ + \sum_{v \in \mathfrak{p}}(proj_\lambda(\beta_v))_+\Big)$. 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 node $v \in V$ 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 now analyze the overall 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 3.3, 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 nodes 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) [12]. 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 α 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 $\mathcal{O}((\sum_{v\in V} |\beta_v| + |\alpha|)(m + (\sum_{v\in V} |\beta_v|) \log(\sum_{v\in V} |\beta_v|) + \log |\alpha|))$.

An algorithm for the general case, avoiding the graph transformation. The presentation of an algorithm for elementary segments has some advantages, but once the principle is understood, one may try to speed up computations by working directly on the initial graph where each function β_v can have several segments. Reconsider Algorithm 1. All the arguments and computations correctly work when applying the algorithm for the initial instance. One only has to put the right weight at each node v, for each λ. A careful look shows that this weight must be the sum of the positive orthogonal projections along the direction λ of all the segments of β_v. This sum can be also described as the orthogonal projection along direction λ of the point of the curve β_v where the slope changes and becomes larger or equal to λ. This is exactly the kind of computation which has been discussed above for α and the same solution applies: at each node v, precompute the coordinates of all the points at angles of β_v where the slope changes, then store these points in a dictionary data structure using slopes as keys for the searches. For instance with a binary search tree or a sorted array, this data structure can be precomputed in $\mathcal{O}(|\beta_v| \log |\beta_v|)$. Then for each λ, the search for the right point to project is $\mathcal{O}(\log |\beta_v|)$ and its projection is done in constant time. It is of course a bit longer than working with elementary segments, but shortest path computations are performed on the initial graph which is smaller than the transformed graph. Up to a constant factor, the overall complexity is finally:

$$\underbrace{|\alpha| \log |\alpha|}_{dictionary\ for\ \alpha} + \sum_{v \in V} \underbrace{|\beta_v| \log |\beta_v|}_{dictionary\ for\ \beta_v} + \underbrace{(|\alpha| + \sum_{v \in V} |\beta_v|)}_{directions\ of\ projection} \Big(\underbrace{\log |\alpha| + \sum_{v \in V} \log |\beta_v|}_{computing\ weights} + \underbrace{m + n \log n}_{Dijkstra} \Big)$$

Algorithms for particular cases. For some particular shapes of functions (like affine arrival curves or rate-latency service curves), there exist more simple and efficient algorithms to compute the best worst-case delay bound or backlog bound [7]. ■

Remark 16 *Two classical optimal path problems from graph theory can be seen as particular cases of our problem. With arcs and/or nodes weighted by non-negative numbers, the first one consists in finding a* classical shortest path *from one source to one destination (minimizing the sum of the weights), and the latter one is to find a path with maximum bottleneck capacity (maximizing the smallest weight of the path). Those two problems exactly occur when the service curves β_v are all burst delay functions or are all peak rate functions (Lemma 13 gives the output of the convolution of such functions).*

Remark 17 *The core of our algorithm is the computation of orthogonal projections of curves along varying directions. Such projections along a direction λ of convex functions correspond to the values of their* Legendre-Fenchel Transform *at λ, up to constant factor only depending on λ. The Legendre-Fenchel Transform $f(t) \mapsto \mathcal{C}(\lambda) = \sup_t(\lambda t - f(t))$, also called convex conjugate function or $\mathcal{C}$-transform [9], is a powerful tool of convex analysis [27,36]. In some way which still has to be formalized, our algorithm works with a Legendre-Fenchel Transform of the instances.*

Remark 18 *The routing algorithm can be used for the discrete time model by working with the piecewise affine interpolation of the input functions originally defined from $\mathbb{N}$ into $\mathbb{R}_{\min}$. If these piecewise affine interpolations defined from $\mathbb{R}_+$ into $\mathbb{R}_{\min}$ satisfies the hypotheses of Theorem 15, one can compute an optimal route with Algorithm 1. If backlog is optimized, its minimum value computed with the interpolations is exactly the same as the one that would have been computed by performing discrete time convolutions [8]. If delay is optimized, the minimum value computed with the interpolations can be different from the one that would have been computed in the discrete time model, i.e. it may belongs to $\mathbb{R}_+ \setminus \mathbb{N}$. However one can check that the discrete time value is then obtained by round-ceiling the value in $\mathbb{R}_+$, and then the optimal path computed with interpolations is also optimal with regard to discrete time.*

4 Optimal routing with general cross-traffic

In the previous section, independent cross-traffic has been tackled by considering the convolution of the residual service curves over a path in order to have an end-to-end service curve. When cross-traffic is not independent, *i.e.* when cross-traffic flows have paths that may cross several nodes of the graph, the previous approach collapses and providing a valid analysis of the performances is more difficult. When the graph is acyclic, one can find an end-to-end service curve for a path using a method suggested in [6]: one has first to fix the path of the routed flow, then follow a topological sort of the graph to compute the arrival curve of each flow on each arc (at each node, the computation of output arrival curves uses Theorem 3 and Lemma 5), and finally an end-to-end service curve can be computed by a convolution of residual services (using Lemma 5 with the arrival curves computed just before). Such an analysis is performed on a simple example in Subsection 4.3. However a first drawback of this approach is that one has to fix at first the path of the routed flow before leading any analysis. A second one is that the bounds obtained in this way can be loose as observed by Schmitt et al [38,39]. These authors have presented an alternative analysis which can be applied in some situation and which gives tighter bounds quite often. This gain is attributed to what they call the *Pay Multiplexing Only Once* (PMOO) phenomenon (see [38,39] for a discussion about the appellation). The whole section is devoted to the routing problem when taking into account PMOO. The first issue comes from the computation of a service curve over a single path. It is addressed in Subsections 4.1 and 4.2, while the problem of optimized routing is addressed in Subsection 4.4. We carefully detail the configurations where those analyses work.

There exists an interesting formula which gives a service curve for a path in case the cross-traffic flows interferes over *sub-paths*, *i.e.* sets of consecutive nodes. It generalizes a formula obtained in [39] for a small example with three nodes and two cross-traffic flows (Example 21).

Here are the configurations we study and the notation we use, illustrated by Fig. 10 and Example 21:

- The main flow F_0 follows a path $\mathfrak{p}$ of n nodes indexed 1,2,...,n with respective *strict* service curves β_j, $1 \leq j \leq n$.
- The cross-traffic interfering with the path is composed of k flows F_i, $1 \leq i \leq k$, with respective arrival curve α_i.
- Each cross-traffic flow F_i intersects the path $\mathfrak{p}$ over a set of consecutive nodes of $\mathfrak{p}$. It joins the path at node s_i and leaves it just after node e_i (in particular $s_0 = 1$ and $e_0 = n$).
- Consider a flow i, $1 \leq i \leq k$, the amount of data that have been served by node j until time t is denoted $A_i^{(j)}(t)$. The amount of data that have entered node s_i until time t is denoted $A_i^{(s_i-1)}(t)$.
- As in the rest of the paper, we assume *blind multiplexing* at the nodes.

