Pay Bursts Only Once Holds for (Some) Non-FIFO Systems

Jens B. Schmitt, Nicos Gollan, Steffen Bondorf, Ivan Martinovic

Technical Report No. 378/10

July 2010

Abstract. Non-FIFO processing of flows by network nodes is not a rare phenomenon. Unfortunately, the state-of-the-art analytical tool for the computation of performance bounds in packet-switched networks, network calculus, cannot deal well with non-FIFO systems. The problem lies in its conventional service curve definitions. Either the definition is too strict to allow for a concatenation and consequent beneficial end-to-end analysis, or it is too loose and results in infinite delay bounds. Hence, in this report, we propose a new service curve definition and demonstrate its strength with respect to achieving both finite delay bounds and a concatenation of systems resulting in a favourable end-to-end delay analysis. In particular, we show that the celebrated pay bursts only once phenomenon is retained even without any assumptions on the processing order of packets. This seems to contradict previous work [24]; the reasons for this are discussed.

Keywords: Performance Evaluation, Performance Bounds, Network Calculus, Non-FIFO, Concatenation.

1 Introduction

1.1 Motivation

In the recent past, network calculus [10, 22] has shown promise as an alternative methodology, besides classical queueing theory, for the performance analysis of packet-switched networks. It is being used as a basic tool for attacking several important network engineering problems: most prominently in the Internet's Quality of Service proposals IntServ and DiffServ, but recently also in other environments like, e.g., wireless sensor networks [19, 26], switched Ethernets [28], Systems-on-Chip (SoC) [8], or even to speed-up simulations [18], to name a few. Unfortunately, it comes up short in a basic aspect: its delay bound calculation depends on a strict FIFO processing of work units within the flow under analysis. This issue about how to deal with a possible *non-FIFO processing* within a flow is largely unexplored (and sometimes may be even overseen in applications). In the next subsection, we provide an overview of the rather little previous work we could find on this. In contrast, *non-FIFO multiplexing* between flows has been intensive subject to previous work (see for example [27, 7] and references therein for recent contributions on so-called arbitrary or general multiplexing). In this report, we are concerned with the scheduling within a single flow under analysis, so we deal with the questions *when* work units are processed by a node and *in which order*. The first question about how a node provides its capacity to a flow is flexibly anwered by the network calculus concept of service curves, the second question has so far always been answered rigidly by assuming a FIFO processing order. The goal of this report is to provide a more flexible answer to this. In the following, we provide some arguments why besides being of theoretical interest this issue should be addressed.

Assuming that the work units of a flow under analysis are processed in FIFO order constitutes a logical break for a worst-case methodology in a certain sense as we discuss now. Assume a flow traverses a system resulting in a certain output process. The *real delay*[1] for a work unit input at time t and output at time t' is simply defined as

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

Any processing order other than FIFO results in an increase of the worst-case real delay. This can be seen by the following argument: Assume at time t_0 a work unit which experiences the worst-case real delay is input to a FIFO system. Now assume we can change the processing order of work units. If the work unit is further delayed by scheduling work units that arrived later, then certainly the real delay of that work unit under the new processing order will be worse. Processing that work unit earlier will make its new real delay $rd'(t_0)$ smaller, yet, the work unit which was processed just ahead of the above work unit is now leaving the system when the above work unit would have left; however, that work unit has arrived at time $t_1 \leq t_0$ such that the new real delay of that work

[1] *The word* real *is chosen for the purpose of contrasting it to the* virtual *delay, later on defined as delay under FIFO processing of a flow.*

unit $rd'(t_1)$ is higher than or equal to the one from the FIFO worst-case work unit, i.e., $rd(t_0) \leq rd'(t_1)$. So, in this sense FIFO can be viewed as a best-case assumption (with respect to the worst-case delay) on the processing order of the flow under analysis. Therefore, it can be seen as consistent with a worst-case methodology to release the FIFO processing assumption.

Furthermore, by providing the following list of real-world examples where non-FIFO behaviour is exhibited, we also want to stress the practical relevance of this work:

Packet Reordering: In several studies of Internet traffic it has been shown that packet reordering is a frequent event (see for example [4, 17]). According to these studies this is due to a growing amount of parallelism on a global (use of multiple paths or parallel links) as well as on a local (device) level. In particular, for scalability reasons high-speed routers often contain a complex multi-stage switching fabric which cannot ensure to preserve the order of arrivals at its output. This is due to a common design trade-off where FIFO service at the input queues is relaxed in order to avoid head-of-line blocking by choosing from a set of packets in the input queue (often some window-based scheme is used). Furthermore, the use of link aggregation, where multiple physical lines are aggregated into a single virtual link, may often lead to non-FIFO behavior [6].

Link-Level Retransmissions: Retransmission schemes that work on the link-level are a further source for non-FIFO behaviour. Especially in wireless networks such link-level retransmission schemes are used to avoid lengthy end-to-end retransmissions and mask the inherent failure characteristics of radio transmissions. To still provide a good throughput these retransmission schemes usually employ sliding window techniques which again may lead to reordering of packets on a flow level. Depending on a radio channel's characteristics the degree of reordering can vary, yet, most often, it is substantial.

