End-to-end performance guarantees for multipath flows

Anne Bouillard, Laurent Jouhet, Eric Thierry

End-to-end performance guarantees for multipath flows

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

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

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

ABSTRACT

When routing data across a network from one source to one destination, instead of following a fixed path, one can choose to spread data on several routes in order to use all potential ressources of the network. This issue has been studied for many models of networks with various objectives to optimize. In this paper we investigate how to route a flow across a network of servers with end-to-end performance guarantees in the framework of Network Calculus. We discuss stability issues (i.e. whether we can ensure that end-to-end delays are bounded) for arbitrary networks, and how to compute bounds on worst-case end-to-end delays and backlogs. The tightness issues are discussed on a small but challenging toy example.

1. INTRODUCTION

Finding good ways to route data across networks is a major issue in computer science and it has lead to a considerable amount of work. It has been studied with many variations depending on the modelling of the network, on the type of routing that is sought, and on the optimization objectives that are wished. It is not possible to provide here an extensive list of all those studies (see some reference books like [1, 13]). However as far as we know, the routing problem has not been yet much investigated in the framework of Network Calculus.

Network Calculus can be presented as a deterministic queuing theory based on the $(\min, +)$ semi-ring, and aimed at worst-case performance analysis in communication networks. From a mathematical point of view, it consists in combining curves which locally describe the shape of the traffic and the services, with $(\min, +)$ or $(\max, +)$ operations in order to predict the global behavior of the networks, and in particular end-to-end measures like delays or backlogs. Given a network modelled by a graph with servers on the arcs (or on the nodes) and a flow to route with its specifications (starting point, destination, arrival curve), the authors of [4, 5] raised the question of computing an "optimal" routing, i.e. which achieves the smallest bound on the worst case end-to-end delay or backlog. They provided efficient algorithms based on classical shortest path algorithms in the absence of cross-traffic. They also considered models with cross-traffic that may interfer with the routed flow, but such models are much more difficult to analyze in the framework of Network Calculus [15, 17] and in this context, optimal routing algorithms could be presented only for very specific configurations.

Rather than routing a flow on a single fixed path, one can hope that distributing data among several path will achieve better performance guarantees. In this paper we investigate this approach in the Network Calculus context. The network is modelled by a directed graph and the servers are located on the arcs (an easy transformation enables to translate such a model into a graph with servers at the nodes [5]). Each server is FIFO and is characterized by a minimum service curve which corresponds to a lower bound on its capacity to serve data (note that we will not require service curve to be strict [3]). The specifications of the flow we wish to route are its source in the network, its destination and its arrival curve, i.e. a function setting an upper bound on the amount of data that may arrive during any fixed duration. We work with a *fluid model* where data can be divided into arbitrarily small packets (we will use the term "bit" only as a divisible unit for data). Then we choose to focus on a routing scheme which distributes data along *fixed proportions*: at each node, the data that leaves the node is divided among the output arcs with fixed proportions (like 50%,20%,30% for three output arcs). Choosing a fluid model and a routing that strongly relies on this model may be questionable with regard to application. We do not discuss in this paper how to adapt the scheme to a discrete model (e.g. it is still possible to use a scheduling process which follows the fixed proportions, up to rounding effects) and whether it jeopardizes all the results. Fluid models are quite often good approximations of discrete models and in our case it appears as a good start to design multipath routing schemes and analyze them.

The Network Calculus notation and basic results are presented in Section 2, as well as our model of network and of routing scheme. Then we investigate in Section 3 the first main issue: we provide a necessary and sufficient condition in order to guarantees that one can route the flow across the network with fixed proportions while preserving its stability (i.e. ensuring that the amount of data in the net-

work remains bounded). This result is intimately linked to the max-flow/min-cut Ford-Fulkerson theorem of graph theory [1, 9]. In case stability can be achieved, we study more precisely the end-to-end performance measures by focusing one simple but challenging toy example. We show how one can obtain interesting bounds on the worst-case backlog in the whole network, but as discussed in Section 3.2 tightness is lost during the computation. Bounding the worst-case end-to-end delay for the routed flow is more involved due to possible permutations of the order of data during its propagation along several paths. Section 4 is devoted to this issue: we indicate how to compute a bound which is not tight and we discuss those tightness issues.

2. NETWORK CALCULUS FRAMEWORK AND THE NETWORK MODEL

2.1 Network Calculus operators

Network Calculus 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 modeled 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 $\mathbb{R}_+$. In this paper, we consider continuous time and data which can be arbitrarily divided. We call *bit* the data unit (divisible). Nevertheless, we do not require our functions to be continuous and authorize bursts of data. 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 [2] 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 has interesting algebraic properties (see [3] 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

Let A be an arrival process, that is, $A(t)$ is the amount of data that arrives until time t. We say that α is an *arrival curve* for A (or that A is upper-constrained by α) if $\forall s, t \in \mathbb{R}_+$, $A(t+s) - A(t) \le \alpha(s)$. This means that the amount of data arriving between time t and $t+s$ is never above $\alpha(s)$. An important particular case of arrival curve is the affine functions: $\alpha(t) = \sigma + \rho t$. Then σ represents the maximal burst that can arrive simultaneously and ρ the maximal average rate of arrivals.

Consider A an arrival process into a system and B the corresponding departure process. Given A, the system provides a (minimum) *service curve* β if $B \ge A * \beta$. Particular cases of service curves are the *peak rate* functions with rate r (the system can serve r packets per unit of time and $\beta(t) = rt$) and the *pure delay* service curves with delay δ: $\beta(t) = 0$ if $t < \delta$ and $\beta(t) = \infty$ otherwise. The combination of those two service curves gives a *rate-latency* function $\beta : t \mapsto R(t-T)_+$. In the examples of the paper, we will mainly study servers with rate-latency service curves, with piecewise affine arrival processes. The next lemma states that the convolution of such functions is simple to compute.

