Optimal routing for end-to-end guarantees: the price of multiplexing

Anne Bouillard
ENS Cachan / IRISA
Campus de Beaulieu
35000 Rennes, France
Anne.Bouillard@irisa.fr

Bruno Gaujal
INRIA / LIG
51, Av Jean Kuntzmann
38330 Montbonnot, France
Bruno.Gaujal@inria.fr

Sebastien Lagrange
INRIA / LISA
62, Av Notre Dame du Lac
49000 Angers, France
Sebastien.Lagrange@istia.univ-angers.fr

Eric Thierry
ENS Lyon / IXXI
46 Allée d' Italie
69007 Lyon, France
Eric.Thierry@ens-lyon.fr

ABSTRACT

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

General Terms

Network Calculus, (min,+) algebra, shortest path, multiplexing

1. INTRODUCTION

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

When the maximal delay over all packets in the flow is the performance index, fewer results are available in the litterature. Under static assumptions on the flows and the network ressources, optimal bandwidth allocation has been investigated in [7]. However, when the flows and the ressources have dynamic features, most focus is on simple systems such as single nodes where the issue becomes optimal scheduling.

Here we consider the problem of computing the route of a flow that provides the best *delay guarantee* $D_{\max}$ (no packet of the flow will ever spend more than $D_{\max}$ seconds in the system) or *backlog guarantee* $B_{\max}$ (the number of packets of the flow inside the network never tops $B_{\max}$), in the presence of cross-traffic. Network Calculus [6, 11] is a framework that allows us to formulate this problem as a mathematical program.

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

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

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

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

2. PERFORMANCES GUARANTEES

In this section, we recall the main definitions and the main properties of the Network Calculus functions and operations. More precise insights can be found in [6, 11].

2.1 Network Calculus functions

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

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

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

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

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

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

2.2 Arrival and service curves

Given a data flow traversing a system, let A be its arrival function (i.e. $A(t)$ is the number of packets that have arrived until time t). We say that α is an *arrival curve* for A (or that A is upper-constrained by α) if $\forall s, t \in \mathbb{R}_+$, $A(t+s) - A(s) \le \alpha(t)$. This means that the number of packets arriving between time s and $t+s$ is never larger than $\alpha(t)$. An important particular case of arrival curve is the affine functions: $\alpha(t) = \sigma + \rho t$. Then σ represents the maximal number of packets that can arrive simultaneously (the maximal burst) and ρ the maximal long-term rate of arrivals.

Consider D the departure function of the flow, defined similarly by the number $D(t)$ of packets that have left the system until time t. The system provides a (minimum) *service curve* β if $D \ge A*\beta$. Particular cases of service curves are the *peak rate* functions with rate r (the system can serve r packets per unit of time and $\beta(t) = rt$) and the *pure delay* service curves with delay d: $\beta(t) = 0$ if $t < d$ and $\beta(t) = +\infty$ otherwise. The combination of those two service curves gives a *rate-latency* function $\beta : t \mapsto R(t-T)_+$ where a_+ denotes $\max(a, 0)$. A *strict service curve* β is a service curve s.t. for all $t \in \mathbb{R}_+$, let u be the last instant before t when there is no packet in the system, then $D(t) \ge A(u) + \beta(t-u)$. This enforcement of the service curve notion is necessary to have refined bounds (*e.g.* positiveness of output service curves in Lemma 2 and Theorem 3). This condition will be often fullfilled in the paper, since we will mainly work with convex service curves which are always strict service curves [11].

2.3 Performance characteristics and bounds

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

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

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

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

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

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

Theorem 1 ([6, 11]). *Let A be the arrival function with an arrival curve α for a flow entering a system with service curve β. Let D be the departure function. Then,*

1. *D has an arrival curve $\alpha' = \alpha \oslash \beta$.*
2. *$b(t) \le B_{\max} = \sup\{\alpha(t) - \beta(t) \mid t \ge 0\} = \alpha \oslash \beta(0)$.*
3. *$d(t) \le D_{\max} = \inf\{d \ge 0 \mid \forall t \ge 0,\ \alpha(t) \le \beta(t+d)\}$ $= \sup\{d \ge 0 \mid (-\beta) \oslash (-\alpha)(d) \le 0\}$*

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

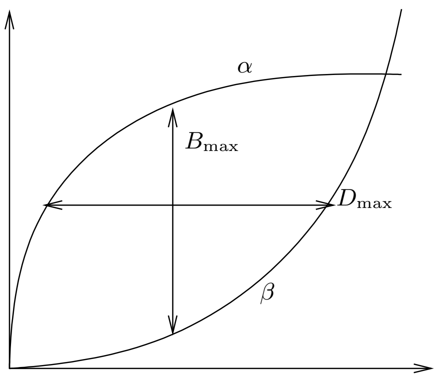

Figure 1: Guarantee bounds on backlog and delay.

