# End-to-End Worst-Case Analysis of Non-FIFO Systems

## Technical Report No. 370/08

Jens B. Schmitt, Nicos Gollan, Ivan Martinovic

Distributed Computer Systems Lab (DISCO), University of Kaiserslautern, Germany

**Abstract.** In this report, delay bounds in data flow systems with non-FIFO service disciplines are explored. It is shown that conventional network calculus definitions of a service curve are not satisfying under the assumption of non-FIFO service. Either the definition is too strict to allow for a concatenation and consequent beneficial end-to-end analysis, or it is too loose and thus results in infinite delay bounds. Hence, a new definition is proposed and demonstrated to achieve both finite delay bounds and a concatenation of systems resulting in a favourable end-to-end analysis. In particular, we show that the celebrated pay bursts only once phenomenon is retained under non-FIFO service. This is contrary to a finding by Rizzo and Le Boudec [20]. The reasons for this contradiction are also discussed.

The new service curve definition preserves the min-plus algebraic approach known from network calculus at the cost of a loss of tightness of delay bounds under certain circumstances. This is shown by devising an alternative method that achieves tight bounds; this method departs from the algebraic approach. Yet, despite the loss of tightness the algebraic approach bears a number of advantages. Most important, it allows to deal elegantly with the frequent situation of a mixture of FIFO and non-FIFO servers, eventually resulting in better delay bounds.

**Keywords**: Network calculus, non-FIFO, pay bursts only once, concatenation property.

## 1 Introduction

### 1.1 Motivation

Network calculus is a min-plus system theory for deterministic queuing systems which builds on the calculus for network delay in [9], [10]. The important concept of service curve was introduced in [1, 6, 11, 16, 21]. The service curve based approach facilitates the efficient analysis of tandem queues where a linear scaling of performance bounds in the number of traversed queues is achieved as elaborated in [8] and also referred to as pay bursts only once phenomenon in [18]. A detailed treatment of min-plus algebra and of network calculus can be found in [2] and [7], [18], respectively.

Network calculus has found numerous applications, most prominently in the Internet's Quality of Service proposals IntServ and DiffServ, but also in other scenarios like wireless sensor networks [15, 22], switched Ethernets [25], Systems-on-Chip (SoC) [5], or even to speed-up simulations [14]. Hence, besides queueing theory it has established as a valuable methodology.

However, with respect to a single flow it is common in network calculus analyses to assume that *FIFO scheduling* is applied. This is restrictive, since for many real systems this assumption cannot always be made: In several studies of Internet traffic it has been shown that packet reordering is a frequent event (see for example [3, 13]). According to these studies this is due to a growing amount of parallelism on a global (use of multiple paths) as well as on a local (device) level. In particular, for scalability reasons routers often contain a complex multi-stage switching fabric which cannot ensure to preserve the order of arrivals at its output. Furthermore, the use of link aggregation, where multiple physical lines are aggregated into a single virtual link, may often lead to non-FIFO behavior [4]. Also, in wireless networks, reordering of packets is a frequent phenomenon due to the use of retransmissions and sliding window protocols to recover from transmission failures. As a last example, let us mention wireless sensor networks in which packet scheduling decisions may be based on the data values contained in the packets following a data-centric paradigm. Under such circumstances hardly anything may be assumed about the scheduling order, let alone FIFO behaviour.

So, from an application perspective there is enough demand to warrant an investigation on how network calculus can be extended towards the analysis of non-FIFO systems. Immediate questions that come up are:

– Can existing network calculus concepts be carried over to the non-FIFO case?
– Is an efficient end-to-end analysis still possible?
– What is the cost in terms of performance bounds compared to pure FIFO systems?

### 1.2 Related Work

To the best of our knowledge, there is amazingly little existing work on the treatment of non-FIFO systems in the context of network calculus. Remarkably, in his pioneering paper [9], Cruz briefly showed how to derive a delay bound for a single work-conserving server under a general scheduling assumption (comprising any non-FIFO behaviour) based on the observation that the maximum backlogged period can be bounded given that traffic is regulated. Similar results can also be found in [7]. Yet, the multiple node case is not treated in these.

In [17], Le Boudec and Charny investigate a non-FIFO version of the Packet Scale Rate Guarantee (PSRG) node model as used in DiffServ's Expedited Forwarding definition. They show that for a single node case the delay bound from the FIFO case still applies while it does not for a specific two node case. They leave more general concatenation scenarios for further study.

The most directly related work to ours was done by Rizzo and Le Boudec [20]. They investigate delay bounds for non-FIFO guaranteed rate nodes and show that a previously derived delay bound [12] is not valid in the non-FIFO case (against common belief). Furthermore, they derive a new delay bound based on network calculus results. Their delay bound does not exhibit the nice pay bursts only once phenomenon any more. Based on sample path arguments they argue that their bound is tight and thus conclude "pay bursts only once does not hold for non-FIFO guaranteed rate nodes". By contrast, in this report we show that non-FIFO systems still possess a concatenation property and that Rizzo and Le Boudec's conclusion is not valid in the general case (their tightness proof contains a simplifying assumption which however is crucial and does restrict generality).

### 1.3 Contributions

In this work, we extend network calculus concepts, in particular the service curve definition, such that FIFO scheduling is no longer a necessary assumption when performing an end-to-end worst-case delay analysis. In particular, the following contributions are made:

– We demonstrate the difficulties with existing service curve definitions under non-FIFO scheduling.
– We introduce a new service curve definition that enables a true end-to-end analysis for non-FIFO systems.
– We show that, contrary to literature results, the pay bursts only once phenomenon still holds for non-FIFO systems.
– We discuss the tightness of the new bounds and show how to apply them in mixed FIFO and non-FIFO scenarios.

## 2 Preliminaries on Network Calculus

As network calculus is built around the notion of cumulative functions for input and output flows of data, the set $\mathcal{F}$ of real-valued, non-negative, and wide-sense increasing functions passing through the origin plays a major role:

$$\mathcal{F} = \left\{ f : \mathbb{R}^+ \to \mathbb{R}^+, \forall t \geq s : f(t) \geq f(s), f(0) = 0 \right\}.$$

In particular, the input function $F(t)$ and the output function $F'(t)$, which cumulatively count the number of bits that are input to, respectively output from, a system $\mathcal{S}$, are in $\mathcal{F}$. Throughout the report, we assume in- and output functions to be continuous in time and space. Note that this is not a general limitation as there exist transformations between discrete and continuous models [18].

There are two important min-plus algebraic operators:

**Definition 1.** *(Min-plus Convolution and Deconvolution) The min-plus convolution and deconvolution of two functions $f, g \in \mathcal{F}$ are defined to be*

$$(f \otimes g)(t) = \inf_{0 \leq s \leq t} \{ f(t-s) + g(s) \},$$

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

It can be shown that the triple $(\mathcal{F}, \wedge, \otimes)$, where $\wedge$ denotes the minimum operator (which ought to be taken pointwise for functions), constitutes a dioid [18]. Also, the min-plus convolution is a linear operator on the dioid $(\mathbb{R} \cup \{+\infty\}, \wedge, +)$, whereas the min-plus deconvolution is not. These algebraic characteristics result in a number of rules that apply to those operators, many of which can be found in [18], [7]. Let us now turn to the performance characteristics of flows which can be bounded by network calculus means:

**Definition 2.** *(Backlog and Virtual Delay) Assume a flow with input function $F$ that traverses a system $\mathcal{S}$ resulting in the output function $F'$. The* backlog *of the flow at time $t$ is defined as*

$$b(t) = F(t) - F'(t).$$

*Assuming* FIFO *delivery, the* virtual delay *for a bit input at time $t$ is defined as*

$$vd(t) = \inf\{\tau \geq 0 : F(t) \leq F'(t+\tau)\}.$$

Next, the arrival and departure processes specified by input and output functions are bounded based on the central network calculus concepts of arrival and service curves:

**Definition 3.** *(Arrival Curve) Given a flow with input function $F$, a function $\alpha \in \mathcal{F}$ is an arrival curve for $F$ iff*

$$\forall t, s \geq 0, s \leq t : F(t) - F(t-s) \leq \alpha(s) \Leftrightarrow F = F \otimes \alpha.$$

A typical example of an arrival curve is given by an affine arrival curve $\gamma_{r,b}(t) = b + rt$, $t > 0$ and $\gamma_{r,b}(t) = 0$, $t \leq 0$, which corresponds to token-bucket traffic regulation.

**Definition 4.** *(Service Curve – SC) If the service provided by a system $\mathcal{S}$ for a given input function $F$ results in an output function $F'$ we say that $\mathcal{S}$ offers a service curve $\beta$ iff*

$$F' \geq F \otimes \beta.$$

*For continuous functions $F$ and $\beta$ this is equivalent to the following condition*

$$\forall t : \exists s : F'(t) \geq F(s) + \beta(t-s).$$

A typical example of a service curve is given by a so-called rate-latency function $\beta_{R,T}(t) = R(t-T) \cdot 1_{\{t>T\}}$, where $1_{\{cond\}}$ is 1 if the condition $cond$ is satisfied and 0 otherwise.