LEMMA 1. *Let f be a non-decreasing piecewise affine function and $g : t \mapsto \rho(t-T)_+$. Then, $f * g$ is $\ell * h$ where ℓ is a pure delay service curve with delay T and h the greatest continuous piecewise affine function h such that every slope of segment is less than ρ ($\forall t \in \mathbb{R}_+$, $h'(t^+) \le \rho$ and $h'(t^-) \le \rho$) and $h \le f$.*

PROOF. First assume that $T = 0$. From [6], the convolution of piecewise affine functions can be computed by calculating the minimum of the convolutions of each segment composing the function f by the line $t \mapsto \rho t$. For a segment $s : t \mapsto a + bt$, $\forall t \in [u, v]$, one have $s * g = g$ if $b \ge \rho$ and $s * g_{|[u,v]} = s$ and $\forall t \ge v$, $s * g(t) = a + b(v-u) + \rho(t-v)$. The minimum of such convolution gives the desired result for $T = 0$. Now, if $T > 0$, Let $f * g = f * g(. + T) * \ell$, so the result is straightforward. □

2.3 Performance characteristics and bounds

There are two performance characteristics that can be computed with network calculus, that are the backlog and the delay.

DEFINITION 1. *Let A be an arrival flow through a system and B be the corresponding departure process. Then,* backlog *of the flow at time t is*

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

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

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

Given an arrival curve and a service curve, it is possible to compute, with the network calculus operations, the maximal backlog (amount of data in the system) and delay. Moreover, one can also compute the arrival curve of the departure process.

THEOREM 1 ([7], THEOREM 2.3.4). *Let A be an arrival process with an arrival curve α entering a system with service curve β. Let denote by D the departure process. Then,*

1. *B has an arrival curve $\alpha' = \alpha \oslash \beta$.*
2. *$b(t) \le \alpha \oslash \beta(0)$.*
3. *$\delta(t) \le \sup\{\delta \ge 0 \mid \beta \oslash \alpha(\delta) \le 0\}$.*

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

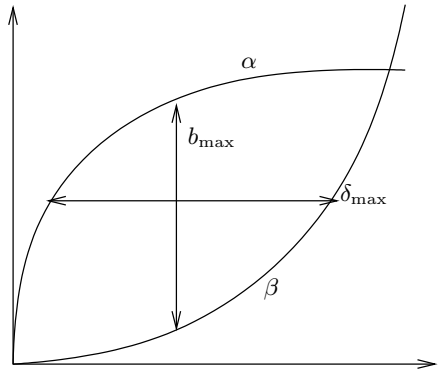

Figure 1: Guaranties bounds.

2.4 Model

In this paper, we study systems where servers are linked to form an acyclic network. There is a flow crossing that network from a source to a destination and that flow can split, so that data can take different paths. Formally, we will consider a connected directed acyclic network that is modeled by a directed acyclic graph $G = (N, A)$, with $|N|$ nodes and $|A|$ arcs. There are two distinguished nodes: a source node s and a destination d. Without loss of generality, as the network in acyclic, one can suppose that there is no input arc in s and no output arc from d and that every node is on a path from s to d. Each arc a is a server for which a non-negative service curve β_a is known. The service policy for each server is assumed to be FIFO (in a server, packets are served in the order of their arrival in this server).

The traffic in the network is as follows: a flow, with cumulative arrival curve A_s, arrives at node s and is transferred into the rest of the network, so that the total flow is directed toward d. The flow can be split and take several paths. When the flow is split, one assumes that it is done in a regular way. More precisely, one have the following notations and assumptions.

1. For every node $n \in N$, n^- is the set of input arcs of n and n^+ in the set of the output arcs of n.
2. For every arc $a \in A$, a_- is the origin node of a and a_+ is the destination node of a. These notations allow the use of multiple arcs between a pair of nodes.
3. For every output arc a from a node n, there exists $p_a \in [0, 1]$, with $\sum_{a \in n^+} p_a = 1$ such that the cumulative arrival process in a is $A_a = p_a A_n$.
4. For every arc a, the cumulative departure process from the server, with service curve β_a is denoted by B_a.
5. For every node n, the cumulative arrival process in n is $A_n = \sum_{a \in n^-} B_a$.

Moreover, for every arc $a \in A$, from the definition of a service curve, $B_a \geq A_a * \beta_a$.

Figure 2 illustrates the different relations between cumulative arrival curves for one server.

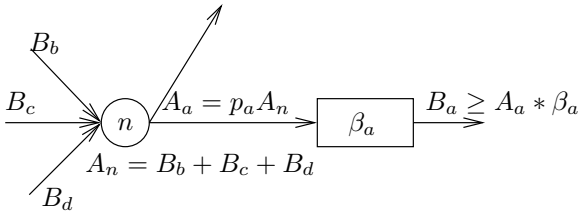

Figure 2: Flow model. At each node, the flows from the input arcs are aggregated and then split among the output arcs before being served by the server of the arc a.

In the following of the paper, we will need to propagate constraints and values from s to d. We then need to consider the nodes in a compatible order with the propagation. Then, each time we consider an order on the node, we will consider a topological order: $m \leq n$ if an only if there is a path from m to n. This order is not total, but any linearization of it will be suitable.

3. THE GLOBAL SYSTEM

In this section, we analyze the global system properties. The first important property for a system is its stability, and we will give a necessary and sufficient condition for the stability of such a system. The second point concerns the departure process. With the network calculus operators, we will see that one can give an upper bound for the departure process and for the backlog of the system. This can be of interest for studying properties that do not concern the order of arrival/departure of the data. But we will see that as far as the delay is concerned, this formula is of no use (and can be mistaking).

First, let us state a preliminary lemma concerning the departure process at a node.