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

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

LEMMA 1 ([6, 11]). *Consider two servers in tandem with respective service curves β_1 and β_2. Then the concatenation of the two servers offers a minimum service curve $\beta_1 * \beta_2$ to the flow.*

LEMMA 2 ([6, 11]). *Consider a server offering a strict service curve β and two flows entering that server, with respective arrival curves α_1 and α_2. Then a service curve for flow 1 is $\beta_1 = (\beta - \alpha_2)_+$.*

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

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

2.4 Representation of the functions

Our main objective is to find algorithms that enable to do some computations in Network Calculus. Then, we have to consider the implementability of the Network Calculus operations and functions. This question is addressed in [5] where it is shown that Network Calculus operations can be performed efficiently on a good class of functions with a finite representation: the piecewise affine functions that are ultimately pseudo-periodic. Here, we will make some further assumptions: our functions are continuous and piecewise affine with a finite number of segments. Such a function can be represented by a linked list of triples, each triple representing a segment, e.g. a triple (x, y, ρ) can represent the coordinates (x, y) of the beginning of the segment, and ρ its slope. The end of the segment is given by the next triple and the last triple represents the last segment which is of infinite length. Let f be such a continuous piecewise affine function, $|f|$ denotes the *size* of f, i.e. its number of segments. The complexity of the algorithms will strongly depend on the size of the functions.

In all the following, the functions we consider are always continuous and piecewise affine with a finite number of segments, even if not stated.

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

3. OPTIMAL ROUTING WITH INDEPENDENT CROSS-TRAFFIC

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

Here, every function is continuous and non-decreasing. Moreover, we consider that the service curves are convex and take into account the cross-traffic with blind multiplexing: if vertex v offers a service curve β_v^0 and the cross-traffic in vertex v has arrival curve α_v, then we will use $\beta_v = (\beta_v^0 - \alpha_v)_+$ as the service curve for the main flow, as shown in Lemma 2. Such a reduction is totally appropriate for independent cross-traffic. Once this is done, no mention of the cross-traffic is necessary anymore in this section.

We also suppose that the arrival curves are concave, which occurs in most classical cases. Note that given an arrival curve for a flow, its concave envelope remains an arrival curve, it provides concavity at the price of loosening the bounds. The service curves will be supposed convex. Moreover, the interferences with other cross-traffic flows with concave arrival curves, captured by Lemma 2, leave the service curves convex.

The general routing problem we consider in this section is:

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

In graph theory, one can mention two classical versions of optimal routing. With arcs and/or vertices weighted by numbers, the first one consists in finding the shortest path from one source to one destination, and the latter one is to find a path with maximum bottleneck capacity. Those two problems can be seen as special cases of our problem, when respectively the service curves β_v are all pure delays or are all peak rates.

We will present some special cases of our general problem. Let first state some general lemmas about computing the maximal backlog and delay.

Let f be a piecewise affine function. We define $r_f(t)$ as $\lim_{u>t, u\to t} \frac{f(u)-f(t)}{u-t}$, the slope of f at the right of t.

LEMMA 3. *Let A be an arrival flow with a concave continuous and piecewise affine arrival curve α and S be a system with a convex continuous and piecewise affine service curve β. Then, the maximal backlog for the system crossed by A is*

$$B_{\max} = \alpha(t_0) - \beta(t_0),$$

where $t_0 = \min\{t \in \mathbb{R}_+ \mid r_\alpha(t) \leq r_\beta(t)\}$.

Before proving that lemma, let first notice that t_0 is a point of change of slope of α or of β and can be computed in linear time in the number of slopes of the functions.

PROOF. As α is concave, continuous and piecewise affine and β is convex, continuous and piecewise affine, $\gamma = \beta - \alpha$ is also convex, continuous and piecewise affine. Then, γ has at most one local maximum (that can be an interval).

For every $t \leq t_0$, $r_\gamma(t) = r_\beta(t) - r_\alpha(t) \geq 0$, and then γ is non-decreasing on $[0, t_0]$. For every $t \geq t_0$, $r_\gamma(t) = r_\beta(t) - r_\alpha(t) \leq 0$ (by convexity of β and concavity of α), and then γ is non-increasing on $[t_0, +\infty[$. The maximum of γ is then obtained at t_0. □

LEMMA 4. *Let A be an arrival process with a concave continuous and piecewise affine arrival curve α and S be a system with a convex continuous and piecewise affine service curve β. Then, the maximal delay for packets of A through S is*

$$D_{\max} = \begin{cases} \beta^{-1}(\alpha(0)) & \text{if } r_\beta(\beta^{-1}(\alpha(0))) \geq r_\alpha(0) \\ \beta^{-1}(\alpha(t_0)) - t_0 & \text{otherwise,} \end{cases}$$

where $t_0 = \min\{t \mid r_\beta(\beta^{-1}(\alpha(t))) \geq r_\alpha(t)\}$.