A number of systems fulfill a stricter definition of service curve [18], which is useful as it permits certain derivations that are not feasible under the more general service curve model:

**Definition 5.** *(Strict Service Curve – $S^2C$) Let $\beta \in \mathcal{F}$. System $\mathcal{S}$ offers a* strict service curve *$\beta$ to a flow, if during any backlogged period of duration $u$ the output of the flow is at least equal to $\beta(u)$. A backlogged period of duration $u$ at time $t$ is defined by the fact that $\forall s \in (t-u, t] : b(s) > 0$.*

Note that any node satisfying $S^2C$ also satisfies $SC$, but not the other way around. For example, nodes operating under a delay-based scheduler and guaranteeing that a work unit arriving at any time $t$ will not leave the node later than $t+T$ for some fixed $T > 0$, are known to provide a service curve $\delta_T = \infty \cdot 1_{\{t>T\}}$ [18]. Yet, such a variable latency node does not provide $\delta_T$ as a *strict* service curve. In fact, it does not provide any strict service curve apart from the trivial case $\beta = 0$. On the other hand, there are still many schedulers that offer strict service curves, for example most of the generalized processor sharing-emulating schedulers, e.g. PGPS [19] offer a strict service curve of the rate-latency type.

Using those concepts it is possible to derive *tight* performance bounds on backlog, *virtual* delay and output:

**Theorem 1.** *(Performance Bounds) Consider a system $\mathcal{S}$ that offers a service curve $\beta$. Assume a flow $F$ traversing the system has an arrival curve $\alpha$. Then we obtain the following performance bounds:*

$$\begin{aligned}
&\textit{backlog: } \forall t : b(t) \leq (\alpha \oslash \beta)(0) =: v(\alpha, \beta),\\
&\textit{virtual delay: } \forall t : vd(t) \leq \inf \{t \geq 0 : (\alpha \oslash \beta)(-t) \leq 0\}\\
&\qquad\qquad\qquad\qquad =: h(\alpha, \beta),\\
&\textit{output (arrival curve } \alpha' \textit{ for } F'\textit{): } \alpha' = \alpha \oslash \beta.
\end{aligned}$$

One of the strongest results of network calculus is the concatenation theorem that enables us to investigate tandems of systems as if they were single systems:

**Theorem 2.** *(Concatenation Theorem for Tandem Systems) Consider a flow that traverses a tandem of systems $\mathcal{S}_1$ and $\mathcal{S}_2$. Assume that $\mathcal{S}_i$ offers a service curve $\beta_i$ to the flow. Then the concatenation of the two systems offers a service curve $\beta_1 \otimes \beta_2$ to the flow.*

Using the concatenation theorem, it is ensured that an end-to-end analysis of a tandem of servers still achieves tight performance bounds, which in general is not the case for an iterative per-node application of Theorem 1.

## 3 Why Conventional Network Calculus Does Not Work Well for Non-FIFO Systems

Our goal is to depart from the FIFO assumption under which the bound on the virtual delay in Theorem 1 can be computed. Hence, we are interested in bounding the *real* delay which is simply defined as follows:

**Definition 6.** *Assume a flow with input function $F$ that traverses a system $\mathcal{S}$ resulting in the output function $F'$. The* real delay *for a bit input at time $t$ and output at time $t'$ is defined as*

$$rd(t) = t' - t.$$

Note that $rd(t)$ can be in any relation to $vd(t)$ $(<, >, =)$. However, any scheduling order other than FIFO results in an increase of the bound for the real delay (obviously, $\forall t : vd(t) = rd(t)$ under FIFO):

**Theorem 3.** *(FIFO is Best-Case Scheduling) With respect to the worst-case real delay, FIFO scheduling is the best scheduling order as there is no other scheduling order that achieves a lower worst-case real delay.*

*Proof.* Assume at time $t_0$ a bit which experiences the worst-case real delay is input to a FIFO system. Now assume we can change the scheduling order of bits. If the bit is further delayed by scheduling bits that arrived later, then certainly the real delay of that new scheduling order will be worse. Scheduling that bit earlier will make the real delay $rd(t_0)$ smaller, yet, the bit which was just ahead of the above bit is now leaving the system when the above bit would have left, yet that bit must have arrived at time $t_1 \leq t_0$ such that the real delay of that bit is higher than or equal to the one from the FIFO worst-case bit, i.e., $rd(t_0) \leq rd(t_1)$.

So, in a certain sense it could be considered a logical break for a worst-case analysis methodology to assume FIFO scheduling as this actually constitutes a best-case assumption. This can be seen as a further motivation for making no restrictive assumptions on the scheduling order.

As mentioned in the previous section, one basically has two general options for node modelling: $SC$ and $S^2C$. In the next two subsections we demonstrate that both options do not result in achieving satisfying bounds for the real delay.

### 3.1 Using Service Curves ($SC$) for Non-FIFO Systems

As the $SC$ definition bears the advantages that many systems belong to that class and that it possesses a concatenation property, it is worthwhile an attempt to apply it also in the case of non-FIFO systems. Yet, the following example shows that it is impossible to bound the real delay in non-FIFO systems based solely on the $SC$ definition:

*Example 1.* Assume a single node system $\mathcal{S}$ which offers a rate-latency service curve $\beta = \beta_{2,1}$ to a flow $F$ which is constrained by an affine arrival curve $\alpha = \gamma_{1,1}$. Now assume the flow to be greedy, that means $F = \alpha$ and the server to be lazy, that means $F' = F \otimes \beta$. Thus, we obtain

$$\begin{aligned} F' &= \alpha \otimes \beta = \gamma_{1,1} \otimes \beta_{2,1} = \gamma_{1,1} \otimes \gamma_{1,1} \otimes \delta_T \\ &= (\gamma_{1,1} \wedge \gamma_{1,1}) \otimes \delta_T = \gamma_{1,1} \otimes \delta_T < \gamma_{1,1} = F. \end{aligned}$$

Hence, $\forall t \geq 0 : F'(t) < F(t)$, or equivalently, $\forall t \geq 0 : b(t) > 0$, which means the system remains backlogged at all times and a certain work unit can be forever in the system under these circumstances. Thus, the real delay of that work unit is unbounded. Note that using the standard FIFO assumption, we can of course bound the virtual delay of the system by $\forall t \geq 0 : vd(t) \leq \frac{3}{2}$.

From this example, we see that the *SC* property is too weak as a concept for analysing non-FIFO systems.

### 3.2 Using Strict Service Curves ($S^2C$) for Non-FIFO Systems

Since, the *SC* property is too weak in order to derive a delay bound without a FIFO assumption on the system's scheduling order, it is interesting to investigate whether $S^2C$ can deal with that situation. In fact, as was already shown by Cruz [9] (and can also be found in [7] (Lemma 1.3.2)), the intersection point between an arrival and a *strict* service curve constitutes a bound on the length of the maximum backlogged period and thus also a bound on the real delay for such a system:

**Theorem 4.** *(Real Delay Bound for Single $S^2C$ Node) Consider a system $\mathcal{S}$ that offers a strict service curve $\beta$. Assume a flow $F$ traversing the system has an arrival curve $\alpha$. Then we obtain the following bound on the real delay:*

$$rd(t) \leq \sup\{s \geq 0 : \alpha(s) \geq \beta(s)\} =: i(\alpha, \beta).$$

So, the situation has improved in comparison to the *SC* case: Based on the single node result one can conceive, for the multiple node case, an iterative application of Theorem 4 together with the output bound from Theorem 1. More specifically, if $n$ $S^2C$ non-FIFO nodes, each providing a strict service curve $\beta_j, j = 1, \ldots, n$, are to be traversed by an $\alpha$-constrained flow then a bound on the real delay can be calculated as

$$rd(t) \leq \sum_{j=1}^{n} i(\alpha \oslash \bigotimes_{k=1}^{j-1} \beta_k, \beta_j).$$

Setting for example $\beta_j = \beta_{R,T}, j = 1, \ldots, n$ and $\alpha = \gamma_{r,b}$ this results in

$$rd(t) \leq \frac{n(b + RT) + \frac{n}{2}(n-1)rT}{R - r}$$

Here, we see the typical drawback of additive bounding methods, with the burst of the traffic being paid $n$ times as well as a quadratic scaling of the bound in the number of nodes [8, 18]. The key to avoid this behaviour is to perform an end-to-end analysis based on the concatenation theorem. Yet, as we demonstrate in the next example $S^2C$ does not possess such a concatenation property.

*Example 2.* ($S^2C$ Possesses No Concatenation Property) Assume two systems $\mathcal{S}_1$ and $\mathcal{S}_2$, both providing a strict rate-latency service curve $\beta^i = \beta_{1,1}, i = 1, 2$, which are traversed in sequence by a flow $F$. Let $F_1'$ and $F_2'$ be the output functions from $\mathcal{S}_1$ and $\mathcal{S}_2$, respectively. As a candidate strict service curve for the composite system, we consider $\beta^{1,2} = \beta^1 \otimes \beta^2 = \beta_{1,2}$.

We now construct a backlogged period $[t_1, t_2]$ of the composite system such that

$$F_2'(t_2) - F_1'(t_1) < \beta^{1,2}(t_2 - t_1).$$

thereby showing that $\beta^{1,2}$ is not a strict service curve for the composite system:

Let $t_1 = 0$ and $t_2 = 3$ and assume the following behaviour of the input and output function

$$F(t) = \begin{cases} \epsilon \ 0 < t < 2 \\ 2\epsilon \ 2 \leq t \leq 3 \end{cases} \qquad F_1'(t) = \begin{cases} 0 \ 0 \leq t \leq 1 \\ \epsilon \ 1 < t \leq 3 \end{cases}$$

$$F_2'(t) = \begin{cases} 0 \ 0 \leq t \leq 2 \\ \epsilon \ 2 < t \leq 3 \end{cases},$$

with any $\epsilon > 0$. It is easy to check that the composite system is continuously backlogged during $[0, 3]$ as well as that each individual system is not violating its strict service curve property. Nevertheless, for any choice of $\epsilon < 1$ we obtain

$$F_2'(3) - F_2'(0) = \epsilon < \beta^{1,2}(3) = 1,$$

which shows that $\beta^{1,2}$ is not $S^2C$ for the composite system (while, of course, being $SC$ for it). In fact, extending the example appropriately it can be shown that the only strict service curve that can be guaranteed by the composite system is the trivial case $\beta = 0$. This can be seen by making $\epsilon$ arbitrarily small and alternating between backlogged and idle periods of the individual systems sufficiently often. Another way to view this, is that the backlogged period of a composite system cannot be bounded based on the individual systems providing a strict service curve.