LEMMA 2. *For every node $n \in N \setminus \{s\}$, the following inequation holds:*

$$A_n \geq \sum_{a \in n^-} \beta_a * (p_a A_{a_-}), \tag{1}$$

with the convention $\beta_{(m,n)} = 0$ and $p_{(m,n)} = 0$ if $(m, n) \notin A$.

PROOF. This is a direct consequence of the definition of our model: $\forall n \in N$,

$$\begin{aligned} A_n &= \sum_{a \in n^-} B_a \\ &\geq \sum_{a \in n^-} \beta_a * A_a \\ &\geq \sum_{a \in n^-} \beta_a * (p_a A_{a_-}). \end{aligned}$$

□

This lemma will be useful to study our system.

EXAMPLE 1. *Consider the network in Figure 3, with $s = 1$ and $d = 6$. the topological order on the nodes corresponds to the natural order of the numbering of the nodes. The following equations hold :*

$$A_6 = B_g + B_h \geq \beta_g * A_g + \beta_h * A_h = \beta_g * p_g A_4 + \beta_h * A_5$$
$$A_5 = B_c + B_f \geq \beta_c * A_c + \beta_f * A_f = \beta_c * p_c A_2 + \beta_f * p_f A_4$$
$$A_4 = B_d + B_e \geq \beta_d * A_d + \beta_e * A_e = \beta_d * p_d A_2 + \beta_e * A_3$$
$$A_3 = B_b \geq \beta_b * A_b = \beta_b * p_b A_1$$
$$A_2 = B_a \geq \beta_a * A_a = \beta_a * p_a A_1.$$

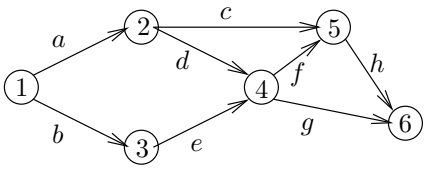

Figure 3: Example of a network.

3.1 Stability of the network

In this section, we give a necessary and sufficient condition for the stability of the system. The proof of this result is based on the Max Flow-Min Cut (or Ford-Fulkerson) theorem, that gives a method to compute a maximal flow in a network with constant capacities. Before this, we investigate the case when the routing proportions are fixed.

3.1.1 Stability for fixed routing proportions

Let α be an arrival curve for the cumulative arrival process A_s. As stated in [7, 3], one can suppose that α is sub-additive. Then, from Kingman theorem, $\lim_{t\to\infty} \alpha(t)/t$ exists. Let us denote by ρ this limit. This value represents the long-range maximal arrival rate for the flow.

We now suppose that for every $a \in A$, $\lim_{t\to\infty} \beta_a(t)/t$ exists and that this limit is r_a. This represents the long-range service rate of server a.

LEMMA 3. *Let α the arrival curve of a flow crossing a server with a service curve β. Pose $\lim_{t\to\infty} \alpha(t)/t = \rho$ and $\lim_{t\to\infty} \beta(t)/t = r$. In addition, suppose that there exist two constants c and c' such that $\alpha(t) \leq c + \rho t$ and $\beta(t) \geq c' + rt$ $\forall t \in \mathbb{R}_+$. Then the amount of data in the server is bounded and the output arrival curve $\lim_{t\to\infty} \alpha \oslash \beta(t)/t = \rho$ if and only if $\rho \leq r$.*

Note that since in our model the servers are FIFO, if the number of packets in the server is bounded, so is the maximal delay for a packet. Then, this system is stable.

PROOF. The quantity $\sup_{t\in\mathbb{R}_+} \alpha(t) - \beta(t)$ is finite if and only if $\rho \leq r$. Indeed $\forall t \in \mathbb{R}_+$, $\alpha(t) - \beta(t) \leq c - c' + (\rho - r)t \leq c - c'$.

But,

$$\begin{aligned} \alpha \oslash \beta(t) &= \sup_{s\geq 0} \alpha(t+s) - \beta(s) \\ &\leq \sup_{s\geq 0} \alpha(t) + \alpha(s) - \beta(s) \\ &\leq \alpha(t) + \alpha \oslash \beta(0). \end{aligned}$$

Then $\lim_{t\to\infty} \alpha \oslash \beta(t)/t \leq \rho$. Moreover, as the departure process B is such that $B(t) \leq \alpha \oslash \beta(t)$ and for $A = \alpha$ (this is always possible as α is sub-additive), if the server serves the data immediately (which is always possible as we only gave a lower bound for the service, $B = \alpha$). Then one has $\alpha \oslash \beta(t) \geq \alpha$ and $\lim_{t\to\infty} \alpha \oslash \beta(t)/t \geq \rho$. □

REMARK 1. *The assumption of the existence of c and c' is necessary: With $\alpha : t \mapsto \ln t$ and $\beta : t \mapsto 0$, one has $r = \rho$ but $\alpha - \beta$ is not bounded. This assumption becomes useless as soon as $\rho < r$.*

For given routing proportions in the network, one can deduce from Lemma 3 the long-range arrival rate in each node if every node preceding it is stable: under this assumption of stability, given a node $a \in A$, the long-range rate for A_a and B_a is the same. Let us denote it by ρ_a, and denote by ρ_n the long-range rate arriving at node n. Then, for every arc $a \in A$,

$$\rho_a = p_a \rho_{a_-} \quad \text{and} \quad \rho_n = \sum_{a\in n^-} \rho_a. \tag{2}$$

From Equation (2), one can easily deduce, as the network is acyclic, the proportion of the flow arriving at each server, by propagating the arrival rates in the server according to a topological order on the nodes.