PROOF. The horizontal distance between α and β at ordinate y is $\beta^{-1}(y) - \alpha^{-1}(y)$, with $\alpha^{-1}(y) = 0$ if $\alpha(0) < y$). As α and β are increasing, α^{-1} and β^{-1} are well-defined and are piecewise affine. Moreover, α^{-1} is convex, and continuous and β^{-1} is concave and continuous, so Lemma 3 can be applied to those two functions. □

Here again, to compute t_0, one only has to look among the points where α or β have a change of slope. That can be done in linear time.

3.1 Concave arrival curve / rate-latency service curves

We consider here concave arrival curves, and service curves that are the combination of a pure delay and a conservative link : $\forall v \in V$, $\beta_v(t) = r_v(t - T_v)_+$.

LEMMA 5. *Let $\beta : t \mapsto r(t-T)_+$ be a service curve for an arrival curve α (concave, piecewise-affine and continuous). Let $t_0 = \min\{t \mid r_\alpha(t) \leq r\}$ be the point where the slope of α becomes less than r. The maximum delay is $D_{\max} = T - \alpha(t_0)/r - t_0$ and the maximum backlog is $B_{\max} = \alpha(T)$ if $t_0 \leq T$ and $B_{\max} = \alpha(t_0) - \beta(t_0)$ otherwise.*

PROOF. It is a direct consequence of Lemmas 3 and 4. □

Consider a simple path $v_1, \ldots, v_\ell$ in the graph. The concatenation of the servers of those vertices is $\beta = \beta_{v_1} * \cdots * \beta_{v_\ell}$ and for all $t \in \mathbb{R}_+$, we have $\beta(t) = (\min_{i\in\{1,\ldots,\ell\}} r_{v_i})(t - \sum_{i=1}^{\ell} T_{v_i})$. Then the previous lemma can apply to the concatenation of servers. If $\min_{i\in\{1,\ldots,\ell\}} r_{v_i}$ is fixed and only the T_i's make β vary, then t_0 is also fixed and the maximum delay is given by $D_{\max} = \alpha(t_0)/r - t_0 + \sum_{i=1}^{\ell} T_{v_i}$. The computation of the maximum delay over simple paths can be reduced to the addition of delays T_{v_i} on the vertices of the paths, plus a constant. As for paths containing cycles, the maximal delay cannot be computed the same way, but is always larger than the maximum delay over the simple path obtained by removing all the cycles. Therefore, they can be discarded in our optimization problem. One can use a shortest path algorithm to compute the path that minimizes the maximal delay.

The basic idea is to consider one by one every vertex of the graph, and for each vertex v_0 compute the shortest path from x to y, with weights on vertex v being T_v or $+\infty$ if $r_v < r$. The weight of the path plus $\alpha(t_0)/r_{v_0} - t_0$ gives the maximal delay for a fixed service rate. The minimal worst-case delay is then the minimum of all the computed delays. Algorithm 1 uses the same principle, but automatically eliminates vertices that cannot lead to better performances on the delay. **Shortest-path**(G, x, y, w) computes the shortest path in G from x to y with weights on the vertices $w(v)$, $v \in V$. This can be done using Dijkstra'a algorithm, which can be implemented with complexity $O(|A| + |V| \log |V|)$ [9]. If the graph is acyclic, it can be done in $O(|A| + |V|)$. The overall complexity of Algorithm 1 is $O(|V|(|A| + |V| \log |V|))$.

Algorithm 1: Min worst case delay (rate-latency services)

Data: $G = (V, A)$ a directed graph, $(T(v))_{v\in V}$ weights on the vertices, $x, y \in V$.
Result: Path from x to y minimizing $D_{\max}$
begin
  $d_1 \leftarrow +\infty$;
  while *x and y are connected* **do**
    $p \leftarrow$ **Shortest-path**(G, x, y, T);
    $d_2 \leftarrow$ weight of p;
    $r \leftarrow$ the minimum service rate of the vertices of p ;
    Compute t_0;
    $d_1 \leftarrow \min(d_1, d_2 + \alpha(t_0)/r - t_0)$;
    $V' \leftarrow \{v \in V \mid r_v \leq r\}$;
    $G \leftarrow (V - V', A_{|V-V'})$;
  return d_1;
end

The same algorithm can be used for computing the best path regarding the backlog. The main difference is that the backlog is not exactly additive, depending on the comparison between t_0 and $\sum_{i=1}^{\ell} T_{v_i}$ on a path. But the smallest T is, the smallest the backlog is. One only has to replace the command $d_1 \leftarrow \min(d_1, d_2 + \alpha(t_0)/r - t_0)$ by Algorithm 2.