Content-Dependent Packet Scheduling: As a last example, let us mention wireless sensor networks (WSN) in which packet scheduling decisions may be based on the contents of packets following a WSN-typical data-centric paradigm. Under such circumstances hardly anything may be assumed about the scheduling order, let alone FIFO behaviour.

So, from a methodological as well as an application perspective there is a clear motivation for an investigation on how network calculus can be extended towards an analysis without any FIFO assumptions. 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

There is surprisingly little existing work on the treatment of non-FIFO systems in the context of network calculus. Remarkably, in his pioneering paper [12], Cruz briefly showed how to derive a delay bound for a single work-conserving server under a general scheduling assumption (comprising any non-FIFO processing order) based on the observation that the maximum backlogged period can be bounded given that traffic is regulated. Similar results can also be found in [10]. Yet, the multiple node case as well as more general server models are not treated therein.

In [21], 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 the delay bound from the FIFO case still applies in the single node case while it does not in a specific two node case. They leave more general concatenation scenarios for further study.

In [27], the problem of computing tight delay bounds for a network of arbitrary (non-FIFO) *aggregate multiplexers* is addressed. The tightness of the bounding method is shown by sample path arguments. Yet, in contrast to the problem setting in this report, a FIFO assumption on the processing order within a flow is made and non-FIFO behaviour is only allowed between flows. Bouillard et al. recently provided more advanced and general results for the same setting in [7], still basing on FIFO processing per flow.

To the best of our knowledge, the only previous work that also tries to derive end-to-end delay bounds without any FIFO processing within the flow under analysis assumption was done by Rizzo and Le Boudec [24]. They investigate delay bounds for a special server model, non-FIFO guaranteed rate (GR) nodes, and show that a previously derived delay bound for GR nodes [15] 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". In contrast, we show that non-FIFO systems may still possess a concatenation property. This puzzling contradiction is discussed in detail in Section 6, since it is a subtle issue that needs to be resolved thoroughly.

2 Preliminaries on Network Calculus

Network calculus is a min-plus system theory for deterministic queuing systems, which builds on the calculus for network delay in [12], [13]. The important concept of service curve was introduced in [1, 9, 14, 20, 25]. 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 [11] and also referred to as pay bursts only once phenomenon in [22]. A detailed treatment of min-plus algebra and of network calculus can be found in [3] and [10, 22], respectively.

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

$$\mathcal{F} = \left\{ f : \mathbb{R}^+ \to \mathbb{N}_0, \forall t \geq s : f(t) \geq f(s), f(0) = 0 \right\}.$$

In particular, the input function $F(t)$ and the output function $F'(t)$, which cumulatively count the number of work units 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 *discrete* in space unless specified otherwise. We assume work units of equal size, which is convenient for some of the discussions with respect to the processing order of work units. Arbitrarily, we assume in- and output functions to be left-continuous. 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*

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

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

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 [22]. 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 [22], [10]. With these definitions, 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*

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

The virtual delay *for a work unit input at time t is defined as*

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

So, this is the point where a FIFO assumption "sneaks" in network calculus as far as delay is concerned, because $rd(t) = vd(t)$ for all t only under FIFO processing of the flow. We use the usual network calculus terminology of the so-called *virtual* delay in contrast to the *real* delay as defined above (see Section 1.1). Next, 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 β iff*

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

If F is left-continuous and β is continuous, this is equivalent to the following condition

$$\forall t \geq 0 : \exists s \leq t : F'(t) \geq F(s) + \beta(t-s).$$

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. Also, nodes operating under a delay-based scheduler and guaranteeing that a work unit arriving at any time t will leave the node at time $t' \leq t + T$ for some fixed $T > 0$, i.e. $\forall t \geq 0 : rd(t) \leq T$, are known to provide a service curve $\delta_T = \infty \cdot 1_{\{t>T\}}$. We also call these bounded latency nodes.

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 β. Assume a flow F traversing the system has an arrival curve α. Then we obtain the following performance bounds:*

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

Here, note again that the delay bound is only a *virtual* one, meaning that it is based on a FIFO assumption for the flow under analysis. 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 β_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 achieves tight performance bounds, which in general is not the case for an iterative per-node application of Theorem 1. In particular, a linear scaling of performance bounds in the number of traversed queues is achieved and also referred to as pay bursts only once phenomenon in [22].

3 Conventional Network Calculus And Non-FIFO Systems

In this section, we investigate how existing network calculus can cope with non-FIFO systems. The crucial aspect is the node model. We start with the typical service curve model as defined in the previous section and turn to other node models like strict and adaptive service curves, only to find out that all of them encounter problems under non-FIFO processing.

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 tempting 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. (SC Cannot Bound the Real Delay) 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 $\alpha = \gamma_{1,0}$. 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

$$\begin{aligned} F' &= \alpha \otimes \beta = \gamma_{1,0} \otimes \beta_{2,1} = \gamma_{1,0} \otimes \gamma_{2,0} \otimes \delta_1 \\ &= (\gamma_{1,0} \wedge \gamma_{2,0}) \otimes \delta_1 \leq \gamma_{1,0} \otimes \delta_1 < \gamma_{1,0} = F. \end{aligned}$$

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. Therefore, without any assumptions on the processing order a certain work unit can be kept forever in the system under these circumstances. Thus, the real delay of that work unit is unbounded. Note that using the standard FIFO processing assumption, we can of course bound the real delay of the system by $\forall t \geq 0 : rd(t) = vd(t) \leq \frac{3}{2}$.