EXAMPLE 2. *Consider again the network of Figure 3. The proportion of flow arriving at server f is:*

$$\begin{aligned} \rho_f = p_f \rho_4 = p_f(\rho_d + \rho_e) &= p_f(p_d \rho_2 + \rho_3) \\ &= p_f(p_d p_a \rho + p_b \rho) = p_f(p_d p_a + p_b)\rho. \end{aligned}$$

A direct consequence of Lemma 3 is:

PROPOSITION 1. *The network is stable if and only if for every arc $a \in A$, $\rho_a \leq r_a$.*

As explained it is easy to check, when the proportions p_a are given, whether the network is stable. Now, we address the problem of deciding whether there exist proportions such that the system can be made stable.

3.1.2 Stability of the network

In this section, we use a transformation of the network into a network with constant capacities in order to decide whether there exist routing proportions that make the network stable, and compute such proportions in case of stability.

A flow network with constant capacities is a 5-tuple (N, A, s, d, c) where (N, A) is a directed graph with two distinguished nodes s, the source, and d, the destination, and where $c : A \mapsto \mathbb{R}_+$ is a capacity function on the arcs. We first transform our network with functional capacities β_a, $a \in A$ into a flow network in the following way:

- the structure of the network remains the same $G = (N, A)$ as well as the two distinguished nodes s and d;
- for each arc $a \in A$, the capacity of a is $c(a) = r_a$.

A flow on a flow network is a function $\phi : A \to \mathbb{R}_+$ such that:

- $\forall a \in A$, $\phi(a) \leq c(a)$ (capacity constraint);
- $\forall n \in N \setminus \{s, d\}$, $\sum_{a \in n^-} \phi(a) = \sum_{a \in n^+} \phi(a)$ (conservation of the flow).

The value of the flow is $|\phi| = \sum_{a \in s^+} \phi(a)$. A cut $C = (E, N - E)$of the network is a partition of N in E and $N - E$ such that $s \in E$ and $d \in N - E$. The capacity of the cut $(E, N - E)$ is $\sum_{a \in A | a_- \in E, a_+ \in N-R} r_a$.

The Max Flow-Min Cut theorem (see [8] for more details) establishes a link between the maximum value of a flow and the capacity of the cuts of the network:

THEOREM 2 (MAX-FLOW/MIN-CUT). *Let (N, A, s, d, c) be a flow network. The maximum value of a flow is equal to the minimal capacity of cut.*

It is possible to effectively build a flow of maximal capacity, thanks to the Ford-Fulkerson algorithm. We will use this theorem to build routing proportions that make our network stable.

Suppose that in the flow network corresponding to our network, there exists a flow ϕ of value $|\phi| \geq \rho$. The routing proportions are defined as follow:

$$p_a = \frac{\phi(a)}{\sum_{b \in (a_-)^-} \phi(b)} \text{ if } a_- \neq s, \text{ and } p_a = \frac{\phi(a)}{|\phi|} \text{ if } a_- = s \tag{3}$$

LEMMA 4. *The network G with routing proportions defined above is stable.*

PROOF. We just need to prove that the routing proportions make the system stable for every server. Let us inspect every node and arc in the topological order.

For every output arc a of s, the long-range arrival rate is at most $p_a\rho$. From Equation (3),

$$p_a\rho = \frac{\phi(a)}{|\phi|}\rho \leq \phi(a) \leq c(a) = r_a.$$

Hence, those servers are stable. Now, suppose that for every server on each input arc a of node n, the inequality $\rho_a \leq \phi(a)$ holds. Let b be an output arc of n. Then, from Equations (2) and (3), the following inequality holds:

$$\rho_b = p_b\rho_n = p_b \sum_{a \in n^-} \rho_a \leq \frac{\phi(b)}{\sum_{a \in n^-} \phi(a)} \sum_{a \in n^-} \phi(a) = \phi(b).$$

Then $\rho_b \leq \phi(b) \leq c(b) = r_b$ and the server of arc b is stable. □

We just exhibit a sufficient condition for the stability. We now establish that this condition is also necessary.

LEMMA 5. *If for every flow ϕ, $|\phi| < \rho$, then for every routing proportions $(p_a)_{a \in A}$, there exists an unstable server.*

PROOF. We proceed by the contrapositive. Suppose that there are proportions p_a, $a \in A$ such that every server is stable (*i.e.* the system is stable). Consider the function $\phi : a \mapsto \rho_a$, where ρ_a is defined in Equation (3). It is enough to check that ϕ is a flow:

- $\forall a \in A$, $\phi(a) \leq r_a$ by construction;
- $\forall n \in N \setminus \{s, d\}$, $\sum_{a \in n^-} \phi(a) = \sum_{a \in n^-} \rho_a = \rho_n = \sum_{b \in n^+} p_b\rho_n = \sum_{b \in n^+} \rho_b$.

The function ϕ is a flow and has value $|\phi| = \sum_{a \in s^+} \phi(a) = \sum_{a \in s^+} p_a\rho = \rho$. □

Lemmas 4 and 5 yield the following theorem.

THEOREM 3. *There exists routing proportions that make the network stable if and only if the flow network with constant capacities on the arcs $(r_a)_{a \in A}$ has a capacity at least ρ.*

About networks with cycles. The assumption about acyclicity of the network can be relaxed for the latter theorem. Indeed, this assumption is stated only to insure the stability of the network for arbitrary networks, the case of network with cycles remaining open. But, when computing a maximal flow in an arbitrary network, the arcs crossed by the flow always form an acyclic network. Then, the stability of the network is ensured.

3.2 Global departure process

From Lemma 2, given a network, we can deduce a formula to compute A_d in function of the parameters of the network and A_s: as the network is acyclic, the inequalities of Equation (1) can be applied on every node in a topological order.