Algorithm 2:Min worst case backlog (rate-latency services)

begin
  if $t_0 \leq d_2$ **then** $d_3 \leftarrow \alpha(d_2)$ **else**
    $d_3 \leftarrow \alpha(t_0) - r(t_0 - D)$;
  $d_1 \leftarrow \min(d_1, d_3)$;
end

3.2 Affine arrival curve / convex service curves

We now consider the dual case, where the service curves are quite general (convex, piecewise affine and continuous) and the arrival curves are affine functions. We will see that it is easier to compute the maximal backlog, as a single shortest path computing is enough, but for the delay, we show that sub-paths may not be optimal in some cases, which makes the shortest path problem very tricky since up to our knowledge no shortest path algorithm exists in that case.

3.2.1 Minimizing the maximal backlog

PROPOSITION 1. *Let β be a non-decreasing, piecewise affine convex service curve for a server S and a flow entering S that has an affine arrival curve α, $\alpha(t) = \sigma + \rho t$. Let $t_0 = \min\{t \in \mathbb{R}_+ \mid \beta(t) \geq \rho\}$. Then, $B_{\max} = \sigma + \rho t_0 - f(t_0)$.*

PROOF. This is a direct consequence of Lemma 3. □

We now give the key theorem of the paragraph, that is the optimality of the sub-paths, regarding the backlog. More precisely, if there exist two paths from u to x, p_1 and p_2 and a path p_3 from x to v, then if p_1 offers a better guarantee than p_2 then p_1p_3 offers a better guarantee than p_2p_3.

THEOREM 2. *Let $\beta_1, \beta_2, \beta_3$ be three non-decreasing convex functions in $\mathcal{F}$ and $\alpha : t \mapsto \sigma + \rho t$. Set $b_i = \max_t[\sigma + \rho t - \beta_i(t)]$ and $b'_i = \max_t[\sigma + \rho t - \beta_i * \beta_3(t)]$, $i \in \{1, 2\}$. If $b_1 \leq b_2$, then $= b'_1 \leq b'_2$.*

PROOF. Let $t_i = \min\{t \in \mathbb{R}_+ \mid r_{\beta_i} \geq \rho\}$, $i \in \{1, 2, 3\}$. As β_1 and β_3 are convex, $\beta_1 * \beta_3$ is also convex, and it is a well-known fact that the convolution of such piecewise affine functions is the concatenation of the segments of the functions in the increasing order of the slopes, starting from $\beta_1 * \beta_3(0) = \beta_1(0) + \beta_3(0)$. Then, the backlog for the concatenation of β_1 and β_3 is obtained as

$$\begin{aligned} b'_1 &= \alpha(t_1 + t_3) - \beta_1 * \beta_3(t_1 + t_3) \\ &= \alpha(t_1 + t_3) - \beta_1(t_1) - \beta_3(t_3) \\ &= b_1 + \rho t_3 - \beta_3(t_3). \end{aligned}$$

The same holds for β_2 and β_3. Then $b'_1 - b'_2 = b_1 - b_2$. □

The proof of the theorem gives a method, shown in Algorithm 3, to minimize the maximal backlog of the system only computing a shortest path in a graph, with the adequate weighting.

Algorithm 3: Min worst case backlog (affine arrivals).

Data: A directed graph $G = (V, A)$, β_v, $v \in V$, the service curve for the servers for each vertex, $x, y \in V$, (σ, ρ) the arrival curve.
Result: Path from x to y minimizing the maximal backlog of the system
begin
  foreach $v \in V$ **do**
    $t_v \leftarrow \min\{t \mid r_{\beta_v}(t) \geq \rho\}$;
    $w(v) \leftarrow \rho t_v - \beta_v(t_v)$;
  $p \leftarrow$ **Shortest-path**(G, x, y, w);
  $B_{\max} \leftarrow w(p) + \sigma$;
end

3.2.2 Maximal delay of a packet

PROPOSITION 2. *Let β be a continuous, convex, piecewise affine service curve for a server S and a flow entering S that has an affine arrival curve α, $\alpha(t) = \sigma + \rho t$. Let $t_0 = \min\{t \mid r_\beta(t) \geq \rho\}$. Then*

$$d_{\max} = \begin{cases} \beta^{-1}(\sigma) & \text{if } t_0 \leq \sigma \\ t_0 - \frac{\beta(t_0) - \sigma}{\rho} & \text{if } t_0 \geq \sigma. \end{cases}$$