From this example, we see that the SC property is too weak as a node model for analyzing non-FIFO systems. Therefore, it is sensible to look for more stringent node models as is done in the following subsection.

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

A number of systems provide more stringent service guarantees than captured by SC, fulfilling a so-called strict service curve [22] (also known as strong service curve [2] and related to the universal service curve concept in [23]).

Definition 5. *(Strict Service Curve – S^2C) Let $\beta \in \mathcal{F}$. System $\mathcal{S}$ offers a* strict *service curve β 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* $\forall s \in (t-u, t] : b(s) > 0$. *More formally,* $\mathcal{S}$ *offers a* strict *service curve* β *to a flow iff*

$$\forall t \geq 0 \wedge \forall u \geq 0 \wedge (\forall s \in (t-u, t] : b(s) > 0) :$$

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

Note that any node satisfying S^2C also satisfies SC, but not vice versa. For example, a bounded latency node does not provide its service curve δ_T as a *strict* service curve. In fact, in a continuous time and space model for in- and output functions, it does not provide any S^2C apart from the trivial case $\beta = 0$ [22] (for a discrete space model with unit packets, a (very low) strict service curve can be derived for a bounded latency node as $\beta_{1/T,T}$). On the other hand, there are many schedulers that offer strict service curves; for example, most of the generalized processor sharing-emulating schedulers (e.g., PGPS [23], WF^{2}Q [5], or round robin schedulers like SRR [16], to name a few), offer a strict service curve of the rate-latency type. More generally, all nodes providing some form of capacity guarantee can be captured by S^2C.

Now for bounding the real delay under S^2C: In fact, as was already shown by Cruz [12] (and can also be found in [10] (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 3. *(Real Delay Bound for Single S^2C Node) Consider a system $\mathcal{S}$ that offers a strict service curve β. Assume a flow F traversing the system has an arrival curve α. 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 3 together with the output bound from Theorem 1 (note that for the latter we only require SC). More specifically, if a tandem of n S^2C non-FIFO nodes, each providing a strict service curve $\beta_j, j = 1, \ldots, n$, is to be traversed by an α-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

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

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 [22]. The key to avoid this behaviour is to perform an end-to-end analysis based on a concatenation theorem. Yet, as is known and demonstrated 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 $\beta^i = \beta_{2,1}, i = 1, 2$, 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 $\beta^{1,2} = \beta^1 \otimes \beta^2 = \beta_{2,2}$.

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

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

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

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

$$F_2'(t) = \begin{cases} 0 & 0 \leq t \leq 2 \\ 1 & 2 < t \leq 3 \end{cases}.$$

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, we obtain

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

which shows that $\beta^{1,2}$ is not S^2C for the composite system (while, of course, being SC for it, but this does not help for bounding the real delay).

The problem is due to the fact that there may be a system backlogged period but some of the subsystems are not backlogged. 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 non-FIFO processing, since it possesses no concatenation property for the multiple node case. Now, since SC is too loose and S^2C too strict, it is sensible to look for some intermediate node model.

3.3 Using Adaptive Service Guarantees for Non-FIFO Systems

The so-called adaptive service guarantee, introduced in the context of window flow control systems [2], is a node model that is known to be falling in between SC and S^2C in a certain sense.

Definition 6. *(Adaptive Service Guarantee – ASG) 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 an adaptive service guarantee $\left(\tilde{\beta}, \beta\right)$ iff $\forall t \geq 0$:*

$$F'(t) \geq \sup_{s<t} \left\{ \left(F'(s) + \tilde{\beta}(t-s) \right) \wedge \inf_{u \in [s,t]} \{ F(u) + \beta(t-u) \} \right\} \tag{2}$$

ASG enables to bound the *virtual* delay based on the backlog of a flow (delay-from-backlog bound), something that can also be done under S^2C but not under SC. In contrast to S^2C, however, ASG has a concatenation property (albeit a bit more complex). So, this makes it interesting because in this sense it lies in between SC and S^2C. Yet, unfortunately, as the following example (actually almost the same as in the SC case) shows it cannot bound the real delay without assuming FIFO processing of the flow under analysis.

Example 3. (*ASG* Cannot Bound the Real Delay) Assume a single node system $\mathcal{S}$ which offers an ASG with $\tilde{\beta} = \beta = \beta_{2,1}$ to a flow F which is constrained by an affine arrival curve $\alpha = \gamma_{1,0}$. Now assume the flow to be greedy, that means $F = \alpha$ and the server to be lazy, that means it meets (2) with equality. Furthermore, we assume a discrete-time model with time indexed by $t = 0, 1, 2, \ldots$ Under these settings, it can be shown that $\forall t \geq 0$

$$F'(t) = [t-1]^+, \tag{3}$$

and thus

$$F'(t) = [t-1]^+ = \left(\gamma_{1,0} \otimes \delta_1\right)(t) < \gamma_{1,0}(t) = \alpha(t) = F(t).$$

This means the system remains backlogged at all times. Therefore, without any assumptions on the processing order a certain work unit can be kept forever in the system, also under ASG. Thus, the real delay of that work unit is unbounded.