So, from this discussion we can conclude that $S^2C$ is too strict as a concept in order to allow for tight bounds under the non-FIFO assumption, since it possesses no concatenation property for the multiple node case.

The bottom line of this section is that we need a new node model: It should allow for calculating a bound on the real delay and, yet, also have a concatenation property in order to avoid loose additive bounds.

## 4 Introducing a New Service Curve Model: Sufficiently Strict Service Curve

In this section, we introduce a new node model that allows to bound the delay over a tandem of non-FIFO systems well as it possesses a concatenation property. Central to the new service curve definition is the notion of a *maximum dwell period*.

**Definition 7.** *(Maximum Dwell Period) The maximum dwell period at time $t$, denoted as $D(t)$, is the length of the interval $[t^0(t), t]$, i.e., $D(t) = t - t^0(t)$, where $t^0(t)$ is the arrival time of the oldest work unit in the system at time $t$ under* all possible *scheduling orders. If the system is empty at time $t$, then by definition $t^0(t) = t$ and $D(t) = 0$.*

For a single node with a simple buffer, the maximum dwell period at time $t$ equals the backlogged period at time $t$; consequently, $t^0(t)$ equals the start of the last backlogged period. The scheduling order that achieves the maximum dwell period for such a simple buffer is LIFO. However, the maximum dwell period of more complex systems can be shorter than the backlogged period of the system, which for complex systems is generally unbounded (see Section 3.2, Example 2).

Furthermore, note that the maximum dwell period is always within a backlogged period of the system, even if the system is not employing the scheduling order leading to the maximum dwell period. This can be understood by assuming that each work unit is time-stamped at entrance to the system and observing that exchanging those time stamps between work units always allows to achieve the maximum dwell period without affecting the length of the backlogged period of the system.

Using the notion of the maximum dwell period, the new service curve definition can be introduced:

**Definition 8.** *($S^3C$) Given a system $S$ with input function $F$ and output function $F'$, $\beta \in \mathcal{F}$ is a* sufficiently strict service curve *($S^3C$) if for any $t \geq 0$ it applies that*

$$F'(t) \geq F(t - D(t)) + \beta(D(t)).$$

Before we apply that definition, let us discuss its relation with the other node models.

*Remark 1.* (Relation with Other Node Models) We have the following implications respectively non-implications:

$$S^2C \Rightarrow S^3C \Rightarrow SC$$
$$SC \not\Rightarrow S^3C \not\Rightarrow S^2C$$

These relations can be readily checked:

1. $S^2C \Rightarrow S^3C$ can be seen from the fact that the maximum dwell period is certainly within a backlogged period of the system.
2. $S^3C \Rightarrow SC$ can be seen from the fact that the required existence of time $s$ in the definition of $SC$ is fixed in the $S^2C$ definition to be $t - D(t)$.
3. $SC \not\Rightarrow S^3C$ is obvious, because it does not have to apply that $s = t - D(t)$.
4. $S^3C \not\Rightarrow S^2C$ is obvious, because $S^3C$ makes no statement for arbitrary backlogged periods, but only for the specific backlogged periods $[t - D(t), t]$.

Since $S^2C$ implies $S^3C$, all schedulers known to deliver strict service curves also deliver sufficiently strict service curves. Hence, $S^3C$ is not too restricting as a node model. Even more so, note that $S^3C$ also applies for delay-based schedulers which guarantee any work unit to be served within a time period $T$ after their arrival, because such a node can be abstracted as providing a sufficiently strict service curve $\delta_T$. This can be understood by realising that at a given time $t$ any work unit that entered the system before $t - D(t)$ must have left the system again (for any scheduling order). Formally,

$$\forall \epsilon > 0 : F'(t) \geq F(t - D(t) - \epsilon).$$

As we assume $F$ to be continuous and because $D(t) \leq T$ according to the guarantee of a delay-based scheduler, this translates into

$$F'(t) \geq F(t - D(t)) = F(t - D(t)) + \delta_T(D(t)),$$

which constitutes $\delta_T$ as a sufficiently strict service curve for a delay-based scheduling node. Note that the node does not have to be FIFO, though its non-FIFO behaviour is restricted due to the delay guarantee: basically, a reordering can actually only lead to scheduling work units ahead of their deadline. On the other hand, if a delay-based scheduling node is abstracted as providing $\delta_T$ as $SC$, then it must be assumed FIFO to calculate its delay bound.

As we show in the next two theorems, the $S^3C$ definition achieves for non-FIFO systems the two attractive features known from conventional network calculus with FIFO systems: a concatenation property and a bound on the delay, yet now on the real instead of the virtual delay.

**Theorem 5.** *(Concatenation Theorem for Tandem $S^3C$ Systems) Consider a flow with input function $F$ that traverses a tandem of systems $\mathcal{S}_1$ and $\mathcal{S}_2$. Assume that $\mathcal{S}_i$ offers a sufficiently strict service curve $\beta_i$, $i = 1, 2$ to the flow. Then the concatenation of the two systems offers a sufficiently strict service curve $\beta_1 \otimes \beta_2$ to the flow $F$.*

*Proof.* Let the output function of system $\mathcal{S}_1$ (respectively $\mathcal{S}_2$) be denoted as $F'$ (respectively $F''$). Let $D_1(t)$ and $D_2(t)$ denote the maximum dwell periods at time $t$ for $\mathcal{S}_1$ and $\mathcal{S}_2$. We look at the composite system at some time $t$. First, we use the $S^3C$ property for $\mathcal{S}_2$, i.e., we have

$$F''(t) - F'(t - D_2(t)) \geq \beta_2(D_2(t)). \tag{1}$$

Next, we bring in the $S^3C$ property for $\mathcal{S}_2$ at time $t - D_2(t)$

$$F'(t - D_2(t)) - F(t - D_2(t) - D_1(t - D_2(t))) \geq \beta_1(D_1(t - D_2(t)). \tag{2}$$

Adding Equ. (1) and Equ. (2) and by denoting $t_1^0(t)$ (respectively $t_2^0(t)$) as the arrival time of a work unit which experiences the maximum dwell period at time $t$ for $\mathcal{S}_1$ (respectively $\mathcal{S}_2$), we obtain

$$F''(t) - F(t - D_2(t) - D_1(t - D_2(t)))$$

$$
\begin{aligned}
&\geq \beta_1(D_1(t - D_2(t)) + \beta_2(D_2(t)) \\
&= \beta_1(t_2^0(t) - t_1^0(t_2^0(t))) + \beta_2(t - t_2^0(t)) \\
&\geq \inf_{t_1^0(t_2^0(t)) \leq s \leq t} \{\beta_1(s - t_1^0(t_2^0(t))) + \beta_2(t - s)\} \\
&= \inf_{0 \leq u \leq t - t_1^0(t_2^0(t))} \{\beta_1(u) + \beta_2(t - t_1^0(t_2^0(t)) - u)\} \\
&= (\beta_1 \otimes \beta_2)(t - t_1^0(t_2^0(t))) \\
&= (\beta_1 \otimes \beta_2)(D_2(t) + D_1(t - D_2(t))).
\end{aligned}
$$

Hence, what remains to be shown to make the proof complete is that

$$D_{1,2}(t) = D_2(t) + D_1(t - D_2(t)),$$

where $D_{1,2}(t)$ denotes the maximum dwell period at time $t$ for the composite system. We proof this in the following lemma. So, finally we arrrive at

$$F''(t) - F(t - D_{1,2}(t)) \geq (\beta_1 \otimes \beta_2)(D_{1,2}(t)),$$

which constitutes $\beta_1 \otimes \beta_2$ as a sufficiently strict service curve for the composite system.

**Lemma 1.** *For the scenario and the notations from Theorem 5 it applies that*

$$D_{1,2}(t) = D_2(t) + D_1(t - D_2(t)). \tag{3}$$

*Proof.* Let us assume there is no backlog at $\mathcal{S}_2$ at time t, that means $D_2(t) = 0$ and the maximum dwell period at time $t$ of the composite system is governed by $\mathcal{S}_1$:

$$D_{1,2}(t) = D_1(t) = D_2(t) + D_1(t - D_2(t)).$$

Hence Equ. (3) is satisfied for that case.