PROOF. This is a direct consequence of Lemma 4. □

The fact that there are two different possible values for the maximum delay prevents from adapting any simple shortest algorithm: the property that an optimal path is also locally optimal cannot be applied anymore. Indeed, let define $\beta_1 : t \mapsto \max(0, 2t - 10)$, $\beta_2 : t \mapsto \max(t/3, 2t - 20)$ and $\beta_3 : t \mapsto \max(0, t/3 - 2, 2x - 22)$, and an arrival flow $\alpha : t \mapsto 2 + 1/2t$. Those function are represented on Figure 2. The maximal delay for β_1 is $d_1 = 6$, and for β_2, $d_2 = 8$. Now, look at the delay for the $\beta_1 * \beta_3$ and $\beta_2 * \beta_3$-servers (d'_1 and d'_2 respectively). We have $d'_1 = 17$ and $d'_2 = 16$. Then the subpath optimality is violated in that case, jeopardizing the shortest path algorithms.

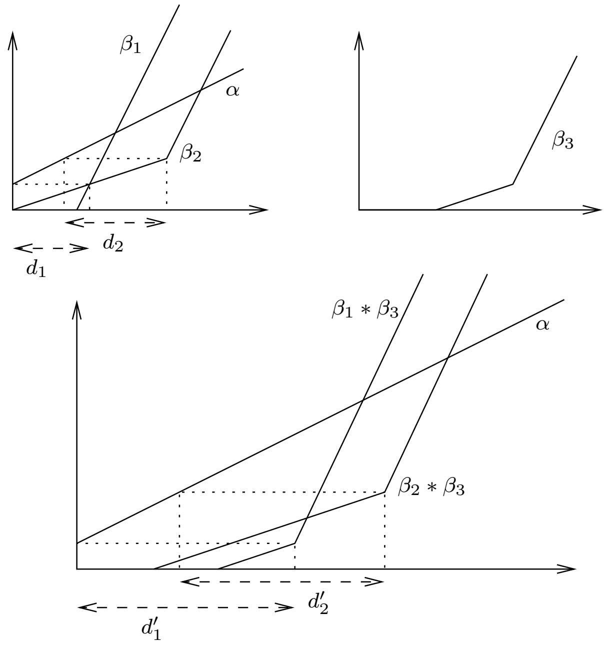

Figure 2: No subpath optimality property for delays.

4. OPTIMAL ROUTING WITH GENERAL CROSS-TRAFFIC

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

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

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

4.1 PMOO for one interfering flow

Let A_1 and A_2 be two arrival processes with respective arrival curves α_1 and α_2, that cross two concatenated servers with strict service curves β_1 and β_2. Let us compute the overall service curve for A_1 under blind multiplexing. Lemma 2 and Theorem 1 ensure that the service curve for A_1 is $(\beta_1 - \alpha_2)_+$ at server 1 and $(\beta_2 - \alpha_2 \oslash \beta_1)$ at servier 2. Lemma 1 then states that the service curve for the two concatenated servers is $\beta = (\beta_1 - \alpha_2)_+ * (\beta_2 - \alpha_2 \oslash \beta_1)_+$. On the other hand, if one sees the two servers as one server (their concatenation) and then compute the service for A_1 under blind multiplexing, then the service curve for A_1 is $\beta' = ((\beta_1 * \beta_2) - \alpha_2)_+$. We have $\forall t \in \mathbb{R}_+$,

$$\begin{aligned}
&(\beta_1 - \alpha_2)_+ * (\beta_2 - \alpha_2 \oslash \beta_1)_+ \\
&\leq (\beta_1 - \alpha_2)_+ * (\beta_2 - \alpha_2)_+ \\
&= \inf_{s \in [0,t]} (\beta_1 - \alpha_2)_+(s) + (\beta_2 - \alpha_2)_+(t-s) \\
&\leq \Big(\inf_{s \in [0,t]} (\beta_1(s) + \beta_2(t-s)) - \alpha_2(t) \Big)_+ \\
&\leq \Big((\beta_1 * \beta_2) - \alpha_2 \Big)_+ (t).
\end{aligned}$$

So $\beta' \geq \beta$ and the second service curve is better than the first one. This illustrates the PMOO phenomenon: considering the multiplexing only once with the concatenation of the servers gives better results. Things become more complex when there are several interfering flows. An example of overlapping flows is given in Figure 3, where PMOO cannot be analysed using only the simple convolution and multiplexing operations described in Lemmas 1 and 2.

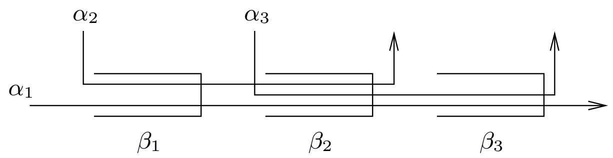

Figure 3: Overlapping flows.