The derivation of Equation (3) can be done via induction on t:

$\underline{t = 0 :}$

$$F'(0) = 0 = [0-1]^+ .$$

$\underline{t \to t+1 :}$

$$\begin{aligned} F'(t+1) &= \sup_{s<t+1} \{ (F'(s) + \beta_{2,1}(t+1-s)) \\ &\qquad\qquad \wedge \inf_{u \in \{s,\ldots,t+1\}} \{ \gamma_{1,0}(u) + \beta_{2,1}(t+1-u) \} \Big\} \\ &= \sup_{s<t+1} \left\{ \left([s-1]^+ + 2\,[t-s]^+ \right) \wedge \inf_{u \in \{s,\ldots,t+1\}} \left\{ u + 2\,[t-u]^+ \right\} \right\} \\ &= \sup_{s<t+1} \left\{ \left([s-1]^+ + 2\,[t-s]^+ \right) \wedge t \right\} \\ &= \sup_{s<t+1} \left\{ [s-1]^+ + 2\,[t-s]^+ \right\} \wedge t \\ &= 2t \wedge t = t = [t+1-1]^+ . \end{aligned}$$

So, ASG suffers from the same deficiency as SC with respect to bounding the real delay, though it is a stronger guarantee but in a sense that does not help here.

3.4 Discussion and Some Remarks

To sum up, we started with the service curve (SC) definition as the standard network calculus node model, yet found it too weak to bound the real delay under non-FIFO processing. Next, we tried a more stringent node model, the strict service curve (S^2C). Actually, under S^2C a single-node bound on the real delay can be given. However, S^2C has no concatenation property, thus impeding a true end-to-end delay analysis. Hence, we were looking for a node model falling in between SC and S^2C. Adaptive service guarantees (ASG) are lying in this intermediate area. They have some properties similar to strict service curves but still possess a concatenation property. Unfortunately, as we demonstrated they do not provide a bound on the real delay. So they are too weak, again.

All these variants of node models are based on min-plus algebraic definitions, which effectively means nodes are characterized by the amount of work they guarantee to do in certain intervals (the intervals for which this is specified differ). In fact, there is a "dual world" of max-plus algebraic definitions for node models. Known models of this category are so-called guaranteed rate (GR) nodes [15] and packet-scale rate guarantees ($PSRG$) [22]. These work by guaranteeing certain delivery deadlines to work units, which they are assigned when arriving at a node. Since these deadlines are in absolute time, these node models can always provide a bound on the real delay (given a finite input at each point in time). Their concatenation properties are derived based on equivalence theorems with min-plus algebraic counterparts. In particular, GR is equivalent with SC (modulo packetization) and $PSRG$ is equivalent with ASG (modulo packetization) (see chapter 2 and 7 of [22] for details). However, the crucial point here is that these equivalences are only applicable for FIFO systems and are actually shown not to be applicable for concatenation of non-FIFO systems in [22, 21]. Thus, GR and $PSRG$ have no known concatenation properties under non-FIFO processing. In fact, the delay bound derived for non-FIFO GR nodes in [24] is an additive bound based on iterative application of per-node delay bounds. This is discussed in more detail in Section 6.

The bottom line of this discussion is that we need a new node model: it should

1. allow for calculating a bound on the real delay without FIFO processing order assumptions, and, yet, also
2. have a concatenation property, and thus be min-plus based, in order to avoid loose additive bounds.

4 Sufficiently Strict Service Curves

In this section, we introduce a new node model called sufficiently strict service curves (S^3C). This node model allows to efficiently bound the real delay over a tandem of non-FIFO systems using a concatenation result. We show that there are actually systems conforming to this new node model and provide numerical examples for the superiority of S^3C-based delay bounds over conventionally derived delay bounds (based on S^2C).

4.1 The New Service Curve Definition

Central to the new service curve definition is the notion of a *maximum achievable dwell period* (MADP).

Definition 7. *(Maximum Achievable Dwell Period) Given a system $\mathcal{S}$, the maximum achievable dwell period at time t, denoted as $D(t)$, is the length of the interval $[t_o(t), t]$, i.e., $D(t) = t - t_o(t)$, where $t_o(t)$ is the arrival time of the oldest work unit in the system at time t under* all possible processing orders. *If the system is empty at time t, then by definition $t_o(t) = t$ and $D(t) = 0$.*

Some comments on this definition of the MADP are appropriate:

- A tacit assumption is that changing the processing order of work units does not affect the output function F' of the system. This is true under identically sized work units (as assumed in this report) and a continuous data model with infinitesimally small work units. The definition would require more elaboration if discrete, but variable size work units were considered, or if the output function of a system depended in some way on the order in which work units are processed. The latter is uncommon in networks, the former we leave for future study.
- The term "all possible processing orders" in the definition may warrant some further clarification. For min-plus algebraic node models, which only provide guarantees on the amount of work done over specific intervals, possible processing orders amount to all permutations of work units that ever share a buffer. More formally, the set "all possible processing orders" can be generated as the closure of all permutations of work units that share a buffer starting from the FIFO processing order. Fortunately, we do not need to deal explicitly with that very large set further on.

An example for an MADP should be illustrative: For a work-conserving constant rate server with a single buffer, the MADP at time t equals the backlogged period at time t; consequently, $t_o(t)$ equals the start of the last backlogged period. The processing order that achieves the MADP for such a simple buffer is LIFO. However, the MADP of a tandem of work-conserving servers can be shorter than the backlogged period of the overall system, which is anyway generally unbounded (see Section 3.2, Example 2).