Now assume there is a non-zero backlog at $\mathcal{S}_2$ at time $t$, that means $D_2(t) > 0$. If there is no backlog at $\mathcal{S}_1$ at time $t - D_2(t)$, then $D_1(t - D_2(t)) = 0$ and the maximum dwell period of the composite system is governed by $\mathcal{S}_2$ (since $\mathcal{S}_1$ can have only newer work units at time $t$ as the older ones were already cleared at time $t - D_2(t)$):

$$D_{1,2}(t) = D_2(t) = D_2(t) + D_1(t - D_2(t)).$$

Hence Equ. (3) is also satisfied for that sub-case. Now assume $\mathcal{S}_1$ is backlogged at time $t - D_2(t)$, that means $D_1(t - D_2(t)) > 0$. Thus, we can easily construct a scheduling order such that we achieve a dwell period of $D_2(t) + D_1(t - D_2(t)))$ (we just need to make sure that the oldest work unit does not leave $\mathcal{S}_2$ before $t$, this is possible no matter where that work unit is located since $\mathcal{S}_2$ is backlogged at time $t$) and therefore

$$D_{1,2}(t) \geq D_2(t) + D_1(t - D_2(t)). \tag{4}$$

What remains to be shown is that

$$D_{1,2}(t) \le D_2(t) + D_1(t - D_2(t)). \tag{5}$$

Assuming that Equ. (5) does not hold implies that there is a scheduling such that a work unit with a dwell period larger than $D_2(t) + D_1(t - D_2(t))$ is in the composite system at time $t$. Assume the dwell period for that work unit to be $D_2(t) + D_1(t - D_2(t)) + \epsilon$, for some $\epsilon > 0$. Such a work unit cannot be in $\mathcal{S}_2$ because otherwise (since $D_1(t - D_2(t)) > 0$)

$$D_2(t) \ge D_2(t) + D_1(t - D_2(t)) + \epsilon > D_2(t)$$

If such a work unit were in $\mathcal{S}_1$, it would obviously have to be there at $t - D_2(t)$ already, yet this would mean

$$\begin{aligned} D_1(t - D_2(t)) &\ge D_2(t) + D_1(t - D_2(t)) + \epsilon - D_2(t) \\ &> D_1(t - D_2(t)) \end{aligned}$$

which completes the contradiction and shows that Equ. (5) must hold, so that, altogether, for all cases it applies that

$$D_{1,2}(t) = D_2(t) + D_1(t - D_2(t)).$$

Next, we show how to bound the real delay for an $S^3C$ node.

**Theorem 6.** *(Real Delay Bound for an $S^3C$ System) Consider a system $\mathcal{S}$ that offers a sufficiently strict service curve $\beta$. Assume a flow $F$ traversing the system has an arrival curve $\alpha$. Then we obtain the following bound on the real delay:*

$$rd(t) \le \sup\{s \ge 0 : \alpha(s) \ge \beta(s)\} = i(\alpha, \beta).$$

*Proof.* As above we denote the maximum dwell period at time $t$ by $D(t)$ and the arrival time of the work unit corresponding to $D(t)$ by $t^0(t)$. We can make the following observation for the backlog of the system at time $t$:

$$\begin{aligned} b(t) &= F(t) - F'(t) \\ &= F(t) - F(t^0(t)) - (F'(t) - F(t^0(t))) \\ &\le \alpha(D(t)) - \beta(D(t)). \end{aligned}$$

Here, we used the arrival curve as well as the $S^3C$ property from the assumptions. This relation implies that

$$\alpha(D(t)) \ge \beta(D(t)) + b(t).$$

This now allows to bound the maximum dwell period at time $t$, by the following observation:

$$\begin{aligned} D(t) &\le \sup\{0 \le s \le t : \alpha(s) \ge \beta(s) + b(t)\} \\ &\le \sup\{0 \le s \le t : \alpha(s) \ge \beta(s)\} \\ &\le \sup\{s \ge 0 : \alpha(s) \ge \beta(s)\}. \\ &= i(\alpha, \beta). \end{aligned}$$

Since the bound is independent of $t$, it applies for all $t$ and, furthermore, since a bound over all maximum dwell periods is certainly a bound for all real delays, the proof is completed.

By combining the results from Theorem 5 and 6, we can, for the multiple node case involving $n$ non-FIFO nodes, each providing an $S^3C$ $\beta_j, j = 1, \ldots, n$, which are traversed by an $\alpha$-constrained flow, derive a bound on the real delay as

$$rd(t) \leq i(\alpha, \bigotimes_{j=1}^{n} \beta_j).$$

Looking at the same special case as in Section 3.2, i.e., $\beta_j = \beta_{R,T}$ and $\alpha = \gamma_{r,b}$, we obtain the following bound on the real delay

$$rd(t) \leq \frac{b + nRT}{R - r},$$

which improves considerably on the additive bound based on $S^2C$ from Section 3.2. We can perceive again the pay burst only once principle as the burst term appears only once as well as a linear scaling in the number of nodes. We provide some more quantitative observations in Subsection 4.1.

The delay bound argues over the maximum dwell period, under more knowledge about the non-FIFOness it could possibly be improved. Such an approach should parallel the characteristic that under FIFO scheduling the horizontal deviation actually allows to tighten the delay bound. We leave this for further study.

Other performance bounds besides the delay bound and further results, like output bound, backlog bound, etc., can be carried over from conventional network calculus as $S^3C$ implies $SC$. In fact, it is only the delay bound that is sensitive on the scheduling order. However, often in applications the delay bound is also the figure of most interest.

### 4.1 Numerical Experiments

To give some feeling for the improvements achievable by using the $S^3C$-based end-to-end analysis compared to an additive bounding based on $S^2C$ we provide some numerical experiments. In addition, we demonstrate what cost is incurred for releasing the FIFO assumption. For these numerical experiments we use simple settings: as arrival curve for the flow to be analysed we assume a token bucket $\gamma_{r,b}$ where we set $r = 10[Mbps]$ and $b = 5[Mb]$ unless we vary the rate $r$ to achieve a certain utilization; for the service curves of the nodes to be traversed we use a rate-latency function $\beta_{R,T}$ with $R = 20[Mbps]$ and $T = 0.01[s]$. Unless we use the number of nodes as a primary factor in the experiments we assume $n = 10$ nodes to be traversed by the flow under investigation.

**Comparison of Different Service Curve Models** In this first set of numerical experiment we investigate how end-to-end ($S^3C$) and additive ($S^2C$) analysis compare to each other. In Figure 1 the two methods are shown for a varying number of nodes (from 2 to 20). To exhibit the quadratic scaling of the $S^2C$-based method we also provide results for the same experiment with a larger number of nodes to be traversed (up to 100) in Figure 2. In both graphs it is obvious that the end-to-end analysis facilitated by the $S^3C$ definition is highly superior and scales linearly with the number of nodes.

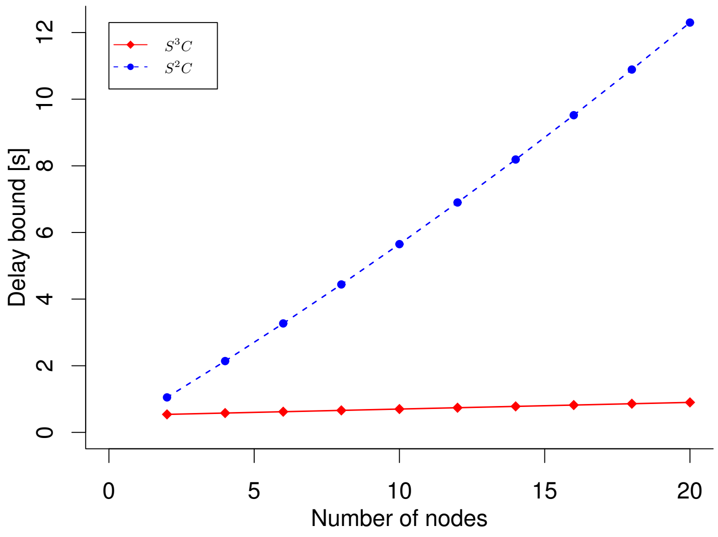

**Fig. 1.** Delay bounds under different service curve models depending on the number of nodes traversed.

A different view on the relative performance of $S^3C$- and $S^2C$-based bounding methods is provided in Figure 3. Here, the acceptable utilizations for a given delay bound are shown for both methods. This information can be used for admission control purposes. Again, as can be clearly seen, the $S^3C$-based method outperforms the $S^2C$-based method by far, especially for lower delay bounds.

**FIFO vs. Non-FIFO Delay Bounds** In the next set of numerical experiments, we investigate the cost of releasing the FIFO assumption in terms of delay bounds. For that purpose, we vary the utilization by increasing the sustained rate of the traffic flow under investigation (while at the same time scaling the bucket depth accordingly). As we can observe from Figure 4, only for higher utilizations there is a significant difference between the FIFO and non-FIFO delay bounds (at least for the $S^3C$ case). The bottom line is that only for highly utilized systems it is necessary to enforce a FIFO behaviour, as far as delay

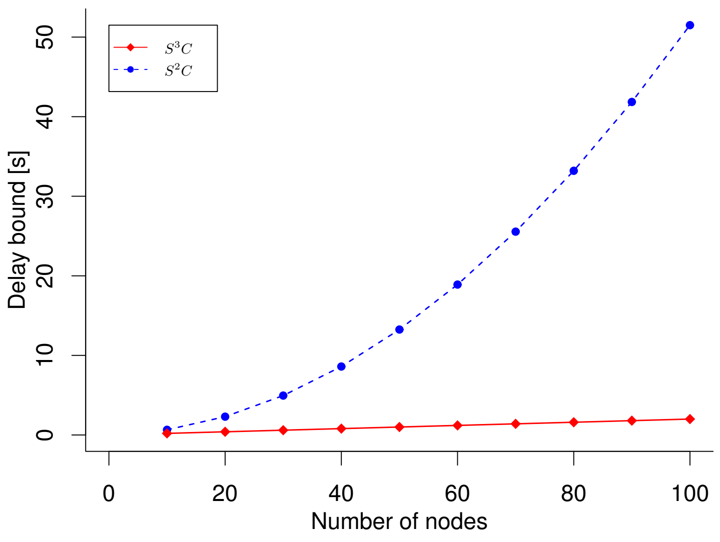

**Fig. 2.** Exposing the quadratic scaling of the additive bound based on $S^2C$.

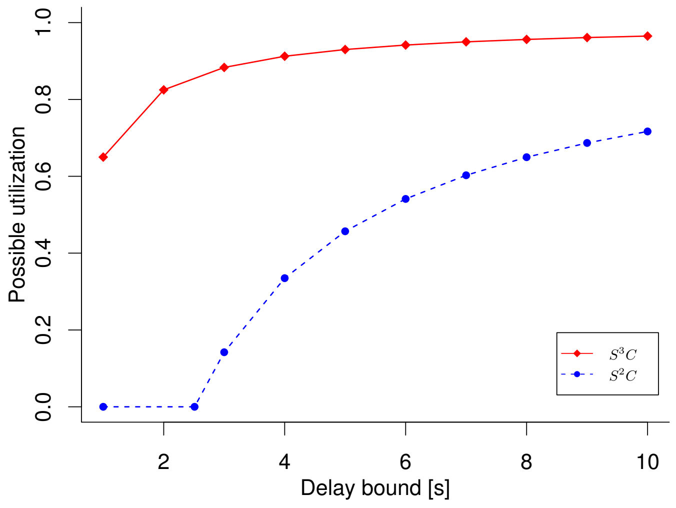

**Fig. 3.** Possible utilizations for a target delay bound under $S^2C$ and $S^3C$.

bounds are concerned. For systems with lower utilizations, optimizations such as for example link aggregation or multi-stage switching fabrics do not incur a high cost in terms of worst-case delay bounds.