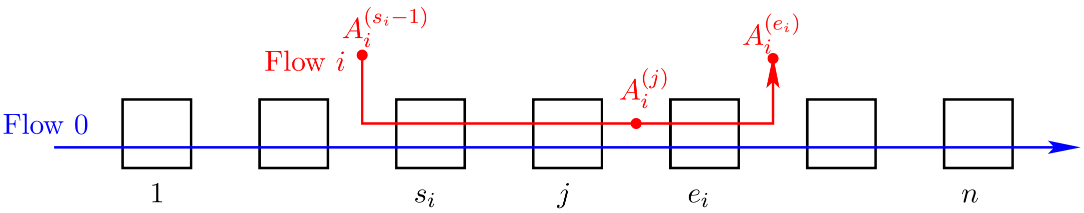

Fig. 10. Path where each cross-traffic flow i over a sub-path.

We first state a lemma for the main flow linking $A_0^{(n)}$ to $A_0^{(0)}$ up to a sequence of jumps $u_1, \ldots, u_n$ in the past.

Lemma 19 *With the preceding assumptions and notation, $\forall t \in \mathbb{R}_+$, $\exists u_1, \ldots, u_n \in \mathbb{R}_+$ such that*

$$A_0^{(n)}(t) - A_0^{(0)}(t - \sum_{j=1}^{n} u_j) \geq \sum_{j=1}^{n} \beta_j(u_j) - \sum_{i=1}^{k} \Big(A_i^{(e_i)}(t - \sum_{j=e_i+1}^{n} u_j) - A_i^{(s_i-1)}(t - \sum_{j=s_i}^{n} u_j) \Big)$$

Moreover, $A_0^{(n)}(t) - A_0^{(0)}(t - \sum_{j=1}^{n} u_j) \geq 0$.

PROOF. For all $1 \leq j \leq n$, let $I_j \subseteq \{0, 1, \ldots, k\}$ be the indices of the flows that cross node j. First note that for all $t \in \mathbb{R}_+$ and for any node j, if the backlog of the aggregated flow is null, *i.e.* $\sum_{i \in I_j} A_i^{(j-1)}(t) = \sum_{i \in I_j} A_i^{(j)}(t)$, then the backlog of each individual flow is also null, *i.e.* $A_i^{(j-1)}(t) = A_i^{(j)}(t)$ for all $i \in I_j$ (since we always have $A_i^{(j-1)}(t) \geq A_i^{(j)}(t)$)

Let $t \in \mathbb{R}_+$ and consider the following sequence of events:

- let $t - u_n$ be the start of the backlogged period of t at node n,
- let $t - u_n - u_{n-1}$ be the start of the backlogged period of $t - u_n$ at node $n-1$,
- ...
- let $t - \sum_{J=j}^{n} u_J$ be the start of the backlogged period of $t - \sum_{J=j+1}^{n} u_J$ at node j,
- ...
- let $t - \sum_{J=1}^{n} u_J$ be the start of the backlogged period of $t - \sum_{J=2}^{n} u_J$ at node 1.

Since the function $(\beta_j)_{1 \leq j \leq n}$ are *strict* service curves, we have:

$$\begin{array}{lrcl}
\textbf{Node } n: & \sum_{i \in I_n} A_i^{(n)}(t) & \geq & \beta_n(u_n) + \sum_{i \in I_n} A_i^{(n-1)}(t - u_n) \\
\textbf{Node } n\text{–}1: & \sum_{i \in I_{n-1}} A_i^{(n-1)}(t - u_n) & \geq & \beta_{n-1}(u_{n-1}) + \sum_{i \in I_{n-1}} A_i^{(n-2)}(t - u_n - u_{n-1}) \\
\vdots & & \vdots & \\
\textbf{Node } j: & \sum_{i \in I_j} A_i^{(j)}(t - \sum_{J=j+1}^{n} u_J) & \geq & \beta_j(u_j) + \sum_{i \in I_j} A_i^{(j-1)}(t - \sum_{J=j}^{n} u_J) \\
\vdots & & \vdots & \\
\textbf{Node } n: & \sum_{i \in I_1} A_i^{(1)}(t - \sum_{J=2}^{n} u_J) & \geq & \beta_1(u_1) + \sum_{i \in I_1} A_i^{(0)}(t - \sum_{J=1}^{n} u_J)
\end{array}$$

Summing all those inequalities and collecting terms with regard to flow indices, we get:

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

By removing terms present on both sides, it simplifies into:

$$\sum_{i=0}^{k} A_i^{(e_i)}(t - \sum_{J=e_i+1}^{n} u_J) \geq \sum \beta_j(u_j) + \sum_{i=0}^{k} A_i^{(s_i-1)}(t - \sum_{J=s_i}^{n} u_J)$$

which gives the first inequality of the lemma.

Considering our sequence of starts of backlogged periods, we also have:

$$\begin{aligned}A_0^{(n)}(t) &\geq A_0^{(n)}(t-u_n) = A_0^{(n-1)}(t-u_n)\\ &\geq A_0^{(n-1)}(t-u_n-u_{n-1}) = A_0^{(n-1)}(t-u_n-u_{n-1})\\ &\geq \cdots\\ &\geq A_0^{(1)}(t-\sum_{j=1}^{n} u_j) = A_0^{(0)}(t-\sum_{j=1}^{n} u_j)\end{aligned}$$

which gives the second inequality. ■

One can deduce from this lemma a service curve provided by the whole path to the main flow F_0. It is defined by a formula which is independent of the specifications of flow F_0, thus it can really be seen as a service curve of the path.

Theorem 20 (PMOO multi-dimensional operator) *With the same assumptions and notation as above, a service curve offered by the whole path* $\mathfrak{p}$ *is:*

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

PROOF. Consider the inequalities proved in Lemma 19. We clearly have $\forall i \in \{1\dots,k\}$, $\forall j \in \{s_i,\dots,e_i\}$, $\forall t \in \mathbb{R}_+$, $A_i^{(j)}(t) \leq A_i^{(s_i-1)}(t)$. Moreover for all $1 \leq i \leq k$,

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

Thus,

$$A_0^{(n)}(t) - A_0^{(0)}(t-\sum_{j=1}^{n} u_j) \geq \sum_{j=1}^{n} \beta_j(u_j) - \sum_{i=1}^{k} \alpha_i(\sum_{j=s_i}^{e_i} u_j).$$

From Lemma 19, we also have $A_0^{(n)}(t)-A_0^{(0)}(t-\sum_{j=1}^{n} u_j) \geq 0$. As a consequence of those two inequalities, we have $A_0^{(n)} \geq A_0^{(0)} * \phi$ where ϕ is the function defined above. ■

This result introduces a new multi-dimensional operator for Network Calculus. It generalizes Lemma 5 which tackles paths with a single node and an unique cross-traffic flow. The theorem is inspired by [39] where the next example is analyzed.