4.2 PMOO with several interfering flows

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

Lemma 6. *With the notations and assumptions above,* $\forall t \in \mathbb{R}_+$, $\exists u_1, \dots, u_n \in \mathbb{R}_+$ *such that*

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

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

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

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

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

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

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

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

$$A_1^{(n)}(t) - A_1^{(n-1)}(t - u_n) \geq 0 \text{ and } \geq \beta_n(u_n) - \sum_{i \in B \cup C} \Big(A_i^{(n)}(t) - A_i^{(n-1)}(t - u_n) \Big), \quad (1)$$

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

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

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

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

Theorem 3. *With the same assumptions and notations as above, if for each* $i \in \{1, \dots, k\}$, α_i *is concave, then a service curve for* F_1 *of the servers* $S_1, \dots, S_n$ *under blind multiplexing is*

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

Proof. Take the formula of the previous lemma. By causality of the system, we have $\forall i \in \{1 \dots, k\}$,

$\forall j \in \{s_i, \dots, e_i\}$, $\forall t \in \mathbb{R}_+$ $A_i^{(j)}(t) \le A_i^{(s_i-1)}(t)$. Then,

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

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

and

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

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

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

EXAMPLE 2. *To illustrate the formula, consider the system of Figure 3. The service curve given by Theorem 3 is ϕ with*

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

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

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

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

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

As a consequence,

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

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

Unfortunately, this formula is not very tight as soon as the functions α_i are composed of many affine pieces with different values at time 0. Another method de compute ϕ has been suggested in [12] to deal with the system of Figure 3 when arrival curves are concave and service curves are convex: the idea is to decompose each α_i as a minimum of affine functions, then use Theorem 3 to compute a service curve of the path for each of these affine arrival curves and finally recompose ϕ by taking the maximum of all those service curve. But the main drawback of this method is that it leads to very long computations, as one has to compute the maximum of many piecewise affine functions. If one decomposes the arrival curves and the service curves as a minimum and maximum of affine functions, one has to compute at the end the maximum of $N = |\alpha_1| \cdots |\alpha_k|.|\beta_1| \cdots |\beta_n|$ affine functions. The complexity of this is at least in $O(N \log N)$, which becomes huge very fast as one increases the number of servers or interfering flows.

Theorem 3 applies for general arrival curves α_i and strict service curves β_i. In case all α_i are concave and all β_j are convex, there is another way to compute the service curve ϕ by taking advantage of the convexity and the concavity of the curves. It directly uses an algorithmic approach which is is detailed in the next section, and it outperforms the algorithm in [12].

4.3 Computation of the service curve of a path

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

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

LEMMA 7. *The function ϕ is convex, continuous and piecewise affine.*

PROOF SKETCH. For all $i \in I$, the function $f_i(\sum_{j \in J_i} u_j)$ is convex because it is a maximum of affine functions. Therefore, $\sum_{i \in I} f_i(\sum_{j \in J_i} u_j)$is also convex over the compact domain $\{u_1, \dots, u_n \ge 0,\ \sum_{j \in J} u_j = t\}$. The minimum over $\{u_1, \dots, u_n \ge 0,\ \sum_{j \in J} u_j = t\}$ is also convex because $\sum_{j \in J} u_j = t$ is a linear constraint over $\mathbb{R}^n$.

Secondly, the functions $f_i(\sum_{j \in J_i} u_j)$ are piecewise affine so that $\sum_{i \in I} f_i(\sum_{j \in J_i} u_j)$ is piecewise affine as well as $\min_{\substack{u_1, \dots, u_n \ge 0 \\ u_1+\dots+u_n = t}} \sum_{i \in I} f_i(\sum_{j \in J_i} u_j)$. $\square$

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

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

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

Let $t \ge a$. Suppose that $\phi(t) = F(u_1, \dots, u_n)$ with $u_1 < a$. We show that there exists another decomposition $\phi(t) = F(u'_1, \dots, u'_n)$ where $u'_1 = a$.

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

$$F(u_1,\dots,u_n)-F(v_1,\dots,v_n) = \sum_{i\in I} f_i(\sum_{j\in J_i} u_i)-f_i(\sum_{j\in J_i} v_j)$$

$$= \sum_{i\in I-I_1}\Big[f_i(\sum_{j\in J_i} u_j)-f_i(\sum_{j\in J_i} v_j)\Big]+\sum_{i\in I_1}\Big[f_i(\sum_{j\in J_i} u_j)-f_i(\sum_{j\in J_i} v_j)\Big]$$

$$= \sum_{i\in I-I_1}\Big[f_i(\sum_{j\in J_i} u_j) - f_i(\sum_{j\in J_i} v_i)\Big] + \sum_{i\in I_1}\Big[f_i(u_1) - f_i(a)\Big] +$$

$$\sum_{i\in I_1}\Big[(f_i(\sum_{j\in J_i} u_i) - f_i(u_1)) - (f_i(\sum_{j\in J_i} v_j) - f_i(v_1))\Big].$$