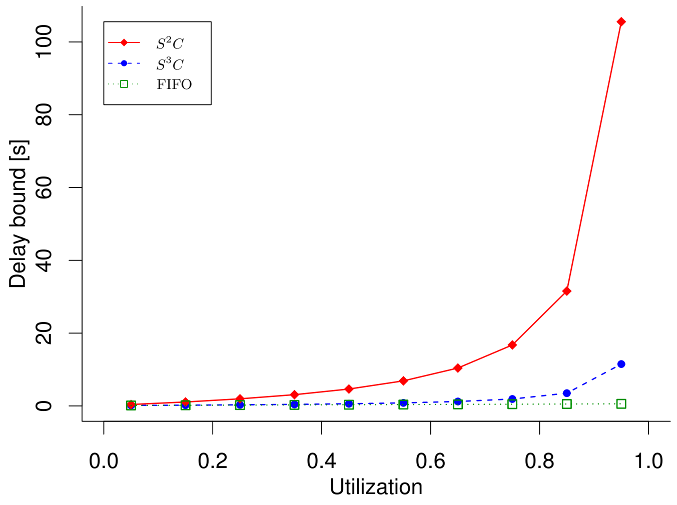

**Fig. 4.** FIFO vs. non-FIFO delay bounds depending on the utilization of the system.

### 4.2 A Note on "Pay bursts only once does not hold for non-FIFO guaranteed rate nodes" [20]

In the previous section, we established that the "pay bursts only once" phenomenon also holds for non-FIFO systems. This seems to be in sharp contrast to a literature result by Rizzo and Le Boudec [20]. They claim (in their article's title) that "pay bursts only once does not hold for non-FIFO guaranteed rate nodes". While they use a different node model which makes a direct comparison with our bounds somewhat difficult, we argue that their claim is not valid in the general case.

Some preliminaries are necessary: A *guaranteed rate (GR) node* is a server that guarantees an upper bound on the departure time of any given packet of a flow, depending on the packet's arrival time and the bounds of preceding packets. To this end, it uses the concept of a *guaranteed rate clock* (GRC) for any given packet $p^j$ of length $l_j$, where $p^j$ is the $j^{\text{th}}$ packet arriving at the node, $A(p^j)$ its arrival time, and $R$ the server's allocated rate for the flow:

$$GRC(p^j) = \begin{cases} 0, & j = 0 \\ \frac{l_j}{R} + \max\left\{A(p^j),\, GRC(p^{j-1})\right\}, & j > 0 \end{cases}$$

A GR node guarantees that any packet $p^j$ is delivered by the time $GRC(p^j) + e$, where $e$ depends on the actual packet scheduling algorithm.

Note that this definition restricts the non-FIFO behavior of a GR node. In general non-FIFO service, a packet may remain in a server as long as that server is backlogged. This is not possible in the GR case, since every packet gets a finite GRC value when it is received; thus reordering of packets is limited, potentially giving way to less pessimistic bounds than under pure service curve considerations.

In [20], essentially an additive delay bound based on a fundamental relationship between GR nodes and service curves as well as a per-node delay bound for non-FIFO GR nodes is derived. Under a token-bucket arrival curve $\gamma_{r,b}$ and assuming a propagation delay $\tau^i$ between nodes $i$ and $i+1$ as well as a maximum packet size $l_{max}$, the bound in [20] for a tandem of $n$ servers is given as

$$d^{[20]} = n\frac{b}{R} + \frac{rl_{max}n\,(n-1)}{2R^2} + \frac{r}{R}\sum_{k=1}^{n}\sum_{i=1}^{k-1} e^i + \sum_{i=1}^{n}\left(e^i + \tau^i\right).$$

As can be seen, the burst is paid $n$ times and a quadratic scaling in $n$ is incurred, as it is typical for additive bounding methods. [20] claims that the bound is tight in general and thus concludes that "pay bursts only once does not hold for non-FIFO GR nodes".

Yet, a counter-example on the tightness of the bound in [20] can be constructed by choosing the following parameters: $n = 2$, $R = 1$, $e^i = 0$, $r = 0$, $b = 2$, $\tau^i = 0$, and all packets being of length 1. Then $d^{[20]} = 4$, however it is fairly obvious that $d^{tight} = 3$ is a valid delay bound for that scenario: In the worst case, the two packets of the flow arrive at the same time and one is served within one time unit and immediately transferred to the second node; the second packet is served in another time unit while the first is served by the second node; now the second node is free to serve the second packet in another time unit, amounting for the second packet in 3 time units delay altogether. Changing the order of the packets cannot matter here, as they are arriving at the same time anyway. Any spreading of the two packets can only result in a lower delay. This example can be generalized to a series of $n \geq 2$ nodes, where the tight delay bound $d^{tight} = n + 1$, but $d^{[20]} = n\frac{b}{R} = 2n$. This shows that the delay bound from [20] is not tight in the general case. In their tightness proof, Rizzo and Le Boudec make the following assumption: "In order to simplify the notation, we assume that $r = R$ ..." Under this assumption their delay bound is actually tight, yet for any scenario $r < R$ it is not tight any more. Thus they restrict generality by that seemingly innocent assumption.

A more comprehensive example for the untightness of the bound is shown in Figure 5. With this example it shall be demonstrated more clearly why the condition that $\rho = r$ actually is necessary for the bound to be tight. This example recreates the scenario from the original paper, but drops the assumption that $\rho = r$. Instead, it uses $\rho = 0.5$ and $r = 1$. According to the original bound, this results in $d^{[20]} = 25.5[tu]$; however, the example of a worst case sample path shown in the figure demonstrates that a packet can at most spend $22tu$

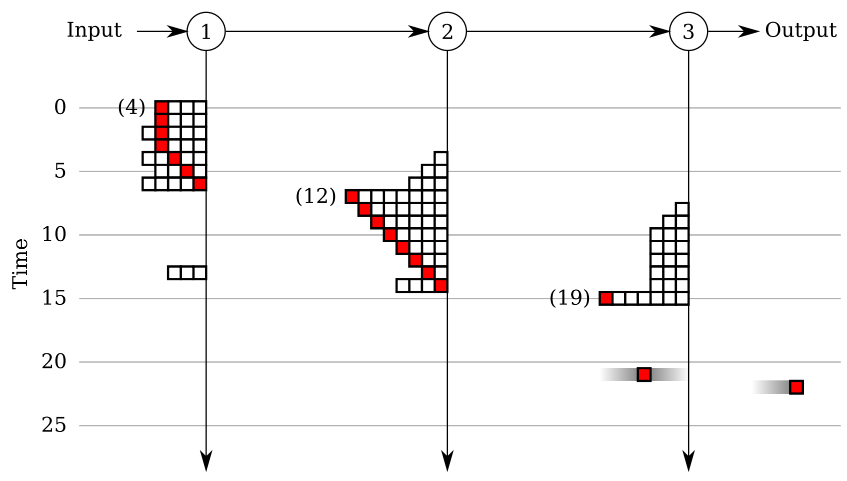

**Fig. 5.** Demonstration of the untighness of the delay bound presented in [20]. Shown are the buffers of three GR nodes over time. Numbers in brackets are the watched packet's (shaded) GRC. Buffer contents are only shown as long as they are relevant to the sample path.

in the system. The construction of that sample path follows the same principle as used in [20]: each time the watched packet $p_w$ (shaded in the figure) leaves a server, it gets sent along with the maximum number of packets, so that the receiving server has maximum leeway when reordering packets to maximize the watched packet's GRC. That value is shown in brackets. The original example used $r = R$, so each server could bunch the watched packet with a full burst of the arrival, thus fulfilling the first term of the delay bound $\left(M\frac{\sigma}{r}\right)$; however, in an unsaturated system, the GRC property may force a server to send $p_w$ *before* a full burst from the input can be propagated down to it:

This happens at server 2 at $t = 14$. The packet $p_w$ has been stamped with a GRC of 12 when it was received, and thus is guaranteed to be sent by the time $GRC_2(p_w) + e = 14$ ($GRC_i(p^j)$ is the GRC assigned to the $j^{\text{th}}$ packet at the $i^{\text{th}}$ server). Note that the packet numbering is per-server. However, to pay a full burst, the input would have to burst at $t = 13$, by which time it is only allowed to send 3 full packets (assuming packetization). This means that instead of a full burst of 4 packets, $p_w$ can only be sent along with at most 3 packets, with all other queued packets at server 2 having already been sent due to their GRCs having expired. This means that $GRC_3(p_w) = 19$, guaranteeing the packet's delivery to the output at or before $t = 21$, so together with the transmission delay $\tau$, it will be delivered after having spent at most $t = 22$ time units in the system.

If the input would wait for a full burst, server 2 would have depleted its queue by the time the first packets from the input arrived, and the same situation would arise again.