Example 21 *To illustrate the PMOO phenomenon, Schmitt et al [39] have detailed the analysis of the configuration pictured on Fig. 11: a path with three nodes and two cross-traffic flows. Theorem 20 provides the following service curve for the path:*

$$\phi(t) = \left(\inf_{\substack{u_1, u_2, u_3 \geq 0 \\ u_1 + u_2 + u_3 = t}} \beta_1(u_1) + \beta_2(u_2) + \beta_3(u_3) - \alpha_1(u_1 + u_2) - \alpha_2(u_2 + u_3) \right)_+ .$$

This formula actually appears during the analysis in [39] (Section 4).

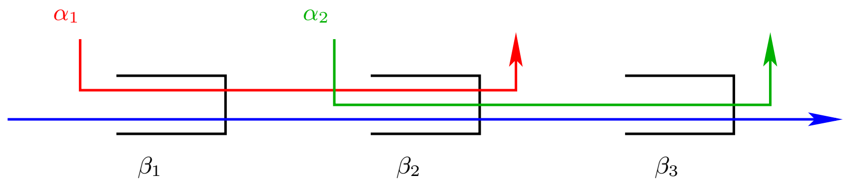

Fig. 11. Schmitt et al's example [39].

Remark 22 *It can be easily seen that the formula of Theorem 20 and its proof also apply in the discrete time model. It should be also reminded that this theorem provides a service curve which is not necessarily* strict *as explained for Lemma 5 in Remark 7.*

Theorem 20 applies for general arrival curves α_i and strict service curves β_j. Now come the algorithmic aspects of the formula. For piecewise affine functions, in case all the curves α_i are affine and all the curves β_j are convex, one can design an efficient algorithm computing ϕ as shown in the next subsection.

4.2 Computation of the PMOO service curve of a path

The formula for ϕ is not easy to simplify, using only the classical network calculus operators. Here is one possible simplification bounding ϕ by a convolution. We use the notation of the preceding subsection.

Proposition 23 *Under the assumptions and notation of Theorem 20, if all the arrival curves α_i are sub-additive, then the PMOO service curve ϕ satisfies:*

$$\phi(t) \geq (\tilde{\beta}_1 * \cdots * \tilde{\beta}_n)_+(t)$$

with $\tilde{\beta}_j = \beta_j - \sum_{i \in I_j} \alpha_i$ where I_j is the set of indices of flows crossing node j. This inequality becomes an equality if all the functions α_i are affine.

PROOF. In the formula of Theorem 20, if the arrival curves are sub-additive, we have

$$\alpha_i(\sum_{j=s_i}^{e_i} u_j) \leq \sum_{j=s_i}^{e_i} \alpha_i(u_j). \tag{3}$$

As a consequence,

$$\phi(t) \geq \Big(\inf_{\substack{u_1,\ldots,u_n \geq 0 \\ u_1+\cdots+u_n=t}} \sum_{j=1}^{n} \tilde{\beta}_j(u_j)\Big)_+ = (\tilde{\beta}_1 * \cdots * \tilde{\beta}_n)_+(t)$$

with $\tilde{\beta}_j = \beta_j - \sum_{i\in I_j} \alpha_i$.

When the functions α_i are affine, Inequality (3) becomes an equality. ■

Note that, although this lower bound to ϕ is not tight unless the functions α_i are affine, it would be interesting to have a precise comparison of the two formulas.

Another useful operator in Network Calculus is the *sub-additive closure* of a function $\alpha \in \mathcal{F}$ denoted by α^* and defined as: $\alpha^* \stackrel{\text{def}}{=} \inf_{p\in\mathbb{N}} \alpha^{(p)}$ where $\alpha^{(0)}(t) = 0$ if $t = 0$ and $= +\infty$ if $t > 0$, and for $p \geq 1$, $\alpha^{(p)} = \underbrace{\alpha * \cdots * \alpha}_{p \text{ times}}$. The preceding proposition can be used to provide another service curve for the whole path.

Corollary 24 *Under the assumptions and notation of Theorem 20, a service curve offered by the whole path $\mathfrak{p}$ is $(\hat{\beta}_1 * \cdots * \hat{\beta}_n)_+$ with $\hat{\beta}_j = \beta_j - \sum_{i\in I_j} \alpha_i^*$ where I_j is the set of indices of flows crossing node j, and α_i^* is the sub-additive closure of α_i.*

PROOF. For each cross-traffic flow F_i, the arrival curve α_i can be replaced by its sub-additive closure α_i^*. It remains an arrival curve for F_i and it is sub-additive [6]. Then one can apply Proposition 23. ■

Proposition 23 also enables to compute efficiently the PMOO service curve ϕ when the arrival curves are affine and the strict service curves are piecewise affine and convex.

Corollary 25 *Under the assumptions and notation of Theorem 20, if the k arrival curves α_i are* affine *and the n strict service curves β_j are* convex *and* piecewise affine *with a finite number of pieces, then the PMOO service curve ϕ can be computed in $\mathcal{O}(c + (\sum_{j=1}^{n} |\beta_j|) \log_2 n)$ time, where c is the number of* *pairs (i, j) such that flow i crosses node j. In case the strict service curves are* rate-latency *functions, it can be computed in $\mathcal{O}(c)$ time.*