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

The equation above can be rewritten as

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

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

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

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

with $u'_1 = u_1 - a$ and $u'_j = u_j$ for $j \ge 2$. The functions f'_i are still convex, continuous and piecewise affine. So one can compute ϕ on an interval $[a, b]$ using the f'_i. Remark also that the total size of the functions f'_i is strictly less than that of the f_i, because a corresponds to a change of slope of a function (so the first segment of that function disappears in at least on of the $f'_i, i \in I_1$). The function ϕ can then be computed in finite time, repeating the computations above at most as many times as the total number of segments of the functions f_i.

Algorithm 4 gives the computation of ϕ. The functions are represented as described in Paragraph 2.4. Operator **Next**.f points on the next triple of f and **AddSegment** construct ϕ adding the last three parameters as the last segment of ϕ. Moreover $\phi.x$, $\phi.y$ and $\phi.\rho$ represent the triple of the last constructed segment. In the outside loop, ρ can be found in time n, ℓ_0 in time at most k. The inside loop has a constant execution time if the total length remains the same, and has a complexity at most n if the total size of the f_i's decreased by one. The overall complexity is then in $O((\sum_{i=1}^{k} |f_i|)(k + n))$.

Applying Algorithm 4 to the functions β_j and $-\alpha_i$ outputs a function ϕ, and then removing the negative part by taking ϕ_+ gives the computation of the service curve of Theorem 3.

Algorithm 4: Computation of ϕ.

Data: I, J two finite sets, $f_i, i \in I$ convex continuous piecewise affine functions, $J_i \subseteq J$, $I_j \subseteq I$ such that $i \in I_j \Leftrightarrow j \in J_i$.
Result: $\phi : t \mapsto \min_{(u_j)_{j\in J}\ge 0, \sum_{j\in J} u_j = t} \sum_{i\in I} f_i(\sum_{j\in J_i} u_j)$.

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

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

4.4 Adaptation of the algorithms to the PMOO

In this section, we consider only the most simple case: arrival curves are affine and service curves are rate-latency curves.

4.4.1 General acyclic graphs

Also, we consider an acyclic graph $G = (V, A)$. Each vertex v is a server with a minimal service curve $\beta_v : t \mapsto R_v(t - T_v)_+$. In that graph, there are some cross-traffic flows F_i, $i \in \{1,\dots,k\} = I$ which respectively follow paths p_i and have respective arrival curves α_i. Consider that the main flow F_0 follows a path p of the graph. Since the network is acyclic, the vertices are sorted according to the topological order. If vertex $v \in p_i$, one computes the service curve $\phi^i_v(t)$ of the cross-traffic F_i just after v. Its arrival curve in the following node will then be $\alpha_i \oslash \phi^i_v$. Using Theorem 3, the service curves $\phi^i_v(t)$ are defined by the following inductive formula, where $v_1,\dots v_m$ are the vertices over path p_i up to node v (included) and $F_{h_1},\dots,F_{h_r}$ are all the flows interfering with flow F_i up to vertex v. $v_h(\ell)$ is the vertex on p_h just before flows F_i and F_h meet for the ℓ^{th} time (they meet w time in total) and $p(\ell)$ is the ℓ^{th} commun sub-path for the flows F_i and F_h after node $v_h(\ell)$:

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

Using these notations, the end to end service curve of the main flow over path p is $\phi^0_y(t)$.

Note that all the functions ϕ_v^i depend on p, the path chosen for the main flow from x to y.

Now the algorithm for finding the best route is:
for all path p from x to y compute ϕ_y^0 and choose the best one w.r.t. the performance measure (delay or backlog).

4.4.2 Strongly acyclic graphs

The main drawback of the previous approach is that one needs to compute the service curve ϕ for each path p from x to y (which can be exponentially many). This is because the arrival curve of the cross-traffic in each node depends on the path p chosen for the main flow. Here, we characterize networks for which this is not the case and where the arrival curve of cross traffic at each node can be precomputed by the classic use of the deconvolution formula as explained e.g. in [11]. Moreover the computation of the best route for the main flow can be carried without computing the service curve on each path.

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

Assuming that all α_i are affine with rate-latency service curves on each node, and using the formula of ϕ in Theorem 3, the service curve on a path $p = v_1, \dots, v_m$ can be written as

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

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

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

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

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

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

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

The maximal delay and backlog strongly depend on R, and if R is fixed, the same kind of algorithm as Algorithm 1 can be applied. In order to compute the maximal delay, for fixed R, the weight of a vertex v is $T_v(1+\sum_{i\in I_v}\frac{\rho_i}{R})$ and the weight of an arc $e = (u, v)$ is $\sum_{i\in I_v - I_u}\sigma_i/R$. The maximal delay on a path is the sum of the weights on that path plus σ/R.