Figure 6 shows a comparison between the delay bounds obtained from the $S^3C$-based method and [20]. Up to a very high utilization, $S^3C$ yields less conservative bounds than [20], although the $S^3C$ node model is much less stringent than the GR node model. This again demonstrates the untightness of the bound from [20] for utilizations below 1.

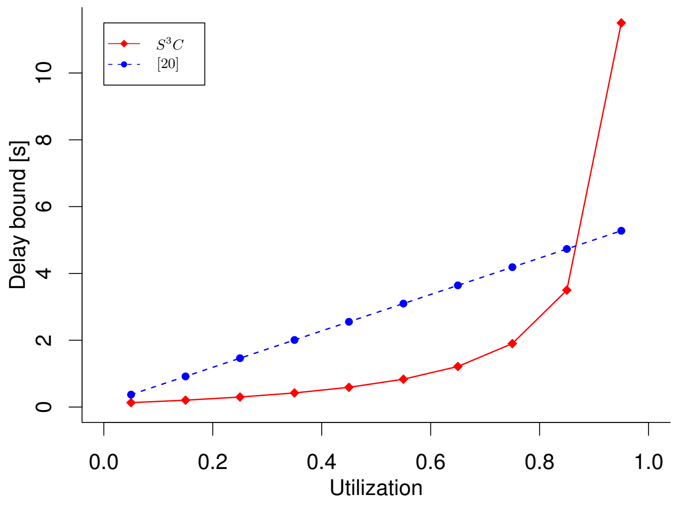

**Fig. 6.** Comparison of delay bounds from $S^3C$ and [20]. The parameter settings are: $R = 20$, $T = e^i = 0.01$,$\tau^i = 0$, $r = \rho R$, $b = \frac{r}{2}$ for a tandem of $n = 10$ servers, with $\rho$ being the utilization.

## 5 Are $S^3C$-based Delay Bounds Tight?

In this section, we discuss the question whether delay bounds for non-FIFO systems based on the $S^3C$ node model are tight. As we will see, they are *not tight* in general. We demonstrate this by devising an alternative approach, called the self-adversarial method, to compute a tight delay bound for non-FIFO systems based on a technique we introduced in [24]. It is shown that $S^3C$-based delay bounds can be larger than those derived by the self-adversarial method. We explore under what conditions this is actually the case.

It has to be noted that by using the self-adversarial method one departs from the elegant min-plus algebraic approach of network calculus, which results in a number of drawbacks. Maybe most important, in mixed FIFO and non-FIFO scenarios this prohibits to benefit from the knowledge about the FIFO behaviour of some of the nodes. As shown below, this drawback can be avoided by adhering to the algebraic approach of network calculus with $S^3C$. So, we are facing an interesting trade-off between algebraic characteristics and tightness of the bounds.

### 5.1 The Self-adversarial Method

In [24], we dealt with the problem of computing tight delay bounds for a network of arbitrary (non-FIFO) *aggregate multiplexers*. In that work we still made a FIFO-per-microflow assumption. Nevertheless, there is a way to exploit the method proposed in [24] for the problem at hand by transforming the single flow non-FIFO problem into an arbitrary aggregate multiplexing problem. More specifically, we split the original flow (with arrival curve $\alpha$) into two sub-flows: one with arrival curve $\alpha_1 = \gamma_{0,\epsilon}$ and the other one with arrival curve $\alpha_2 = \alpha - \gamma_{0,\epsilon}$ (where we assume that $\epsilon > 0$ is chosen such that $\alpha_2 \geq 0$). Both flows traverse the same servers as the original flow. This transformation is illustrated in Figure 7.

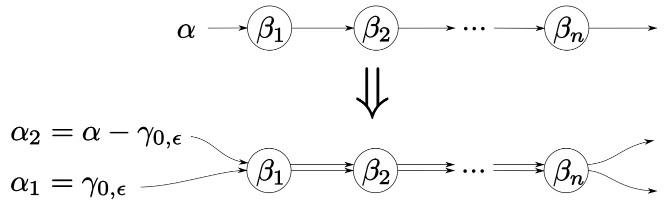
$\alpha$
$\beta_1$
$\beta_2$
$\beta_n$
$\alpha_2 = \alpha - \gamma_{0,\epsilon}$
$\alpha_1 = \gamma_{0,\epsilon}$

**Fig. 7.** Transformation of the single flow non-FIFO problem into an arbitrary aggregate multiplexing problem.

Now [24] allows us to find the maximum left-over end-to-end service curve under arbitrary multiplexing, i.e., under any possible interleaving of the two sub-flows. To that end, the problem is reformulated as an optimization problem which can be solved using standard methods. In [24], it is shown that this approach achieves tight delay bounds. So, in our case we can proceed with the following steps:

1. Computation of the left-over service curve for sub-flow 1 according to [24]: $\beta_1^{l.o.}$.
2. Computation of the delay bound for sub-flow 1: $d_1(\epsilon) = h\left(\alpha_1, \beta_1^{l.o.}\right)$.
3. Letting the delay bound for sub-flow 1 go to the limit: $d = \lim_{\epsilon \to 0} d_1(\epsilon)$.

What is effectively done here, is to assume that a part of the flow pretends to be an adversary to the other part of the flow when it comes to competition for the forwarding resource. This is why we call this the *self-adversarial* method. Taking this behaviour to the limit, i.e., making the adversary part as large as possible, gives us a real delay bound as experienced by a single bit. Note that the computation of the horizontal deviation in step 2 implicitly makes a FIFO assumption for sub-flow 1. Yet, in the limit this is not a problem, because a flow consisting of a single bit can obviously not experience any reordering any more.

### 5.2 $S^3C$-based vs. Self-Adversarial Method

First let us investigate by a simple example when the self-adversarial method can arrive at a better bound than the $S^3C$-based method. Assume a token-bucket arrival curve $\gamma_{r,b}$ for the flow under investigation, which traverses two servers providing *strict* rate-latency service curves $\beta_{R_iT_i}, i = 1, 2$. According to the $S^3C$-based method the delay bound then becomes:

$$d^{S^3C} = T_1 + T_2 + \frac{b + r\,(T_1 + T_2)}{\min\{R_1, R_2\} - r}.$$

For the self-adversarial method we first split the flow into two sub-flows: subflow 1 with $\gamma_{0,\epsilon}$ and subflow 2 with $\gamma_{r,b-\epsilon}$ as arrival curves. Proceeding with the steps as described in the previous section we obtain the following delay bound:

1. Computation of the left-over service curve for sub-flow 1:

$$\beta_1^{l.o.} = \beta_{\min\{R_1,R_2\}-r, T_1+T_2+\frac{b-\epsilon+rT_1}{\min\{R_1,R_2\}-r}+\frac{rT_2}{R_2-r}}.$$

2. Computation of the delay bound for sub-flow 1:

$$d_1\,(\epsilon) = T_1 + T_2 + \frac{b - \epsilon + rT_1}{\min\{R_1, R_2\} - r} + \frac{rT_2}{R_{2-r}}.$$

3. Letting the delay bound for sub-flow 1 go to the limit ($\epsilon \to 0$):

$$d^{SA} = T_1 + T_2 + \frac{b + rT_1}{\min\{R_1, R_2\} - r} + \frac{rT_2}{R_2 - r}.$$

A simple inspection shows that $d^{SA} < d^{S^3C}$ if $R_2 > R_1$. Hence, this demonstrates that the $S^3C$-based method is not tight under these circumstances. Similar problems with purely min-plus algebraic methods are reported and extensively discussed in [24]. These problems are inherent in using the min-plus algebraic approach. In particular, by the application of a min-plus convolution the knowledge on the order of servers is lost. Yet, this order is crucial to derive tight delay bounds.

So, with respect to the tightness of the computed bounds, the self-adversarial method is superior to the $S^3C$-based method. However, there are also a number of drawbacks for the self-adversarial method:

– The computational effort for the self-adversarial method can become very high. In particular, if arrival and service curves are piecewise linear functions then a set of optimization problem needs to be solved first before the final left-over service curve can be constructed. The cardinality of that set grows exponentially in the number of nodes traversed and may quickly become prohibitive. For details see [23]. In contrast, the $S^3C$-based method is computationally very cheap.

- The applicability of the self-adversarial method is restricted to $S^2C$ servers. This requirement is crucial for setting up the optimization problem in [24] and a relaxation towards only assuming $S^3C$ seems infeasible. The scope of the $S^3C$-based method is also larger with respect to the shape of arrival and service curves that may be used. According to [24], the self-adversarial method can only be applied to piecewise-linear concave arrival and convex service curves. Such a restriction does not apply for the $S^3C$-based method.
- The self-adversarial method is essentially a computational method rather than an analytical method yielding closed-form expressions. Thus, if one is interested in deriving equations for the bounds which may then be used further on, the self-adversarial method might be difficult to apply whereas the $S^3C$-based method can usually provide closed-form expressions.
- As the last, but practically perhaps most important drawback, we note that the self-adversarial method can deal with scenarios where we face a mixture of FIFO and non-FIFO servers only by assuming non-FIFO behaviour of all servers. In contrast, with the $S^3C$-based method, we show in Subsection 5.3 how the knowledge of some servers being FIFO can be leveraged to provide better delay bounds.