PROOF. In case the functions α_i are affine, Proposition 23 states that $\phi = (\tilde{\beta}_1 * \cdots * \tilde{\beta}_n)_+$ where $\tilde{\beta}_j = \beta_j - \sum_{i \in I_j} \alpha_i$ and I_j is the set of indices of flows crossing node j. Moreover if β_j is convex and piecewise affine with $|\beta_j|$ pieces, then $\tilde{\beta}_j$ remains convex and piecewise affine with $|\beta_j|$ pieces. Each function $\tilde{\beta}_j$ can be computed from β_j in $\mathcal{O}(|\beta_j| + |I_j|)$ time. This initial step has a $\mathcal{O}(c + \sum_{j=1}^{n} |\beta_j|)$ complexity with $c \stackrel{\text{def}}{=} \sum_{j=1}^{n} |I_j|$. Then one can use Lemma 13 to compute $\tilde{\beta}_1 * \cdots * \tilde{\beta}_n$. For all $1 \leq j \leq n$, if $\tilde{\beta}_j$ ends by a semi-infinite segment, let r_j be its slope, otherwise let $r_j = +\infty$. Consider $r = \min_{1 \leq j \leq n} r_j$, then ϕ is the concatenation of all the finite pieces of slope $< r$, of the functions $\tilde{\beta}_j$, $1 \leq j \leq n$, in non-decreasing order of the slopes, starting at $\phi(0) = \sum_{j=1}^{n} \tilde{\beta}_j(0)$ and ended by a semi-infinite segment of slope r (if $r < +\infty$). The pieces are already sorted with regard to slopes, in each $\tilde{\beta}_j$. Thus the computation comes to merging n sorted lists whose respective sizes are $\leq |\beta_j|$. It can be done in $\mathcal{O}((\sum_{j=1}^{n} |\beta_j|) \log_2 n)$ time [12]. Once the function $\tilde{\beta}_1 * \cdots * \tilde{\beta}_n$ is computed, since it is convex and piecewise affine with $\leq \sum_{j=1}^{n} |\beta_j|$ pieces, the function $\phi = (\tilde{\beta}_1 * \cdots * \tilde{\beta}_n)_+$ can be easily computed in linear time by a single pass through the pieces of $\tilde{\beta}_1 * \cdots * \tilde{\beta}_n$.

Now consider the more restrictive case when the strict service curves are rate-latency functions. Let $\alpha_i(t) = \sigma_i + \rho_i t$, $1 \leq i \leq k$, and $\beta_j(t) = R_j(t - T_j)_+$, $1 \leq j \leq n$. Then applying the formula of Theorem 20 yields for all $t \in \mathbb{R}_+$:

$$
\begin{aligned}
\phi(t) &= \Bigg(\inf_{\substack{u_1, \ldots, u_n \geq 0 \\ u_1 + \cdots + u_n = t}} \sum_{j=1}^{n} \beta_j(u_j) - \sum_{i=1}^{k} \alpha_i \Big(\sum_{j=s_i}^{e_i} u_j \Big) \Bigg)_+ \\
&= \Bigg(-\sum_{i=1}^{n} \sigma_i + \inf_{\substack{u_1, \ldots, u_n \geq 0 \\ u_1 + \cdots + u_n = t}} \sum_{j=1}^{n} \Big(R_j(u_j - T_j)_+ - \big(\sum_{i \in I_j} \rho_i \big) u_j \Big) \Bigg)_+ \\
&= \Bigg(-\sum_{i=1}^{n} \sigma_i + (\gamma_1 * \cdots * \gamma_n)(t) \Bigg)_+
\end{aligned}
$$

where, for all $1 \leq j \leq n$, $\gamma_j(t) = R_j(t - T_j)_+ - (\sum_{i \in I_j} \rho_i) t$ is a convex piecewise affine with two pieces from $\gamma(0) = 0$: a segment of slope $-\sum_{i \in I_j} \rho_i$ for $t \in [0, T_j]$ and a semi-infinite segment of slope $R_j - \sum_{i \in I_j} \rho_i$ for $t \in [T_j, +\infty[$.

Using Lemma 13 to compute $\gamma_1 * \cdots * \gamma_n$, one gets that ϕ is a rate-latency function $\phi(t) = R(t-T)_+$ with

$$R = \min_{1 \le j \le n} \left(R_j - \sum_{i \in I_j} \rho_i \right) \text{ and } T = \sum_{i=1}^{k} \frac{\sigma_i}{R} + \sum_{j=1}^{n} T_j \left(1 + \sum_{i \in I_j} \frac{\rho_i}{R} \right).$$

■

Note that all the results of this subsection also hold in the discrete time model.

Remark 26 (Schmitt et al's algorithm) *Another method to compute the PMOO service curve ϕ is suggested by [38,39] to deal with Example 21 when arrival curves are concave and strict service curves are convex. The idea is to decompose each function α_i as a minimum of affine functions and each function β_j as a maximum of rate-latency functions. Then one computes the PMOO service curve for each combination of affine arrival curves and rate-latency strict service curves. It can be done easily as shown in [38,39] for Example 21 or more generally with the preceding Corollary 25. Then they suggest to reconstruct the PMOO service curve ϕ for the initial input functions by computing the maximum of the PMOO service curves already obtained for all the possible elementary combinations. Their approach based on a decomposition into elementary functions and then a reconstruction is very interesting, but it has two main drawbacks.*

First, one can be tempted to compare their algorithm with our algorithm when arrival curves are affine. However as far as we know they do not compute the same function ϕ. Their claim that one can exactly compute the PMOO service curve ϕ by taking the maximum of the PMOO service curves over all combinations of elementary functions, is unfortunately false in general. It is based on a wrong lemma proving the commutativity between supremum and infimum (Lemma 3, page 7, in [39]). It is not clear which assumptions should be added so that their algorithm actually computes ϕ. For now, it can only be stated that they compute a lower bound to ϕ, which is consequently also a service curve for the path.

Concerning the computational complexity of their approach, in Example 21, the number of combinations to consider is $|\beta_1|\cdot|\beta_2|\cdot|\beta_3|\cdot|\alpha_1|\cdot|\alpha_2|$ and one may guess that in the general case of Theorem 20 the complexity of this approach will have a lower bound $\Omega(|\beta_1|\cdots|\beta_n|\cdot|\alpha_1|\cdots|\alpha_k|)$. When arrival curves are affine ($|\alpha_i| = 1$), it has a much higher complexity than the algorithm of Corollary 25 although they do not necessarily compute the same functions. The underlying problem of the Schmitt et al's algorithm is that many combinations output an end-to-end service curve which won't appear in the final service curve ϕ.

Remark 27 *In [7], a polynomial algorithm was presented to compute ϕ for* *piecewise affine input functions when the strict service curves are convex and the arrival curves are concave. Unfortunately this algorithm does not work in general. Consider the path of Fig. 12, it offers a PMOO service curve*

$$\phi(t) = \Big(\inf_{\substack{u_1,\dots,u_n \geq 0 \\ u_1+\cdots+u_n = t}} \beta_1(u_1)+\beta_2(u_2)+\beta_3(u_3)-\alpha_1(u_1)-\alpha_2(u_1+u_2)-\alpha_3(u_2+u_3)-\alpha_3(u_3)\Big)_+$$

Suppose that $\beta_j(t) = 2t$, $j \in \{1,2,3\}$, and $\alpha_i(t) = \min(t,1)$, $i \in \{1,2,3,4\}$. Then we have:

$$\phi(t) = \Big(\inf_{\substack{u_1,\dots,u_n \geq 0 \\ u_1+\cdots+u_n = t}} \beta_{1,1}(u_1) + \beta_{1,1}(u_1+u_2) + \beta_{1,1}(u_2+u_3) + \beta_{1,1}(u_3)\Big)_+$$

with the rate-latency function $\beta_{1,1}(t) = (t-1)_+$. The algorithm presented in [7] may output the rate-latency function $\beta_{1,1}(t) = (t-1)_+$ as ϕ (it occurs when the algorithm chooses to increment u_2 at first). It is a wrong answer: in this example, a few calculations show that ϕ is the rate-latency function $\beta_{1,2}(t) = (t-2)_+$.