EXAMPLE 3. *Consider the network of Figure 3. Combining the inequalities of Example 1, one can express A_6 as:*

$$A_6 \geq \beta_g * p_g(\beta_d * p_d(\beta_a * p_a A_1) + \beta_e * (\beta_b * p_b A_1)) + \beta_h * [\beta_c * p_c(\beta_a * p_a A_1) + \beta_f * p_f(\beta_d * p_d(\beta_a * p_a A_1) + \beta_e * (\beta_b * p_b A_1))].$$

Note that this formula is tight: if the servers are all exact servers, that is, for every $a \in A$, $B_a = \beta_a * A_a$, then equalities are propagated instead of equalities.

It would be nice to deduce performance bounds from this global departure process. There are two main obstacles for this. First, the convolution is not distributive over the addition, so the formula cannot be developed and simplified. Second, the order of arrival of data is forgotten. The first obstacle can be overcome by loosing the tightness, as we will see in the next paragraph. The second one is an obstacle only for properties that are dependent of the data order. we first deal with properties that are independent of the ordering of data, namely, the backlog and the constraints for the departure process. Let set out a preliminary result.

LEMMA 6. *Let f, g and h be functions from $\mathbb{R}_+$ to $\mathbb{R}^+_{\min}$. For any $p \in [0,1]$, the following inequality holds.*

$$f * (g + h) \geq (pf) * g + ((1-p)f) * h. \quad (4)$$

PROOF. Let $t \in R_+$.

$$\begin{aligned} f * (g+h)(t) &= \min_{s\in[0,t]}[f(s) + g(t-s) + h(t-s)] \\ &= \min_{s\in[0,t]}[pf(s) + (1-p)f(s) \\ &\quad + g(t-s) + h(t-s)] \\ &\geq \min_{s\in[0,t]}[pf(s) + g(t-s)] \\ &\quad + \min_{s\in[0,t]}[(1-p)f(s) + h(t-s)] \\ &\geq (pf) * g(t) + ((1-p)f) * h(t). \end{aligned}$$

□

In the rest of the paper, we will study extensively the following toy example represented in Figure 4. We first show on this example how to compute an upper bound for the backlog and an arrival curve for the departure process. It should be clear from Lemma 6 that this can be easily generalized, but as this is not our main concern, we will not prove this here.

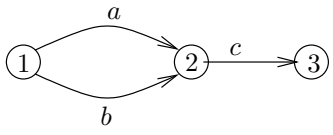

Figure 4: Toy example.

From Lemma 2, one gets the formula:

$$A_3 \geq \beta_c * [(\beta_a * pA_1) + (\beta_b * (1-p)A_1)].$$

From Lemma 6, one has:

$$A_3 \geq p\beta_c * \beta_a * pA_1 + (1-p)\beta_c * \beta_b * (1-p)A_1. \quad (5)$$

The maximum backlog $b_{\max}$ satisfies the inequality (with $q = 1-p$):

$$\begin{aligned} b_{\max} &\leq \sup_{t\in\mathbb{R}_+}[A_1 - p\beta_c * \beta_a * pA_1 - q\beta_c * \beta_b * qA_1](t) \\ &\leq \sup_{t\in\mathbb{R}_+}[pA_1(t) - p\beta_c * \beta_a * pA_1(t)] \\ &\quad + \sup_{t\in\mathbb{R}_+}[qA_1(t) - q\beta_c * \beta_b * qA_1(t)] \\ &\leq p\alpha \oslash p\beta_c * \beta_a(0) + (1-p)\alpha \oslash q\beta_c * \beta_b(0). \end{aligned}$$

The arrival curve for the departure process is:

$$\begin{aligned} A_3(t) - A_3(s) &\leq A_1(t) - p\beta_c * \beta_a * pA_1(s) - q\beta_c * \beta_b * qA_1(s) \\ &\leq p\alpha \oslash (p\beta_c * \beta_a) + q\alpha \oslash (q\beta_c * \beta_b). \end{aligned}$$

Note that the tightness is lost when the formula is developed in Lemma 6. The loss of the tightness can be explained differently: the possible bursts in the arrival processes arriving from server a and server b into server c are dependent from each other, as they split at node 1. Then, the worst-cases in servers a and b may not always happen at the same time, whereas the upper bound is computed as if they were synchronized.

3.3 Global maximum delay?

The third performance parameter we are interested in is the maximum delay. Here, the global method we used until now cannot be applied. Let first give an example, based on our toy example, where the delay cannot be computed with the formula.

From the usual Network Calculus formula, the delay can be computed as the smallest δ such that $\forall t \in \mathbb{R}_+$, $A_3(t+\delta) \geq A_1(t)$.

$$\begin{aligned} &\forall t \in \mathbb{R}_+,\ A_3(t+\delta) - A_1(t) \geq 0 \Leftarrow \\ &\forall t \in \mathbb{R}_+,\ p\beta_c * \beta_a * pA_1(t+\delta) + q\beta_c * \beta_b * qA_1(t+\delta) - A_1(t) \geq 0 \Leftarrow \\ &\quad \forall t \in \mathbb{R}_+,\ p\beta_c * \beta_a * pA_1(t+\delta) - pA_1(t) \geq 0 \text{ and} \\ &\qquad \forall t \in \mathbb{R}_+,\ q\beta_c * \beta_b * qA_1(t+\delta) - qA_1(t) \geq 0. \end{aligned}$$

Let δ_1 (resp. δ_2) be the maximal delay for an arrival process $p\alpha$ (resp. $q\alpha$) constrained in a server with a service curve $p\beta_c * \beta_a$ (resp. $q\beta_c * \beta_b * qA_1$). Then one can choose $\delta = \max(\delta_1, \delta_2)$. The delay obtained is a worst-case average delay: as the data can follow different paths, the FIFO order in the system is lost. Then, there could exist some data that have a delay greater than the one computed, as shown in the following example.