We remark that the MADP is always within a backlogged period of the system, even if the system is not employing the processing order leading to the MADP, since the output function F' remains unchanged.

Now using the notion of the MADP, the new service curve definition can be introduced:

Definition 8. *(Sufficiently Strict Service Curve - 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) iff*

$$\forall t \geq 0 : F'(t) \geq F(t - D(t)) + \beta(D(t)).$$

Note that S^3C implies SC.

4.2 Properties of S^3C

Now we show that the S^3C definition achieves for non-FIFO systems the two attractive features known from conventional network calculus with FIFO systems: a bound on the delay, yet now on the real instead of the virtual delay, *and* a concatenation property. These two properties of S^3C allow to recover the celebrated pay bursts only once phenomenon in the non-FIFO case.

S^3C Can Bound the Real Delay First we show that S^3C allows to bound the real delay:

Theorem 4. *(Real Delay Bound for an S^3C System) Consider a system $\mathcal{S}$ that offers a sufficiently strict service curve β. Assume a flow F traversing the system has an arrival curve α. Then we obtain the following bound on the real delay:*

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

Proof. As above we denote the MADP at time t by $D(t)$ and the arrival time of the oldest possible work unit (corresponding to $D(t)$) by $t_o(t)$. We can make the following observation for the backlog of the system at time t:

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

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

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

This now allows to bound the MADP at time t, by the following observation:

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

Since the bound is independent of t, it applies for all t and, furthermore, since a bound over all MADPs is also a bound for all real delays, the proof is completed. □

So, like S^2C, S^3C is strong enough to enable a finite bound on the real delay, in contrast to SC and ASG.

S^3C Has a Concatenation Property Now we show that S^3C also has a concatenation property. The concatenation property is not as simple to achieve as for SC and some restrictions apply. To make these restrictions clearly visible and demonstrate where they are required, we develop the concatenation result in a modular fashion, from a set of lemmas, which eventually result in a concatenation theorem for S^3C.

We use the following notation, assuming a tandem $\mathcal{T}$ of n systems $\mathcal{S}_1, \ldots, \mathcal{S}_n$:

- $t_o^{(i)}(t)$: the (local) arrival time of the oldest possible work unit at time t for system $\mathcal{S}_i$,
- $t_o^{(1,\ldots,n)}(t)$: the arrival time (at $\mathcal{S}_1$) of the oldest possible work unit at time time t for the tandem $\mathcal{T}$,
- $T_o(t) := t_o^{(1)}\left(t_o^{(2)}\left(\cdots t_o^{(n)}(t)\right)\right)$.

The first lemma states a basic concatenation rule for a tandem of S^3C nodes:

Lemma 1. *Consider a flow with input function F that traverses the tandem $\mathcal{T}$. Assume that each system $\mathcal{S}_i$ offers a sufficiently strict service curve β_i, $i = 1, \ldots, n$ to the flow. Denoting the output function of $\mathcal{T}$ as F', we obtain*

$$F'(t) \geq F(T_o(t)) + \left(\bigotimes_{i=1}^{n} \beta_i\right)(t - T_o(t)).$$

Proof. The proof of Lemma 1 is relatively straightforward, though notationally somewhat cumbersome. The following further notation is used

- $F^{(i)}$, $i = 1, \ldots, n$ denotes the output function from system $\mathcal{S}_i$ (and the input function for system $\mathcal{S}_{i+1}$, respectively), we set $F^{(0)} := F$ as the input function to the tandem $\mathcal{T}$; note that $F^{(n)} = F'$.
- $T_o^{(i)}(t) := t_o^{(i+1)}\left(t_o^{(i+2)}\left(\cdots t_o^{(n)}(t)\right)\right)$, $i = 0, \ldots, n-1$ and $T_o^{(n)}(t) := t$; note that $T_o^{(0)}(t) = T_o(t)$.

From the S^3C property for each of the systems $\mathcal{S}_i$ we obtain $\forall n \geq i \geq 1$:

$$F^{(i)}\left(T_o^{(i)}(t)\right) - F^{(i-1)}\left(T_o^{(i-1)}(t)\right) \geq \beta_i\left(T_o^{(i)}(t) - T_o^{(i-1)}(t)\right)$$

Summing all these inequations yields

$$\begin{aligned} F^{(n)}\left(T_o^{(n)}(t)\right) - F^{(0)}\left(T_o^{(0)}(t)\right) &\geq \sum_{i=1}^{n} \beta_i\left(T_o^{(i)}(t) - T_o^{(i-1)}(t)\right) \\ &\geq \left(\bigotimes_{i=1}^{n} \beta_i\right)\left(T_o^{(n)}(t) - T_o^{(0)}(t)\right) \\ &= \left(\bigotimes_{i=1}^{n} \beta_i\right)(t - T_o(t)) \end{aligned}$$

Noting that $F^{(n)}\left(T_o^{(n)}(t)\right) = F'(t)$ and $F^{(0)}\left(T_o^{(0)}(t)\right) = F(T_o(t))$ gives the statement of the lemma. □

Note that Lemma 1 is close to the concatenation property we desire, but not yet there. What is further required is how the arrival times of the oldest possible work units in each of the systems are related with the arrival time of the oldest possible work unit in the overall system. More technically, we need to show that $t_o^{(1,\dots,n)}(t) = T_o(t)$. This is done in the next two lemmas. Both lemmas assume that input and output functions at each of the systems of a tandem $\mathcal{T}$ are considered as given.