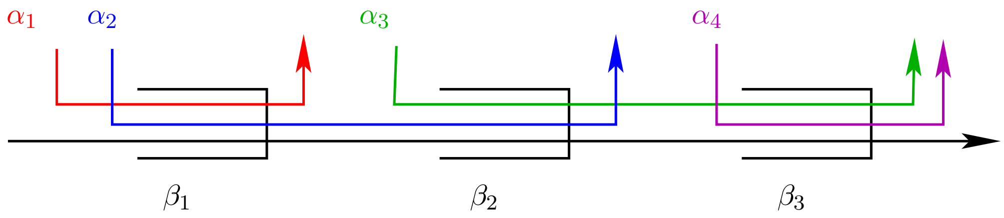

Fig. 12. A counter-example to the algorithm of [7] computing PMOO service curves.

4.3 Tightness of the PMOO analysis

Before addressing the routing problem with the PMOO service curve given by Theorem 20, one may wonder how tight are the bounds on delay and backlog provided by this formula. We can already draw a first comparison with the node-by-node analysis suggested in [6] and described at the beginning of the section. Consider the toy example pictured on Fig. 13, where the main flow F_0 with arrival curve α_0, as well as a cross-traffic flow F_1 with arrival curve α_1, go through two nodes with respective *strict* service curves β_1 and β_2.

The node-by-node analysis first reviews Node 1. Due to Lemma 5 for blind multiplexing, we can only say that a service curve $(\beta_1 - \alpha_1)_+$ is offered to Flow 0 and a service curve $(\beta_1 - \alpha_0)_+$ is offered to Flow 1. Then Theorem 3 ensures that when leaving Node 1, Flow 1 (resp. Flow 0) has an arrival curve $\alpha_1 \oslash (\beta_1 - \alpha_0)_+$ (resp. $\alpha_0 \oslash (\beta_1 - \alpha_1)_+$. Then the analysis of Node 2 with Lemma 5 again shows that it offers a service curve $(\beta_2 - \alpha_1 \oslash (\beta_1 - \alpha_0)_+)_+$ to

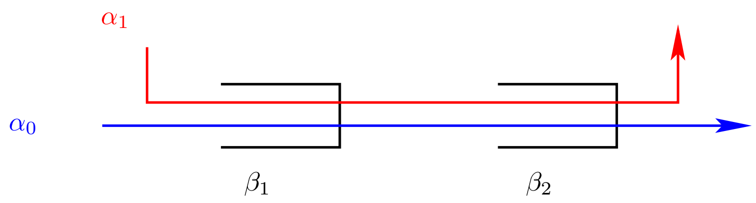

Fig. 13. A simple example to investigate *Pay Multiplexing Only Once*.

Flow 0. Thanks to the Tandem Lemma 4, a end-to-end service curve offered to Flow 0 is $\beta_{Classic} = (\beta_1 - \alpha_1)_+ * (\beta_2 - \alpha_1 \oslash (\beta_1 - \alpha_0)_+)_+$.

Another way to analyze the path is to apply Theorem 20 which ensures a service curve $\beta_{PMOO} = (\beta_1 * \beta_2 - \alpha_1)_+$ to Flow 0.

Unfortunately none of these two service curves is always better than the other: neither $\beta_{PMOO} \geq \beta_{Classic}$ nor $\beta_{Classic} \geq \beta_{PMOO}$ always holds. Suppose that $\alpha_i(t) = \sigma_i + \rho_i t$, $i \in \{0, 1\}$, and $\beta_j = R_j(t - T_j)_+$, $j \in \{1, 2\}$, then $\forall t \in \mathbb{R}$,

$$\beta_{Classic}(t) = (\min(R_1, R_2) - \rho_1)\left(t - \frac{R_2T_2 + \sigma_1 + \rho_0\frac{R_1T_1+\sigma_0}{R_1-\rho_0}}{R_2 - \rho_1} - \frac{R_1T_1 + \sigma_1}{R_1 - \rho_1}\right)_+$$
$$\beta_{PMOO}(t) = (\min(R_1, R_2) - \rho_1)\left(t - \frac{\min(R_1, R_2)(T_1 + T_2) + \sigma_1}{\min(R_1, R_2) - \rho_1}\right)_+.$$

Let R_2 tend to $+\infty$ and set $\sigma_1 = 0$, then $\beta_{Classic}(t) \to (R_1 - \rho_1)\left(t - \frac{R_1T_1}{R_1-\rho_1} - T_2\right)_+$ and $\beta_{PMOO}(t) \to (R_1 - \rho_1)\left(t - \frac{R_1T_1}{R_1-\rho_1} - \frac{R_1T_2}{R_1-\rho_1}\right)_+$. Thus for an appropriate choice of the parameters and for some $t \in \mathbb{R}_+$, $\beta_{Classic}(t) > \beta_{PMOO}(t)$.

Set $R_1 \leq R_2$, $T_2 = T_1 = \sigma_1 = 0$, then $\beta_{Classic}(t) = (R_1 - \rho_1)\left(t - \frac{\rho_0}{R_2-\rho_1}\frac{\sigma_0}{R_1-\rho_0}\right)_+$ and $\beta_{PMOO}(t) = (R_1 - \rho_1)t_+$. If $\sigma_0 > 0$, then for some $t \in \mathbb{R}_+$, $\beta_{Classic}(t) < \beta_{PMOO}(t)$.

The search of tight bounds and end-to-end service curves requires further investigations. Nevertheless, an experimental study has been led in [38,39] to compare the PMOO analysis and the node-by-node analysis (that they call *straightforward analysis*). The results are considerably better with the PMOO analysis. As a consequence, using the PMOO service curve of Theorem 20 for the cost function of our routing problem is a relevant choice.

4.4 Optimal routing under PMOO

We now wish to reconsider the initial routing problem with cross-traffic (referred as Problem 8) using the PMOO service curve of Theorem 20 for the cost function bounding end-to-end delays or backlogs.

We first discuss the case of general acyclic graph and then some special cases where the structure of cross-traffic enables to speed up computations of optimal paths.