EXAMPLE 4. *Suppose that $\alpha : 0 \mapsto 0$; $t \mapsto 2\sigma + 2\rho t$, $p_a = 1/2$, $\beta_a = \beta_b : t \mapsto R(t-T)_+$ and $\beta_c : t \mapsto 2R_c(t-T_c)_+$.*

Then, the delay computed with this method is $\delta = T + T_c + \frac{\sigma}{\min(R_c, R)}$. *Now, we build an admissible evolution for the system with some data having delays greater than* δ.

The arrival process is exactly α: $A_1(0) = 0$ *and otherwise* $A_1(t) = 2\sigma + 2\rho t$. *Server a and c are exact servers. Server b is an exact server in the interval* $[0, T]$ *and then is an infinite server.*

The output process of server a is $B_a(t) = A_1/2 * \beta_a(t)$. *More precisely, it is a continuous piecewise-affine function with three different slopes:* $B_a(0) = 0$, B_a *has slope 0 on the interval* $[0, T]$, *slope* R *on* $[T, T + \sigma/(R - \rho)]$ *and slope* ρ *on* $[T + \sigma/(R - \rho), \infty[$. *The last bit of the burst at time 0 exits server a at time* $T + \sigma/R$. *The output process of server b is* $\alpha(t - T)$ *if* $t > T$ *and 0 otherwise. At time* $T + \sigma/R$, $\sigma(R + \rho)/R$ *bits have been served. In server c, the arrival process until the arrival of the last bit of the initial burst is* $A_2(t) = 0, \forall t \in [0, T]$, $A_2(t) = \sigma + (R+\rho)(t-T), \forall t \in]T, T + \sigma/R]$. *Then,* $A_3(t) = A_2 * \beta_c(t)$. *The last bit of the burst is served at time* $T' = \beta^{-1}(2\sigma + (R + \rho)/R)$. *Computations show that* $T' = T + T_c + (2\sigma + (R + \rho)/R)/2R_c > \delta$. *The computations can be followed on Figure 5*

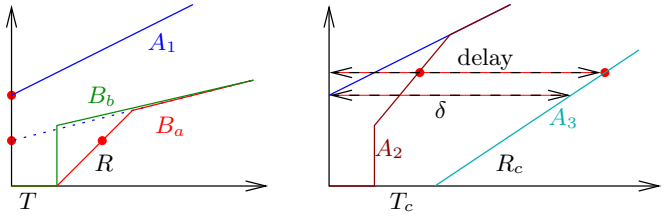

Figure 5: The worst-case delay cannot be computed with the global formula. The delay of the bit of data marked by a dot is greater than δ.

4. DELAY OF DATA IN THE SYSTEM

As seen in the previous paragraph, the maximum delay cannot be easily bounded by a global function describing the system. The main goal of this section is to give means for computing upper bounds of the worst-case delay. But before doing this, we need to precise the notion of delay we consider. Indeed, as the model we study is fluid and the traffic is split, the delay cannot be defined in terms of packets and by the difference between the departure and arrival time.

4.1 Maximum delay in the network

For a single FIFO server, with a single arrival process A and a service curve β, the departure process is $B \geq A * \beta$. Since B in non-decreasing, one can define its pseudo-inverse $B^{-1} : x \mapsto \sup\{t \in \mathbb{R}_+ \mid B(t) \leq x\}$. The maximal delay of a bit that arrives at time t is $B^{-1}(A(t)) - t$. Then, the maximal delay is

$$\delta_{A,B} = \max_{t \in \mathbb{R}_+} B^{-1}(A(t)) - t = \max_{x \in \mathbb{R}_+} B^{-1}(x) - A^{-1}(x).$$

The maximal delay, if the arrival process is α-upper constrained for the server is then

$$\delta = \max_{A \ \alpha\text{-uc}} \max_{B \geq A * \beta} \delta_{A,B}.$$

This formula is proved to coincide with the well-known formula given in Theorem 1.

This formula can be easily extended to servers in tandems when only considering one flow that follows only one path: assuming that every bit of data follows the same path, that the servers are FIFO, and (implicitly) that there is no overtaking of bits in the links, the data exit the system in the same order that they entered it. In our model, this is not the case anymore.

Now, consider a FIFO server with cross-traffic: the server has a service curve β and there are two arrival processes, A and A'. The two flows merge in the server so that the departure process is the departure process of the aggregate flow $A + A'$. The departure process is then B such that $B \geq (A + A') * \beta$.For data arriving at time t in the process A, the quantity of data that arrived between 0 and t is $A(t) + A'(t)$. Those data will leave at time t' such that $B(t') = A(t) + A'(t)$. Then, the delay is $B^{-1}(A(t) + A'(t)) - t$ and the maximum delay is:

$$\delta = \sup_{A, A', B} \sup_{t \in \mathbb{R}_+} B^{-1}(A(t) + A'(t)) - t.$$

Note that there can be conditions on the possible A and A' depending on the topology of the system (for example, they can depend from each other or not).

There is no well-known formula for this type of server and this case may be difficult to solve with no further assumption. A simple case that can be computed is when the two arrival processes A and A' are independent and respectively α and α' upper-constrained. If α and α' are affine, and β is rate-latency, then $\delta = T + (\sigma + \sigma')/R$.

That notion of delay can be generalized to our network model: for the different servers, arrival and departure processes can be computed with the formulas describing the model. Then, for any time t, the delay for the data arrived until time t can be computed node per node according to the following algorithm:

1. $t_s = t$; Take nodes n in a topological order;
2. if $n \neq s$, then $t_n = \max_{a \in n^-} t_a$;
3. $\forall a \in n_+$, $t_a = B_a^{-1}(A_a(t_a))$;
4. return $t_d - t_s$.

The value $t_d - t_s$ returned is the delay for given arrival and departure processes. The maximal delay is the the maximum delay for all t, and all possible arrival and departure processes.

It seems from this definition that exact bounds may be very difficult to obtain, as the optimization has to be taken over an infinite range of functions. We next give an upper bound for the delay.