Lemma 2. *Assuming a tandem $\mathcal{T}$ of n systems $\mathcal{S}_1, \dots, \mathcal{S}_n$, it applies that*

$$t_o^{(1,\dots,n)}(t) \geq T_o(t).$$

Proof. The proof of Lemma 2 can be done via a mathematical induction over the number of systems n:
$\underline{n = 2:}$

Consider a certain time t. Assume $t_o^{(1,2)}(t) < T_o(t) = t_o^{(1)}\left(t_o^{(2)}(t)\right)$. That means

$$\exists \delta > 0 : t_o^{(1,2)}(t) = t_o^{(1)}\left(t_o^{(2)}(t)\right) - \delta.$$

Now assume we are on a sample path of the tandem $\mathcal{T}$ that achieves the oldest possible work unit with arrival time $t_o^{(1,2)}(t)$. Further assume this work unit would be at $\mathcal{S}_1$. The work unit must have been there already at $t_o^{(2)}(t)$ (otherwise we would have a sample path that achieves an older work unit), however this would contradict the definition of $t_o^{(1)}$ at time $t_o^{(2)}(t)$, as the work unit would have arrived earlier than $t_o^{(1)}\left(t_o^{(2)}(t)\right)$ (by δ time units). Hence, such a work unit cannot be at $\mathcal{S}_1$ at time t.

Now assume the oldest possible work unit is at system $\mathcal{S}_2$. It must have arrived at system $\mathcal{S}_2$ during $\left[t_o^{(2)}(t), t\right]$; let us assume it arrived at time $t_o^{(2)}(t) + \gamma$ with $\gamma \in \left[0, t - t_o^{(2)}(t)\right]$, which means it still had been at system $\mathcal{S}_1$ at time $t_o^{(2)}(t)$[2], which, in turn, would contradict the definition of $t_o^{(1)}$ at time $t_o^{(2)}(t)$, again. Hence, such a work unit cannot be at $\mathcal{S}_2$ at time t and consequently it cannot exist, so that we obtain

$$t_o^{(1,2)}(t) \geq T_o(t) = t_o^{(1)}\left(t_o^{(2)}(t)\right).$$

$\underline{n \to n+1:}$

Reusing the argument from the case with $n = 2$ systems for the concatenation of the first n systems of the tandem $\mathcal{T}$ and system $\mathcal{S}_{n+1}$ and applying the induction hypothesis yields

$$\begin{aligned} t_o^{(1,\dots,n+1)}(t) &\geq t_o^{(1,\dots,n)}\left(t_o^{(n+1)}(t)\right) \\ &= t_o^{(1)}\left(t_o^{(2)}\left(\cdots t_o^{(n+1)}(t)\right)\right). \end{aligned}$$

[2] This is due to the assumption of left-continuity for input and output functions; with right-continuous functions the argument would be technically somewhat more cumbersome.

□

So far, we made no specific assumptions about the systems $\mathcal{S}_i$ (in case of Lemma 1 they had to provide S^3C). Yet, for the last step we now need to make a further assumption: We assume each of the systems to have a *common waiting room* for the work units that await service at that system. A common waiting room (c.w.r.) in this context implies that the scheduling of a node is free to choose *any* of the backlogged work units for being processed next. This is, for example, the case if a single buffer is used at a node.

Lemma 3. *Assuming a tandem $\mathcal{T}$ of n systems $\mathcal{S}_1, \ldots, \mathcal{S}_n$, each with c.w.r.*[3]*, it applies that*

$$t_o^{(1,\ldots,n)}(t) \geq T_o(t).$$

Proof. Assume the tandem to be in a backlogged period at time t, i.e., $F'(t) < F(t)$. At time t, the oldest possible work unit at system $\mathcal{S}_n$ can have arrived at time $t_o^{(n)}(t)$. Assume we are on a sample path that achieves such an old work unit at system $\mathcal{S}_n$ and call this work unit u_o. As input and output functions are considered to be given, this sample path is purely governed by local work unit selection decisions at system $\mathcal{S}_n$. Now assume that when u_o was passed over from system $\mathcal{S}_{n-1}$ to $\mathcal{S}_n$ at time $t_o^{(n)}(t)$ it was selected to be the oldest work unit at $\mathcal{S}_{n-1}$ at time $t_o^{(n)}(t)$. Note that this selection is possible due to the c.w.r. assumption for system $\mathcal{S}_{n-1}$, i.e., we are free to choose any of the work units awaiting service at $\mathcal{S}_{n-1}$. Next, we further assume to be on a sample path that achieves the oldest possible work unit arriving at system $\mathcal{S}_{n-1}$ at time $t_o^{(n)}(t)$ (again this is purely governed by local decisions at system $\mathcal{S}_{n-1}$). This means that u_o arrived at system $\mathcal{S}_{n-1}$ at time $t_o^{(n-1)}\left(t_o^{(n)}(t)\right)$. Obviously, we can proceed like that until system $\mathcal{S}_1$ is reached and the work unit u_o would have arrived to the tandem $\mathcal{T}$ at time $t_o^{(1)}\left(t_o^{(2)}\left(\cdots t_o^{(n)}(t)\right)\right) = T_o(t)$, such that we can safely conclude that

$$t_o^{(1,\ldots,n)}(t) \geq T_o(t).$$

□

The following theorem sums up what is a consequence of Lemma 1, 2, and 3:

Theorem 5. *(Concatenation Theorem for a Tandem of n S^3C Systems) Consider a flow with input function F that traverses a tandem $\mathcal{T}$ of n systems $\mathcal{S}_1, \ldots, \mathcal{S}_n$. Assume that each $\mathcal{S}_i$ has a c.w.r. and offers a sufficiently strict service curve β_i to the flow. Then the tandem $\mathcal{T}$ offers a sufficiently strict service curve $\bigotimes_{i=1}^{n} \beta_i$ to the flow F.*

[3] In fact, it is sufficient if $\mathcal{S}_1, \ldots, \mathcal{S}_{n-1}$ have a c.w.r., but we neglect this detail here.

Proof. Simply combine Lemma 2 and 3 to find that $t_o^{(1,\dots,n)}(t) = T_o(t)$. Enter this into Lemma 1's statement and we obtain (denoting the output function after $\mathcal{T}$ as F' again)

$$\begin{aligned} F'(t) &\geq F\left(t_o^{(1,\dots,n)}(t)\right) + \left(\bigotimes_{i=1}^{n} \beta_i\right)\left(t - t_o^{(1,\dots,n)}(t)\right) \\ &= F\left(t - D^{(1,\dots,n)}(t)\right) + \left(\bigotimes_{i=1}^{n} \beta_i\right)\left(D^{(1,\dots,n)}(t)\right), \end{aligned}$$

where $D^{(1,\dots n)}(t)$ denotes the MADP of the tandem $\mathcal{T}$. □

We point out that the concatenation needs to be performed in one step because after the concatenation we generally have no c.w.r. for the overall system and thus Lemma 3 is not applicable any more. This is also why we have to prove all these statements in generality for n systems, so we can apply them together in one step. Nevertheless, we now have both properties we strived for: S^3C provides a real delay bound and has a concatenation property under the further assumption of a c.w.r. at the individual systems. The c.w.r. assumptions should not be a strong limitation in practice as the modeling starts with simple components for which this assumption is usually met by using a single buffer for the backlogged work units at such components. Based on the concatenation theorem a complex system can be constructed. For such a complex system, the c.w.r. assumption will not and does not need to hold any more since the bound on the real delay from Theorem 4 does not depend on that assumption.

Pay Bursts Only Once Holds for Non-FIFO S^3C Tandem By combining the results from Theorem 4 and 5, we can, for the case of a tandem of n non-FIFO nodes, each providing an S^3C $\beta_j, j = 1, \dots, n$, which is traversed by an α-constrained flow, derive a bound on the real delay as

$$\forall t \geq 0: \; rd(t) \leq i\left(\alpha, \bigotimes_{j=1}^{n} \beta_j\right).$$

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

$$\forall t \geq 0: \; rd(t) \leq \frac{b + nrT}{R - r} + nT,$$

which generally improves considerably on the additive bound based on S^2C from Section 3.2. We can perceive again the pay bursts only once phenomenon 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 4.4. Yet, before doing so, we first want to answer an important question: are there any systems providing this nice S^3C guarantee?

4.3 Are There Any Systems Providing S^3C ?

Admittedly, the definitions of the MADP and consequently S^3C are somewhat peculiar, raising the question whether there are any systems actually satisfying S^3C. In fact, it is hard to *directly* verify that a system provides S^3C. Instead, we show that S^2C nodes with c.w.r. provide S^3C.

Proposition 1. *A node with c.w.r. providing an S^2C β also provides β as S^3C.*

Proof. First, we show that the MADP is identical to the length of the backlogged period for S^2C with c.w.r. Let us define the backlogged period at time t as $B(t) = \sup_{s \le t} \{t - s : \forall u \in (s, t] : b(u) > 0\}$. That $D(t) \le B(t)$ is clear from the renewal character of an empty system just before $t - B(t)$, so there can be no more work units in the system with an arrival time earlier than $t - B(t)$. Further, under the assumption of c.w.r., it is clear that LIFO is a possible processing order and results in an arrival time for the oldest work unit of $t - B(t)$. Thus $D(t) \ge B(t)$, and altogether $D(t) = B(t)$. Now as the S^2C guarantee applies to any backlogged periods, it also applies for $(t - B(t), t]$. Further using that $t - B(t)$ is the start of the last backlogged period before t, i.e. $F(t - B(t)) = F'(t - B(t))$, we obtain

$$F'(t) \ge F'(t - B(t)) + \beta(B(t)) = F(t - B(t)) + \beta(B(t)) = F(t - D(t)) + \beta(D(t)),$$

which constitutes β as S^3C. □

So, the question whether there are any systems providing S^3C can be answered positively, because many schedulers are known to provide S^2C and thus also provide S^3C if c.w.r is satisfied.