4.4.1 General acyclic graphs

Let us consider an instance of Problem 8 , that is a directed graph $G = (V, A)$, *strict* services curves $(\beta_v)_{v\in V}$ at the nodes, k cross-traffic flows F_i with respective paths $(\mathfrak{p}_i)_{1\leq i\leq k}$ and arrival curves $(\alpha_i)_{1\leq i\leq k}$. The flow to route from v_{in} to v_{out}, denoted F_0, has an arrival curve α_0.

We are looking for a cost function for delays or backlogs taking into account PMOO. As a first remark, some additional assumptions on the shape of constraint functions are required to compute some PMOO service curves. We choose to work with *convex piecewise affine strict service curves* with a finite number of pieces and *affine arrival curves*, so that the algorithm of Corollary 25 applies.

Consider a *fixed* path $\mathfrak{p}_o$ from v_{in} to v_{out}. One can compute a service curve provided by this path to flow F_0 in the following way. Since the network is acyclic, the nodes can be sorted in linear time, according to a *topological order* of the graph [12]. Let us fix such a topological order on $V = \{v_1, \dots, v_n\}$. We denote by $w \preccurlyeq v$ that node w is node v or comes before node v in the topological order. For all $0 \leq i \leq k$ and for each node $v \in \mathfrak{p}_i$, we inductively define $\phi_v^i(t)$ as below, so that Theorem 20 ensures it is an end-to-end service curve offered by the network to flow F_i over its sub-path starting at the beginning of $\mathfrak{p}_i$ and ending at node v:

$$\phi_v^i(t) = \Big(\inf_{\sum_{j\in J} u_j = t} \sum_{j\in J} \beta_{v_j}(u_j) - \sum_{h\in H} \sum_{\ell=1}^{c_i^h} \alpha_h \oslash \phi^h_{v^h(\ell)}\big(\sum_{v_j\in\mathfrak{p}_h(\ell)} u_j\big)\Big)_+ \tag{4}$$

where

- $J = \{j \mid v_j \preccurlyeq v \text{ and } v_j \in \mathfrak{p}_i\}$.
- $H = \{h \mid \exists w \in \mathfrak{p}_i \cap \mathfrak{p}_h,\ w \preccurlyeq v\}$.
- If $h \in H$, c_i^h is the number of common sub-paths between $\mathfrak{p}_i$ and $\mathfrak{p}_h$ before v.
- If $h \in H$, $v^h(\ell)$ is the node on $\mathfrak{p}_h$ just before F_i and F_h meet for the ℓ^{th} time if it exists (otherwise $\alpha_h \oslash \phi^h_{v^h(\ell)}$ must be replaced by α_h in the equation).
- If $h \in H$, $\mathfrak{p}_h(\ell)$ is the ℓ^{th} common sub-path of $\mathfrak{p}_i$ and $\mathfrak{p}_h$.

It uses the fact that an arrival curve for flow F_i just after node v is $\alpha_i \oslash \phi_v^i$ due to Theorem 3. If α_i is affine, Example 6 shows that $\alpha_i \oslash \phi_v^i$ remains affine and can be computed in linear time if ϕ_v^i is piecewise affine and convex. This latter

Algorithm 2. Computation of a best route in acyclic graphs

Require: an acyclic digraph $G = (V, A)$, strict service curves $(\beta_v)_{v \in V}$, cross-traffic paths $(\mathfrak{p}_i)_{1 \le i \le k}$ and arrival curves $(\alpha_i)_{1 \le i \le k}$, arrival curve α, source v_{in} and destination v_{out} of the flow to route.

Ensure: a path $\mathfrak{p}$ providing some "best" bound on the worst-case delay (resp. backlog) for the routed flow;

Compute a topological order of G;

for all paths $\mathfrak{p}$ in G from v_{in} to v_{out} **do**

Compute ϕ_v^i for all $1 \le i \le v$ and $v \in \mathfrak{p}_i$ using Equation (4) along the topological order;

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

property is preserved by Equation (4) if the initial curves β_v are piecewise affine and convex.

The end-to-end service curve of the main flow over path $\mathfrak{p}_0$ is finally $\phi^0_{v_{out}}$. Note that all the functions ϕ_v^i depend on $\mathfrak{p}_0$, the path chosen for the main flow F_0 from v_{in} to v_{out}. Algorithm 2 describes without details an exhaustive search of an optimal route with regard to the cost functions associated with $\phi^0_{v_{out}}$.

We mainly present this algorithm to show that it is possible to do some computations. However this approach has many issues. First we do not exactly know the quality of our performance analysis using the intermediate service curves ϕ_v^i. Then even if we obtain tight bounds with regard to end-to-end worst-case delays or backlogs, this brute force algorithm has a high complexity (*e.g.* the number of paths between v_{in} and v_{out} can be exponential in the number of nodes).

4.4.2 Strongly acyclic cross-traffic

Restrictions on the cross-traffic and the shapes of the input curves By adding some assumptions on the cross-traffic structure and on the shapes of the input curves, one can design faster routing algorithms taking into account PMOO.

Definition 28 *Given an instance of the general routing problem (Problem 8) where the graph is acyclic, the cross-traffic is called* strongly acyclic *if for any pair of nodes in a connected component of the sub-graph $\mathcal{T}$ obtained by keeping only the arcs used by the cross-traffic, the two nodes are connected by at most one path in the initial graph and this path necessarily belongs to the sub-graph $\mathcal{T}$.*

An example is shown in the next subsection on Fig. 14. This definition ensures that any path in the graph intersects any cross-traffic flow at most once and then the intersection is an unique common sub-path. Consequently Theorem 20 applies for any path.

Suppose that all arrival curves are affine, $\alpha_i(t) = \sigma_i + \rho_i t$, and $\alpha(t) = \sigma + \rho t$, and that all the nodes have rate-latency strict service curves, $\beta_v(t) = R_v(t - T_v)_+$. From the proof of Corollary 25, we know that the PMOO service curve offered by a path $\mathfrak{p}$ is a rate-latency function $\beta_{R,T}(t) = R(t - T)_+$ with

$$R = \min_{v \in \mathfrak{p}} \tilde{R}_v \text{ and } T = \sum_{i \in I} \frac{\sigma_i}{R} + \sum_{v \in \mathfrak{p}} T_v \left(1 + \sum_{i \in I_v} \frac{\rho_i}{R}\right)$$

where I_v is the set of indices of cross-traffic flows intersecting node v, $\tilde{R}_v \stackrel{\text{def}}{=} R_v - \sum_{i \in I_v} \rho_i$ and $I = \bigcup_{v \in \mathfrak{p}} I_v$.