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

Following the scheme of Algorithm 1, in the worst case, one has to execute one shortest path algorithm per vertex (it covers all the possible rates R). All this is summarized in the following proposition.

PROPOSITION 3. *For a strongly acyclic network (V, A), with affine arrival curves and rate-latency service curves, the optimal end-to-end service curve is a rate-latency service curve that can be computed using a shortest path algorithm with an overall complexity in $O(|V|(|V| + |A|))$.*

Like in the case without the interfering flows in Section 3.1, α could be any concave function. However, if the α_i were not affine, the computation of ϕ on a path would not be based on these simple convolutions, but on the more complex operations explained in Section 4.3, preventing us from using the preceding reduction to a shortest path problem.

4.5 Implementation work

Following the algorithmic framework of [5], a software for worst case performance evaluation with Network Calculus is currently under development. The main Network Calculus operations have been implemented for a large class of piecewise affine functions including the classical arrival and service curves of network calculus. A first version should be released soon for downloads (COINC Project [8]).

The new multi-dimensional operator, described in Algorithm 4 and corresponding to the service curve ϕ on a path with cross-traffic has been incorporated to the software. Routing algorithms presented in this paper have also been implemented.

These implementations are now tested on the example of Figure 4, which a strongly acyclic graph.

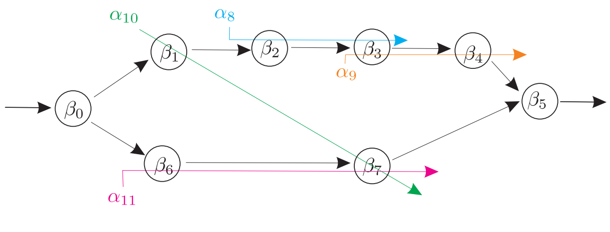

Figure 4: An example of a strongly acyclic network.

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

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

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

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

In order to check the results and to point out the benefits of taking into account PMOO compared to the classical approach [11] treating the cross-traffic as independent in each node, we have also computed for each path:
- the service curve ϕ^i_{PMOO} for the path i when taking into account PMOO, thanks to Algorithm 4,
- and the service curve $\phi^i_{classic}$ for the path i computed in the classical way.
These two service curves are (*e.g.* on Path 1):

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

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

The six curves are pictured on Figure 5 and their parameters are listed below. One can measure the gain of ϕ_{PMOO} over $\phi_{classic}$ and note that with $\phi_{classic}$, Path 3 would have been wrongly chosen as the best for the backlog.

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

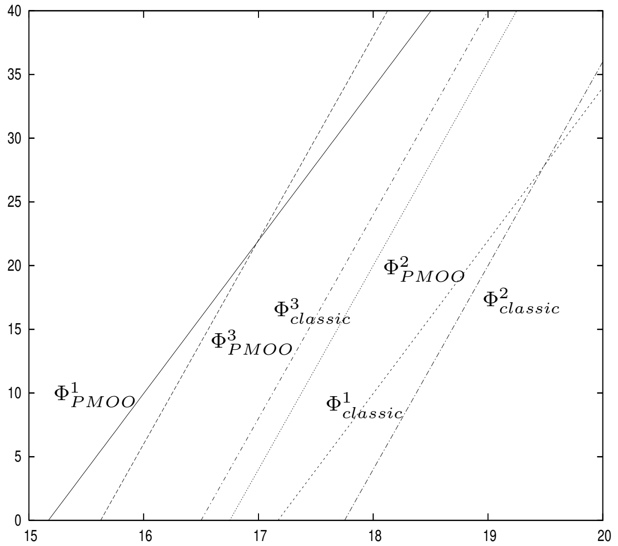

Figure 5: Service curves of paths.

5. CONCLUSION

Network Calculus does lend itself well to routing problems optimizing performance guarantees. We have shown how to solve them efficiently in some usual cases for an independent cross-traffic where extensions to more general arrival and service curves are under study. Then we have provided a new multi-dimensional operator quantifying the PMOO phenomenon for interfering flows, generalizing results of [12], and we have shown how to use it in routing problems for more general cross-traffic. Our results give insights into the benefit, but also the cost, of taking into account PMOO. These are some new steps in the construction of efficient software tools for the analysis and the control of complex networks, based on Network Calculus.

6. REFERENCES

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

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

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

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

[5] A. Bouillard and E. Thierry. An algorithmic toolbox for network calculus. Technical report, IRISA, 2007.

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

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

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

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

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

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

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

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