4.2 Upper bound for the maximal delay

Here again, we use the acyclicity of the network and the topological order on the nodes. For each server a, one can compute an upper bound for the delay for each bit of data entering the server and an arrival curve for the departure process, in function of $\alpha_{a\_}$, an arrival curve for $A_{a\_}$, and of β_a, according to Theorem 1.

1. $\alpha_s = \alpha$;
2. $\forall m \in N$, $\forall a \in m^+$, $\alpha_a = p_a \alpha_m$;
3. $\forall a \in A$, $\delta_a = \sup\{\delta \geq 0 \mid \beta \oslash \alpha(\delta) \leq 0\}$... and $\alpha'_a = \alpha_a \oslash \beta_a$;
4. $\forall n \in N - \{s\}$, $\alpha_n = \sum_{a \in n^-} \alpha'_a$.

Any bit of data that follows a path $p = (a_1, \ldots, a_k)$ will have a delay less than $\sum_{i=1}^{k} \delta_{a_i}$. Then an upper bound on the delay corresponds to the length of a longest path from s to d in the graph G where the length of each arc a is δ_a. As the graph is acyclic, this length can be computed in linear time (in the size of the graph).

Example 5. *To illustrate this method, consider the toy-example in Figure 4 The respective worst-case delays δ_a and δ_b for servers a and b can be easily computed from α, p, β_1 and β_2. An arrival curve for the arrival process in server c is $\alpha' = p\alpha \oslash \beta_a + (1-p)\alpha \oslash \beta_b$ and the worst-case delay δ_c for such an arrival curve in β_c can also be easily computed. An upper bound for the delay is then* $\max(\delta_a, \delta_b) + \delta_c$.

If α is affine and the service curves are rate-latency (with the same notations as above), we get:

$$\begin{aligned} \delta_a &= T_a + \frac{p\sigma}{R_a} \\ \delta_b &= T_b + \frac{(1-p)\sigma}{R_b} \\ \alpha'(t) &= \sigma + T_a p\rho + T_b(1-p)\rho + \rho t \\ \delta_c &= T_c + \frac{\sigma + T_a p\rho + T_b(1-p)\rho}{R_c}. \end{aligned}$$

This method is similar to per-node analysis discussed in [14, 5]. It is well-known that this method is not optimal and that the delay can be arbitrary smaller than the bound computed. However, in absence of general means to compute an exact bound, it is shown that it is not worse than any other method : in [14], there are examples such that the bound computed this way is better than a bound computed by the other tractable approximation methods known up to now.

4.3 A tight bound?

An other way to compute the delay of data when the arrival and departure processes are known is to observe the possible paths for a bit of data. The delay on a path corresponds to the delay of the bit of data that follows that path. Then, as our model is fluid, one has to take the maximum on every possible path of the delays of that bit of data.

In this section, we try to give an exact bound for our toy example. The idea here is to compute worst-case trajectories of the system for each of the two paths. Let us take any bit of data arriving at time t directed to server a and see what is the latest time it can be served. We call this bit the *bit under observation*, or b.u.o., similarly to what is done in [14]. Here, the main difference is that the trajectories are far more difficult to find, because of the dependence between the arrival flows in servers a and b. Moreover, we do not assume the servers to have strict service curves.

We have the following monotony properties for single servers.

Proposition 2. *Consider a FIFO server with a service curve β, two different possible arrival process A and A' and two possible cross-traffic (that cross the server), C and C'. Pose $B_1 = (A + C) * \beta$, $B_2 = (A' + C) * \beta$ and $B_3 = (A + C') * \beta$.*

1. *The delay of bits of A when C is fixed is maximized if the server is an exact server.*
2. *If the server is exact, if C is fixed and if $A \leq A'$, the delay for transferring the x first bits of data of the process A is greater than the one to transfer the same amount of data of A':*
$$B_1^{-1}(A + C)(A^{-1}(x)) \geq B_2^{-1}(A' + C)(A'^{-1}(x)).$$
3. *If the server is exact, if A is fixed and if $C \leq C'$ and $C(t) = C(t')$, then the delay of a bit of data of A arriving at date t is greater with cross-traffic C than with C':*
$$B_1^{-1}((A + C)(t)) \geq B_3^{-1}((A + C')(t)).$$

Proof. 1. Trivial

2. Since $A \leq A'$, $B_1 \leq B_2$, $B_1^{-1} \geq B_2^{-1}$ and $A^{-1}(x) \geq A'^{-1}(x)$.
3. Since $C \leq C'$, $B_1 \leq B_3$ and $B_1^{-1} \geq B_2^{-1}$. Since $C(t) = C'(t)$, $(A + C)(t) = (A' + C)(t)$.

□

Proposition 2 has the following consequences:

- server c should be exact (1.);
- server a should also be exact (2.);
- server b should be exact while the b.u.o. is in server a and become an infinite server at the time the b.u.o. exits from server a (3.).

Now, the service policies are fixed for every server. The only other parameter to fix is the arrival process, and find the one that enables the worst-case delay. Against the intuition, and to what the worst-case is in tandem networks, we will not

take $A_1 = \alpha$, the maximal arrival curve, but we will build a little more bursty process.

Consider that the b.u.o. arrives in the system at time t_0. The bit of interest is divided in two, one part of it going to server a and the other part to server b. The delay of that bit will be given by the path where it departs the latest among server a and b. Until the b.u.o. exits servers a and b, both servers are exact servers, as explained before. Without loss of generality, suppose that the b.u.o. last exits from a. Then, at time it exits from a, server b is an infinite server, and all the data that where in the buffer of b arrive just before the b.u.o.