Path optimizing the delay bound From the preceding formulas, if we choose path $\mathfrak{p}$, $D_{\max}(\alpha, \beta_{R,T}) = T + \sigma/R$ if $R \geq \rho$ (otherwise it is $+\infty$). In order to find a path from v_{in} to v_{out} which minimizes this bound on the worst-case end-to-end delay, one can first fix a node x and then consider only the paths from v_{in} to v_{out} such that x is on the path and $\tilde{R}_x$ is the minimum rate over all the service curves of the path. Such a choice means removing all the nodes v of the graph such that $\tilde{R}_v < \tilde{R}_x$ (one only has to mark the deleted nodes). The next key idea is to weight both nodes and arcs of the remaining graph G_x:

- For each node v, the weight of v is $T_v \left(1 + \sum_{i \in I_v} \frac{\rho_i}{R_x}\right)$.
- For each arc (v, w), the weight of (v, w) is $\sum_{i \in I_w \setminus I_v} \sigma_i$.

One can easily check that for a path $\mathfrak{p}$ of G_x from v_{in} to v_{out} via x the bound $D_{\max}(\alpha, \beta_{R,T})$ is equal to $\sigma/\tilde{R}_x$ plus the sum of the weights of all the nodes and the arcs of $\mathfrak{p}$. Then finding an optimal path can be done in $\mathcal{O}(|V| + |A|)$ time thanks to a classical shortest path algorithm for acyclic graphs [12] executed from v_{in} to x, and from x to v_{out}. It has to be performed for all possible choices of x, leading to an overall $\mathcal{O}(|V|(|V| + |A|))$ complexity.

Path optimizing the backlog bound The same scheme applies for the backlog bound $B_{\max}(\alpha, \beta_{R,T}) = \sigma + \rho T$ if $R \geq \rho$ (otherwise it is $+\infty$). After fixing node x as the candidate for minimum rate $\tilde{R}_x$, the same weights can be associated with the nodes and arcs of G_x. The same classical shortest path computations can be performed since one only has to minimize T over all the possible paths crossing x. The difference from the delay computations arises when computing the minimum cost for fixed x which is $\sigma + \rho T$ instead of $T + \sigma/\tilde{R}_x$. It may lead to different optimal paths when comparing the optimal costs over all the possible choices of x. The overall complexity remains $\mathcal{O}(|V|(|V| + |A|))$.

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

Proposition 29 *For an acyclic graph $G = (V, A)$ and a strongly acyclic cross-traffic, with affine arrival curves and rate-latency strict service curves, one can find in $\mathcal{O}(|V|(|V| + |A|))$ time a path minimizing the delay (resp. backlog) bound associated to the PMOO service curve of Theorem 20.*

5 Implementation work

Using the algorithmic framework of [8], a Network Calculus software is currently under development. The main Network Calculus operations have been implemented for a large class of piecewise affine functions. It includes the functions with a finite number of affine pieces, and thus the classical arrival and service curves of the literature. A first version can be downloaded at the COINC web site [11].

The new multi-dimensional operator, introduced in Theorem 20 and providing a service curve for paths in PMOO situations, 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 Fig. 14, which a strongly acyclic cross-traffic.

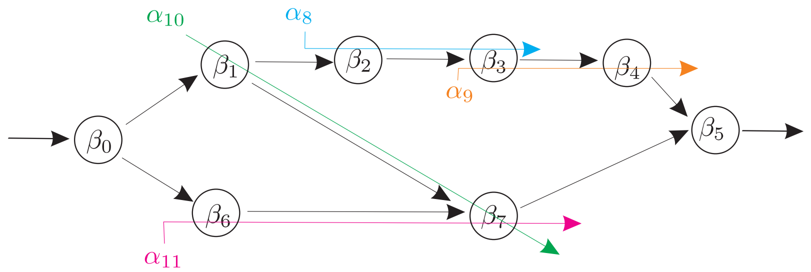

Fig. 14. An example of strongly acyclic cross-traffic.

The routing problem has been solved for a main flow with arrival curve α that enters the network at node β_0 and leaves at node 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 [6] 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 the algorithm presented in Corollary 25,
- 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):

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

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

The six curves are pictured on Figure 15 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

6 Conclusion

Network Calculus does lend itself well to routing problems where one wish to optimize end-to-end performance guarantees. Under some reasonable assumptions like independent cross-traffic and convex/concave shapes of the curves,

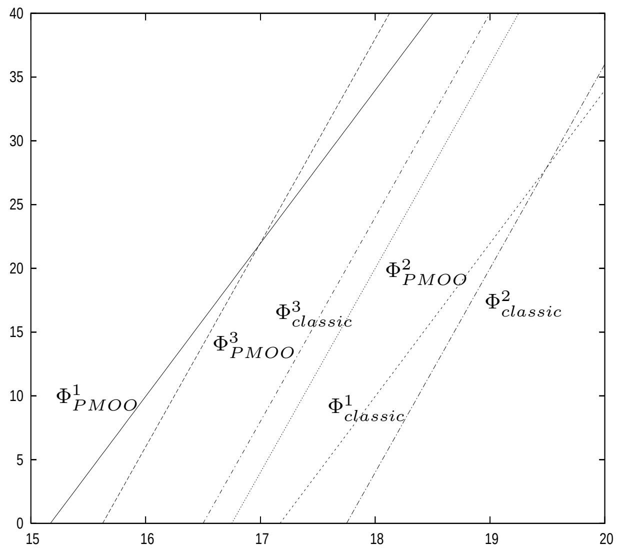

Fig. 15. Service curves of paths.

those problems are tractable and appear as variations of classical shortest path problems. The generalization of the formula of [38,39] quantifying the PMOO phenomenon has led us to introduce a new multi-dimensional operator. It suggests that the Network Calculus theory may well accommodate new operators and new formulas in order to refine the analyses.

As specified in Section 3.1, routing has been optimized with regard to upper bounds on delay or backlog which are provided by Network Calculus formulas. In order to say that the optimal path we compute is actually offering the smallest worst-case delay or backlog, one must wonder whether the formulas we use are tight ? Is there always a trajectory of the system where the worst-case delay reaches the bound we used ? The answer requires some additional work. In the absence of cross-traffic and with service curves which are not assumed strict, it seems that $D_{\max}(\alpha, \beta_1 * \cdots * \beta_p)$ (resp. $B_{\max}(\alpha, \beta_1 * \cdots * \beta_p)$) are tight bounds on the worst-case delay (resp. backlog) over the path. An argument would be a proof that Lemma 4 provides the "best" service curve for the tandem of nodes, given the initial service curves which are not supposed strict. However, in the case of independent cross-traffic or when PMOO is taken into account, where initial service curves are assumed strict, the tightness of the formulas we use is more questionable as suggested by the analysis of the toy example in Section 4.3. Those questions about tightness are definitely a major issue of Network Calculus.