Besides these drawbacks of the self-adversarial method we also want to make the point that often the $S^3C$-based method computes the same delay bound as the self-adversarial method or stays at least very close to it. For all cases in which subsequent servers are not faster than their predecessors, the $S^3C$-based delay bound is the same as the one computed by the self-adversarial method. We now investigate the critical case where subsequent servers become faster and faster by some numerical experiments in more detail:

In particular, we assume for a tandem of $n$ rate-latency servers that each server's rate is increased by a constant amount $\Delta$ over its predecessors rate, i.e. it applies that $R_k = R_{k-1} + \Delta, k = 2, \ldots, n$. This constitutes a very bad condition for the $S^3C$-based method as the min-plus convolution swallows the knowledge of these rate increases and consequently accounts for the burstiness increases with the overall minimum rate instead of the correct minimum rates of the servers that actually "see" the respective burstiness increases. By raising the rate increase $\Delta$ we can worsen the situation for the $S^3C$-based method.

Figure 8 shows a comparison between the results of the self-adversarial method and $S^3C$ for varying utilizations $\rho \in [0.05; 0.95]$, with $n = 10$, $R_1 = 20$, $\Delta = 1$, $T = 0.01$, and an arrival curve $\alpha = \gamma_{\rho R, \rho \frac{R}{2}}$. It is clear that both methods deteriorate at high utilizations; however, the difference between the delay bounds is very small.

In a further experiment, we investigate the dependence of the delay difference on the rate increase $\Delta$. Figure 9 demonstrates the behavior of the delay bound difference (in % of the delay bound for the self-adversarial method) for several rate increase values from 0.1 to 100. The fixed parameters are the same as in the previous experiment; also shown is the analytical limit of the delay bound difference. It can be seen that the difference is limited to well under 20%, even under extreme rate increases and a high bottleneck utilization.

For the interested reader, the calculation of the limiting behavior for the delay bound ratio $\rho$ is as follows:

$$\begin{aligned}\rho(\Delta) &= \frac{d^{S^3C}_{\beta,n}(\Delta)}{d^{SA}_{\beta,n}(\Delta)} \\ &= \frac{b + nT \cdot \min\{R_i\}}{\min\{R_i\} - r} \cdot \frac{1}{\frac{b}{R_1 - r} + nT + rT\sum_{i=1}^{n}\frac{1}{R_i - r}}\end{aligned}$$

Using that

$$R_i = R_1 + (i-1)\Delta$$

and

$$\begin{aligned}\lim_{\Delta\to\infty} \min\{R_i\} &= R_1, \\ \lim_{\Delta\to\infty} \sum_{i=1}^{n}\left(\frac{1}{R_i - r}\right) &= \frac{1}{R_1 - r},\end{aligned}$$

the limit can be calculated as

$$\lim_{\Delta\to\infty} \rho(\delta) = \frac{b + nR_1T}{b + rT + nT(R_1 - r)}.$$

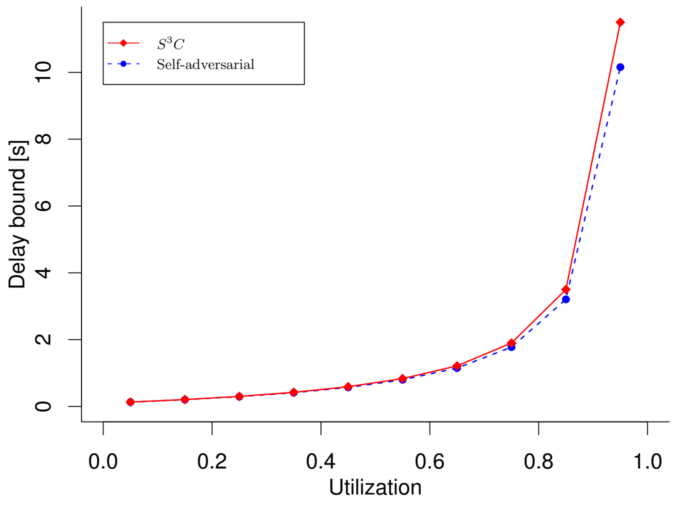

**Fig. 8.** Investigating the untightness of the $S^3C$-based delay bound.

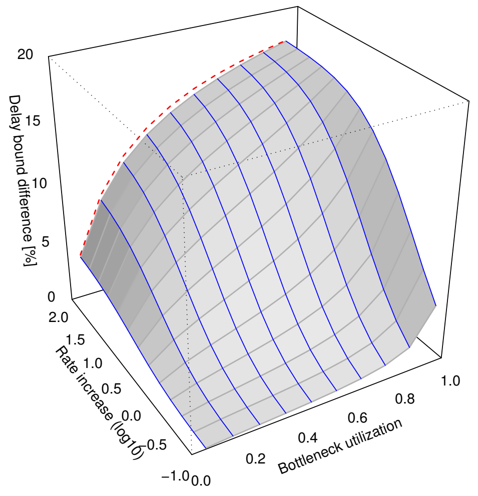

**Fig. 9.** Investigating the untightness of the $S^3C$-based delay bound for various rate increments. The rate increase is on a logarithmic scale, for any axis value $y$, the plot shows the value for a rate increase of $10^y$. The dotted line in the rear plane shows the upper limit for the delay bound as the rate increase $\Delta$ approaches infinity.

## 5.3 Mixed FIFO and non-FIFO Scenarios

The goal in mixed FIFO and non-FIFO scenarios should be to exploit the knowledge about some nodes being FIFO. Yet, at the same time concatenation should be used as much as possible. Therefore, the idea is to build two sections of the network one of which is purely FIFO and the other one purely non-FIFO. These two sections can then be analyzed separately with respect to the delay bound: for the FIFO section, the horizontal deviation can be used, whereas for the non-FIFO section, the intersection point must be used. However, in general $S^3C$ elements do not commute with each other due to the fact that maximum dwell periods may become different under commutation, i.e., in general for a tandem of two systems $D_{1,2}(t) \neq D_{2,1}(t)$ (see also Lemma 1).

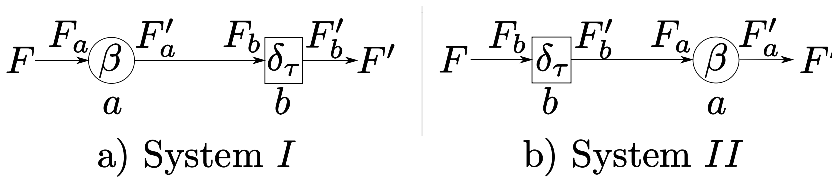

**Fig. 10.** Reordering of FIFO and non-FIFO elements. Subfigure a) shows an $S^3C$ node followed by a variable latency node. Subfigure b) shows the situation with switched service elements.

A practically important case of a mixed scenario is that we have a mixture of non-FIFO variable latency elements and FIFO $S^3C$ elements. For this case we have an interesting result on how these elements commute with each other:

**Theorem 7.** *(Shifting Variable Latency Elements ahead of $S^3C$ Nodes) Let us assume a system $I$ as illustrated in Figure 10 a), i.e. a node $a$ providing a $S^3C$ $\beta$ followed by a variable latency node $b$ providing a $S^3C$ $\delta_\tau$. System $I$'s maximum dwell period at time $t$ shall be denoted as $D_{a,b}(t)$. If we now alter the order of nodes $a$ and $b$, i.e. we switch to system $II$ (illustrated in Figure 10 b)) with maximum dwell period $D_{b,a}(t)$, then it applies that*

$$\max_{t\geq 0}\{D_{b,a}(t)\} \geq \max_{t\geq 0}\{D_{a,b}(t)\}.$$

*Remark 2.* Theorem 7 implies that it is always possible to shift a variable latency element ahead of an $S^3C$ node as this only worsens the largest maximum dwell period and thus the real delay bound. Thus shifting never compromises the bounds, although it may loosen them. For a fixed latency element, i.e., a node providing $\delta_\tau$ as minimum and maximum $S^3C$, it can be established that $D_{a,b}(t) = D_{b,a}(t)$ and thus any movement preserves the worst-case bounds.

*Proof.* In the proof we use the notation as illustrated in Figure 10 and further use the system descriptors $I, II$ as superscripts to avoid ambiguity for quantities that play a role in both systems where it seems appropriate.

Assume the largest maximum dwell period in system $II$ is taken on at time $t_{max}$, i.e.

$$D_{a,b}(t_{max}) = \max_{t\geq 0}\{D_{a,b}(t)\}.$$

Surely, $D_b(t_{max}) = \tau$, since otherwise the maximum dwell period could easily be prolonged. Hence, according to Lemma 1 we obtain

$$D_{a,b}(t_{max}) = \tau + D_a^I(t_{max} - \tau)$$

If we now turn to system $II$, we know that the input to node $a$ satisfies the following:

$$F_a(t) = F_b'(t) \geq F_b(t-\tau) = F(t-\tau).$$

Now, one way to do the scheduling in node $b$ for system $II$ is to delay every bit exactly by $\tau$ time units, i.e. we effectively set $F_b'(t) = F_b(t-\tau)$, which in turn means that

$$F_a^{II}(t) = F(t-\tau) = F_a^I(t-\tau).$$

So, under this fixed latency scheduling at node $b$, node $a$ in system $II$ sees exactly the same input as node $a$ in system $I$ shifted by $\tau$ time units. This then implies that

$$D_a^{II}(t) = D_a^I(t-\tau),$$

and thus (using the fact that, if node $b$ has work at time $t$ then $D_b(t) = \tau$, as well as Lemma 1)

$$\begin{aligned} D_{a,b}(t_{max}) &= \tau + D_a^I(t_{max} - \tau) \\ &= \tau + D_a^{II}(t_{max}) \\ &= D_a^{II}(t_{max}) + D_b\left(t_{max} - D_a^{II}(t_{max})\right) \\ &= D_{b,a}(t_{max}). \end{aligned}$$

So, under the assumption of node $b$ providing a fixed latency, which represents one of its possible scheduling options, we have created a schedule for system $II$ that achieves the largest possible maximum dwell period in system $I$. In fact, by using the higher degree of freedom we have for scheduling at node $b$, i.e. scheduling some data ahead of its deadline, we can worsen the situation for node $a$ in system $II$ by providing more data items for reordering at node $a$. Such a higher reordering potential can achieve a higher maximum dwell period, which justifies the inequation in the statement of the theorem.