First, we show that the arrival process before the arrival of the b.u.o. is of the form $A_1(t) = \rho(t - t_{-1})_+$ if $0 \leq t < t_0$, and $A_1(t_0) = x + \rho t_0$, with $x \leq \sigma$: let $u_0 \in [0, t_0]$ such that $\rho u_0 - A_1(u_0) = \inf_{t\in[0,t_0]}(\rho t - A_1(t))$ (or $\rho u_0 - A_1(u_0^+)$ or $\rho u_0 - A_1(u_0^-)$ if A_1 is not continuous at u_0. The (σ, ρ) constraint on A_1 is given at that point and the arrival process A such that

- $A(u_0) = \min(A_1(u_0), A_1(u_0^+), A_1(u_0^-))$,
- $A(t_0) = A_1(t_0)$,
- A is affine of slope ρ on $[u_0 - A(u_0)/\rho, t_0[$

satisfies exactly the same constraint. From Proposition 2.2 and .3, the delay of the b.u.o. is maximized when the arrival curve (and cross-traffic) is minimized. This is the case of the function described above. Now, remark that we could arbitrary make data arrive before t_0, so that those data do not interfere with the original process. It is enough to make those data arrive long before t_{-1}, satisfying the same (σ, ρ) constraint given at point u_0: let a period of time more than the delay bound obtained in Section 4.2 with no arrival. As a consequence, one can suppose $t_0 - t_{-1}$ is big enough for the further calculations without adding any constraint on the flow.

The b.u.o exits from server a at time $t_0 + T_a + x\sigma/R_a$ (where $x = A(t_0) - A(t_0^-)$) and from server b at time $t_0 + T_b + (1-p)x/R_b$. Suppose without loss of generality that $T_a + px/R_a \geq T_b + (1-p)x/R_b$.

Between time t_0 and t_1, one has to find the arrival process that will maximize the delay of the b.u.o. in server c. The maximal quantity of data that can arrive in server b between t_0 and t_1 is $(1-p)(\rho(T_a + px/R_a) + \sigma - x)$. If less than this quantity arrives, then the delay will not be maximal. From 3., the cross-traffic in b must arrive late and must satisfy the constrained on the arrival process. The smallest arrival process that meets the constraints is: $A_1(t) = A_1(t_0)$ if $t \leq t_2$, $A(t) = A(t_2) + \rho(t - t_2)$ if $t \in [t_2, t_1[$ and $A(t_1) = A(t_1^-) + \min(\sigma, \rho(T_a + px/R_a) + \sigma - x)$, with $t_2 = t_1 - (\frac{\rho(T_a+px/R_a)-x}{\rho})_+$. Figure 6 explicits the two different cases that can occur, if $t_1 = t_2$ or not.

For any fixed parameter, one has exhibited the arrival process that maximizes the delay. In order to compute the delay, one just need to apply Lemma 1 for the parts of the process obtained with exact servers (the only other service policy used in the infinite server). Figure 7 shows graphically such a computation.

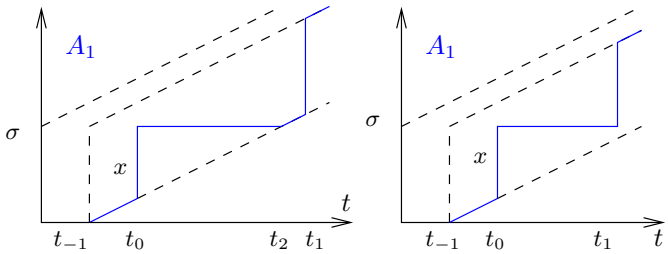

Figure 6: Worst arrival processes. There are two possible processes: $t_1 > t_2$ (left) or $t_1 = t_2$ (right).

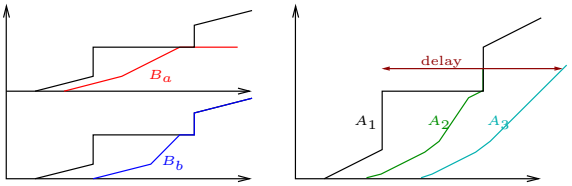

Figure 7: Input process in server c for the arrival process of Figure 6(a).

It seems very difficult to generalize such a process to any acyclic network. Here, the principle is to generate bursts of maximal possible size that are synchronized to the output dates of the b.u.o. in the server it crosses.

5. CONCLUSION

In this paper we have investigated how Network Calculus lends itself to the performance evaluation of multipath routing schemes. In a fluid model and for a routing scheme distributing data with respect to fixed proportions, we have shown that one can provide efficiently some qualitative guarantees like the ability to route data so that the network remains stable. When it comes to computing tight bounds on the worst-case end-to-end delay or backlog, our toy example shows that the analysis may be much more difficult. It does not mean that routing in models with Network Calculus constraints can not be evaluated tightly, but just that new reasonings and tools are required to achieve this goal. As a matter of fact, the difficulties encountered in the analysis of our routing scheme are strongly linked to the difficulties raised in the analysis of models with aggregate scheduling [15, 16, 17, 5, 11, 10, 12, 3]. It is likely that further work about multipath routing will use some results from those later studies.

6. REFERENCES

[1] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin. *Network Flows*. Prentice Hall, 1993.

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

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

[4] 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.

[5] A. Bouillard, B. Gaujal, S. Lagrange, and E. Thierry. Optimal routing for end-to-end guarantees using network calculus. Technical report, INRIA, 2008.

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

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

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

[9] L. R. Ford and D. R. Fulkerson. *Flows in Networks*. Princeton University Press, 1962.

[10] 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.

[11] 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.

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

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

[14] J. Schmitt, F. Zdarsky, and M. Fidler. Delay bounds under arbitrary multiplexing: When network calculus leaves you on the lurch... In *INFOCOM*, 2008.

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

[16] J. B. Schmitt, F. A. Zdarsky, and M. Fidler. Delay bounds under arbitrary multiplexing. Technical report, University of Kaiserslautern, 2007.

[17] 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.