Among other problems related to our work, one might wish to tackle the same routing problem with particular scheduling policies at the nodes, *e.g.* assuming that all nodes are FIFO. It requires a tight analysis of FIFO multiplexing in the Network Calculus framework. The reader is referred to [21–23] for the latest results and in particular to [22] for a nice survey of all the previous results.

Our work and in particular the new algorithms presented in this paper are new bricks in our project of implementing an efficient software tool based on Network Calculus, for the analysis and the control of complex networks [11].

Acknowledgements

The authors wish to thank the anonymous referees for their precious comments which have helped to improve both the presentation and the content of this paper.

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 Transactions 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] F. Baccelli, G. Cohen, G.Y. Olsder, and J.P. Quadrat. *Synchronization and linearity*. Wiley, 1992.

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

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

[7] A. Bouillard, B. Gaujal, S. Lagrange, and E. Thierry. Optimal routing for end-to-end guarantees: the price of multiplexing. In *Proceedings of Valuetools'07*, 2007.

[8] A. Bouillard and E. Thierry. An algorithmic toolbox for network calculus. *Discrete Event Dynamic Systems*, 18(1):3–49, 2008.

[9] C. S. Chang. Deterministic traffic specifications via projections under the min-plus algebra. In *Proceedings of INFOCOM'99*, pages 43–50, 1999.

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

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

[12] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms*. MIT Press, 2001.

[13] R. L. Cruz. A calculus for network delay, part i: Network elements in isolation. *IEEE Transactions on Information Theory*, 37(1):114–131, 1991.

[14] R. L. Cruz. A calculus for network delay, part ii: Network analysis. *IEEE Transactions on Information Theory*, 37(1):132–141, 1991.

[15] V. Firoiu, J.-Y. Le Boudec, D. Towsley, and Zhi-Li Zhang. Theories and models for internet quality of service. *Proceedings of the IEEE*, 90(9):1565–1591, 2002.

[16] J.-P. Georges, E. Rondeau, and T. Divoux. Evaluation of switched ethernet in an industrial context by using the network calculus. In *Proceedings of 4th IEEE Workshop on Factory Communication Systems*, pages 19–26, 2002.

[17] R. A. Guérin and A. Orda. Qos routing in networks with inaccurate information: theory and algorithms. *IEEE/ACM Transactions on Networking*, 7(3):350–364, 1999.

[18] Integrated services ietf working group. http://www.ietf.org/html.charters/OLD/intserv-charter.html.

[19] A. Juttner, B. Szviatovski, I. Mecs, and Z. Rajko. Lagrange relaxation based method for the qos routing problem. In *Proceedings of INFOCOM'01*, pages 859–868, 2001.

[20] F. Kuipers, P. Van Mieghem, T. Korkmaz, and M. Krunz. An overview of constraint-based path selection algorithms for qos routing. *IEEE Communications Magazine*, 40(12):50–55, 2002.

[21] L. Lenzini, L. Martorini, E. Mingozzi, and G. Stea. A novel approach to scalable cac for real-time traffic in sink-tree networks with aggregate scheduling. In *Proceedings of Valuetools'06*, 2006.

[22] L. Lenzini, L. Martorini, E. Mingozzi, and G. Stea. Tight end-to-end per-flow delay bounds in fifo multiplexing sink-tree networks. *Performance Evaluation*, 63(9-10):956–987, 2006.

[23] L. Lenzini, E. Mingozzi, and G. Stea. End-to-end delay bounds in fifo-multiplexing tandems. In *Proceedings of Valuetools'07*, 2007.

[24] D. H. Lorenz and A. Orda. Optimal partition of qos requirements on unicast paths and multicast trees. *IEEE/ACM Transactions on Networking*, 10(1):102–114, 2002.

[25] D. H. Lorenz, A. Orda, D. Raz, and Y. Shavitt. Efficient qos partition and routing of unicast and multicast. *IEEE/ACM Transactions on Networking*, 14(6):1336–1347, 2006.

[26] D. H. Lorenz and D. Raz. A simple efficient approximation scheme for the restricted shortest path problem. *Operations Research Letters*, 28(5):213–219, 2001.

[27] Y. Lucet. What shape is your conjugate ? a survey of computational convex analysis and its applications. Submitted, December 2007.

[28] Q. Ma and P. Steenkiste. Quality-of-service routing for trac with performance guarantees. In *Proceedings of 5th IFIP Workshop on Quality of Service*, 1997.

[29] D. Medhi and K. Ramasamy. *Network Routing: Algorithms, Protocols, and Architectures*. Morgan Kaufmann, 2007.

[30] S. Murthy and J. J. Garcia-Luna-Aceves. Congestion-oriented shortest multipath routing. In *Proceedings of INFOCOM'96*, 1996.

[31] A. Orda. Routing with end-to-end qos guarantees in broadband networks. *IEEE/ACM Transactions on Networking*, 7(3):365–374, 1999.

[32] N. Rao and S. G. Batsell. Algorithm for minimum end-to-end delay paths. *IEEE Communications Letters*, 1(5):152–154, 1997.

[33] N. Rao and S. G. Batsell. On routing algorithms with end-to-end delay guarantees. In *Proceedings of 7th Conf. on Computer Communications and Networks*, pages 162–167, 1998.

[34] N. Rao and S. G. Batsell. Qos routing via multiple paths using bandwidth reservation. In *Proceedings of INFOCOM'98*, pages 11–18, 1998.

[35] S. Recker. Service curve based routing subject to deterministic qos constraints. *Kluwer Telecommunications Systems*, 24(2-4):385–413, 2003.

[36] R. T. Rockfellar. *Convex Analysis*. Princeton University Press, 1996.

[37] J. B. Schmitt and U. Roedig. Sensor network calculusa framework for worst case analysis. In *Proceedings of 1st International Conference on Distributed Computing in Sensor Systems*, 2005.

[38] J. B. Schmitt and F. A. Zdarsky. The disco network calculator: a toolbox for worst case analysis. In *Proceedings of Valuetools'06*, 2006.

[39] 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, 2006.

[40] J. B. Schmitt, F. A. Zdarsky, and U. Roedig. Sensor network calculus with multiple sinks. In *Proceedings of IFIP Networking'2006*, 2006.

[41] L. Thiele, S. Chakraborty, M. Gries, A. Maxiaguine, and J. Greutert. Embedded software in network processors models and algorithms. In *Proceedings of Embedded Software Workshop EMSOFT'2001*, 2001.

[42] Z. Wand and J. Crowcroft. Quality-of-service routing for supporting multimedia applications. *IEEE Journal on Selected Areas in Communications*, 14(7):1228–1234, 1996.