A practically very relevant incarnation of the special case dealt with in Theorem 7, and also presented in [20], is to model a router as consisting of (1) a non-FIFO switching fabric that is traversed with a certain maximum latency $\tau$, i.e. it provides $\delta_\tau$ as $S^3C$, and (2) an output link which provides a FIFO rate-latency $S^3C$ $\beta_{R,T}$. Assuming we have a tandem of $n$ such routers and an arrival curve $\gamma_{r,b}$ for the flow under investigation, there are basically two ways to derive a delay bound:

1. Ignore the FIFO behaviour of the output link and do a pure non-FIFO analysis, which results in the following delay bound
$$d^{non-FIFO} = n(T+\tau) + \frac{b + rn(T+\tau)}{R-r}.$$
2. Use Theorem 7 to build two subsections of the tandem by moving all non-FIFO latency elements in front of the FIFO elements and analyse the two subsections separately and add their delay bounds. Doing so, we arrive at the following delay bound:
$$d^{mixed-mode} = n(T+\tau) + \frac{b + rn\tau}{R}.$$

Note that for the pure non-FIFO option it does not matter whether we apply the self-adversarial or the $S^3C$- based method since under the given assumptions they compute the same bound. Clearly, the option of building two subsections is superior since the second term is strictly smaller for $r > 0$. In Figure 11, this is shown to be very significant for higher utilizations. Furthermore, the results for an additive analysis (which can also make use of the FIFO behaviour of the output links) are shown. For the interest reader, the delay bound for an additive analysis is calculated as

$$d^{additive} = n\left(T + \tau + \frac{b}{R}\right) + \frac{nr}{2R}\left((n-1)T + (n+1)\tau\right).$$

For a utilization of 95%, the additive method even outperforms the pure non-FIFO analysis, which further demonstrates the benefit of exploiting the knowledge about the FIFO elements.

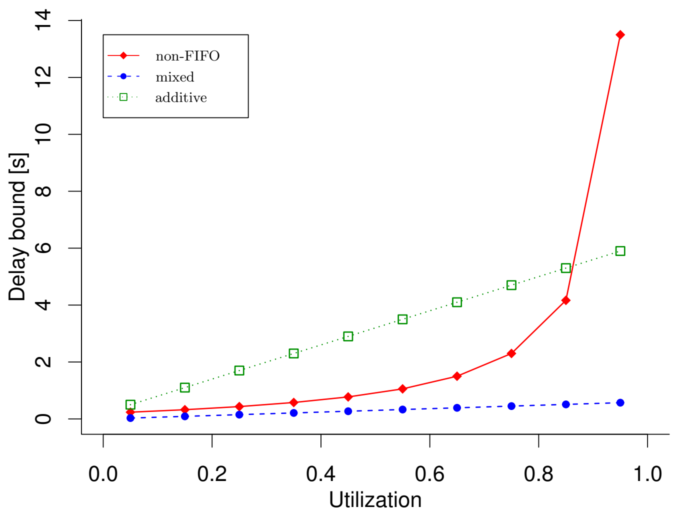

**Fig. 11.** $S^3C$-based mixed mode analysis vs. pure non-FIFO analysis. The fixed parameters are $n = 10$, $R = 20$, $T = \tau = 0.01$, $r = \rho R$, $b = \frac{r}{2}$, with $\rho$ being the utilization.

## 6 Conclusion

In this report, it was our goal to extend the scope of network calculus towards non-FIFO systems, as non-FIFO behaviour is a reality in many networking scenarios. It turned out that existing service curve definitions are not satisfying under non-FIFO scheduling: they are either to loose to enable any bounding or too strict to allow for an efficient end-to-end analysis. Therefore, we introduced a new service curve definition, $S^3C$, which allows to bound the delay and at the same time enables an end-to-end analysis. By numerical examples, we showed that the new analysis based on $S^3C$ is far superior to existing methods. $S^3C$ allows to recover the pay bursts only once phenomenon for non-FIFO systems, which had been disputed to be valid under non-FIFO scheduling in literature. While the network calculus extension to non-FIFO systems runs smoothly up to that point, the devil is in the tightness of the new bounds. By devising an alternative method we show that the new bounds can be conservative. Investigating this further shows, however, that they stay very close to the tight bound. Furthermore, adhering to the min-plus algebraic nature of network calculus avoids several drawbacks of the alternative method. In particular, in mixed FIFO and non-FIFO scenarios this pays off in significantly better delay bounds.

## Acknowledgements

We are very grateful to Markus Fidler for insightful discussions and comments on an early version of this report.

## References

1. R. Agrawal, R. L. Cruz, C. Okino, and R. Rajan. Performance bounds for flow control protocols. *IEEE/ACM Transactions on Networking*, 7(3):310–323, June 1999.
2. F. Baccelli, G. Cohen, G. J. Olsder, and J.-P. Quadrat. *Synchronization and Linearity: An Algebra for Discrete Event Systems.* Probability and Mathematical Statistics. John Wiley & Sons Ltd., 1992.
3. J. C. R. Bennett, C. Partridge, and N. Shectman. Packet reordering is not pathological network behavior. *IEEE/ACM Trans. Netw.*, 7(6):789–798, 1999.
4. J. M. Blanquer and B. Özden. Fair queuing for aggregated multiple links. *SIGCOMM Comput. Commun. Rev.*, 31(4):189–197, 2001.
5. S. Chakraborty, S. Kuenzli, L. Thiele, A. Herkersdorf, and P. Sagmeister. Performance evaluation of network processor architectures: Combining simulation with analytical estimation. *Computer Networks*, 42(5):641–665, 2003.
6. C.-S. Chang. On deterministic traffic regulation and service guarantees: A systematic approach by filtering. *IEEE Transactions on Information Theory*, 44(3):1097–1110, May 1998.
7. C.-S. Chang. *Performance Guarantees in Communication Networks.* Telecommunication Networks and Computer Systems. Springer-Verlag, 2000.
8. F. Ciucu, A. Burchard, and J. Liebeherr. A network service curve approach for the stochastic analysis of networks. In *Proc. ACM SIGMETRICS*, pages 279–290, June 2005.
9. R. L. Cruz. A calculus for network delay, Part I: Network elements in isolation. *IEEE Transactions on Information Theory*, 37(1):114–131, January 1991.
10. R. L. Cruz. A calculus for network delay, Part II: Network analysis. *IEEE Transactions on Information Theory*, 37(1):132–141, January 1991.
11. R. L. Cruz. Quality of service guarantees in virtual circuit switched networks. *IEEE Journal on Selected Areas in Communications*, 13(6):1048–1056, August 1995.
12. P. Goyal, S. S. Lam, and H. M. Vin. Determining end-to-end delay bounds in heterogeneous networks. *Multimedia Syst.*, 5(3):157–163, 1997.
13. S. Jaiswal, G. Iannaccone, C. Diot, J. Kurose, and D. Towsley. Measurement and classification of out-of-sequence packets in a tier-1 IP backbone. *IEEE/ACM Trans. Netw.*, 15(1):54–66, 2007.
14. H. Kim and J.C. Hou. Network calculus based simulation: theorems, implementation, and evaluation. In *Proc. IEEE INFOCOM*, March 2004.
15. A. Koubaa, M. Alves, and E. Tovar. Modeling and worst-case dimensioning of cluster-tree wireless sensor networks. In *Proc. 27th IEEE International Real-Time Systems Symposium (RTSS'06)*, pages 412–421, Rio de Janeiro, Brazil, 2006. IEEE Computer Society.

16. J.-Y. Le Boudec. Application of network calculus to guaranteed service networks. *IEEE Transactions on Information Theory*, 44(3):1087–1096, May 1998.
17. J.-Y. Le Boudec and A. Charny. Packet scale rate guarantee for non-fifo nodes. In *Proc. IEEE INFOCOM*, pages 23–26, June 2002.
18. J.-Y. Le Boudec and P. Thiran. *Network Calculus A Theory of Deterministic Queuing Systems for the Internet*. Number 2050 in Lecture Notes in Computer Science. Springer-Verlag, Berlin, Germany, 2001.
19. A. K. Parekh and R. G. Gallager. A generalized processor sharing approach to flow control in integrated services networks: The single-node case. *IEEE/ACM Transactions on Networking*, 1(3):344–357, June 1993.
20. G. Rizzo and J.-Y. Le Boudec. Pay bursts only once does not hold for non-fifo guaranteed rate nodes. *Performance Evaluation*, 62(1-4):366–381, 2005.
21. H. Sariowan, R. L. Cruz, and G. C. Polyzos. Scheduling for quality of service guarantees via service curves. In *Proc. IEEE ICCCN*, pages 512–520, September 1995.
22. J. Schmitt and U. Roedig. Sensor network calculus - a framework for worst case analysis. In *Proc. Distributed Computing on Sensor Systems (DCOSS)*, pages 141–154, June 2005.
23. J. Schmitt, F. Zdarsky, and M. Fidler. Delay bounds under arbitrary multiplexing. Technical Report 360/07, University of Kaiserslautern, Germany, July 2007.
24. J. Schmitt, F. Zdarsky, and M. Fidler. Delay bounds under arbitrary aggregate multiplexing: When network calculus leaves you in the lurch... In *Proc. IEEE INFOCOM*, April 2008.
25. T. Skeie, S. Johannessen, and O. Holmeide. Timeliness of real-time IP communication in switched industrial ethernet networks. *IEEE Transactions on Industrial Informatics*, 2(1):25–39, February 2006.