4.4 Numerical Examples

To give some feeling for the improvements achievable by using the S^3C-based end-to-end delay analysis compared to a node-by-node bounding based on S^2C we provide a numerical example when applying the results for admission control purposes. 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 analyzed we assume a token bucket $\gamma_{r,b}$, where we set $b = 5[Mb]$ and 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 $T = 0.01[s]$. We assume $n = 10$ nodes to be traversed by the flow under analysis.

Worst-Case Admission Control A comparison between S^3C and S^2C-based bounding methods when used in admission control is provided in Figure 1. Here, the acceptable utilizations for a given delay bound are shown for both methods. This information can be used to decide how much traffic can be admitted. As can be clearly seen, the S^3C-based method outperforms the S^2C-based method by far, especially for lower delay bounds.

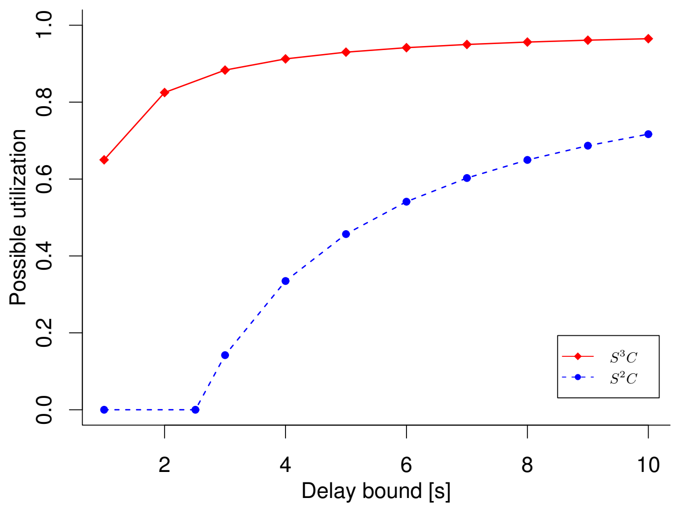

Fig. 1. Possible maximum utilizations for a target delay bound under S^2C and S^3C.

FIFO vs. Non-FIFO Delay Bounds In another numerical example, 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 flow under analysis (while at the same time scaling the bucket depth accordingly). As can be observed from Figure 2, only for higher utilizations there is a significant difference between the FIFO and non-FIFO delay bounds (at least for the S^3C-based bound). The bottom line is that only for highly utilized systems it is necessary to enforce a FIFO behaviour, as far as delay 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.

5 Simulations

In this section, by means of discrete-event simulations we investigate how actual queue management strategies, resulting in specific processing orders of work units, behave in comparison to our new bounds. To that end, we have built a queueing model for discrete-event simulations using `OMNeT++`[4] and implemented several queue management strategies: FIFO, LIFO, Shortest-In-System (SIS), and windowed random (RND(w)). SIS always selects the packet with the latest arrival time to the overall system. RND(w) chooses randomly, but restricted to the first w packets from a FIFO buffer, mimicking typical optimizations for core Internet routers with high-speed switching fabrics to avoid head-of-line blocking.

[4] `www.omnetpp.org`.

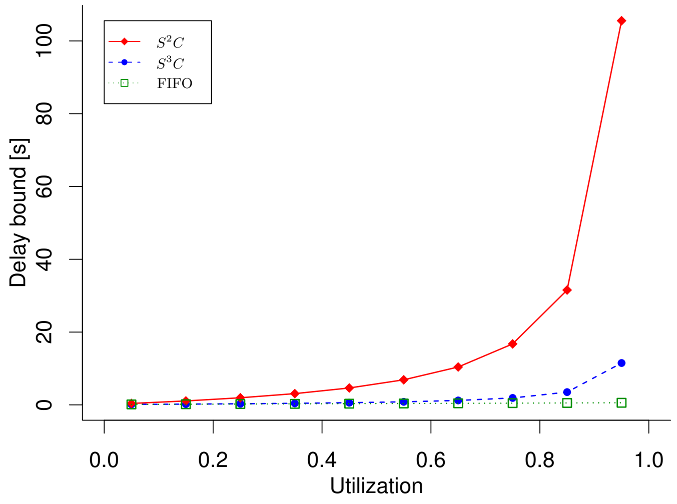

Fig. 2. FIFO vs. non-FIFO delay bounds depending on the utilization of the system.

5.1 Experimental Design

We simulate a tandem of n servers, each equipped with a single buffer; each server is TDMA-like with On and Off periods of duration $t_{on}^{(i)}$ and $t_{off}^{(i)}$, respectively; during an On period each server sends at rate $C^{(i)}$; each server applies one of the queue management strategies mentioned above to select the next packet eligible for service. For the analytical derivation of the bounds, we can abstract server i as S^3C with $\beta_{C^{(i)}/\left(1+t_{on}^{(i)}/t_{off}^{(i)}\right),t_{off}^{(i)}}$, because it provides this function as S^2C and has a c.w.r. (see Proposition 1).

We use a Markov-Modulated On-Off source with rates λ for the On-Off and μ for the Off-On transition, as well as a data rate P while in state On. Furthermore, the source is regulated by a token bucket $\gamma_{r,b}$ before entering the network.

The factor values corresponding to the simulation results in the next subsection can be found in Table 1. The primary factors in these experiments are the utilization, which is varied by using different token bucket rates, and the number of nodes to be traversed. Note that we chose descending Off times for the servers, because this yields more interesting results, since otherwise all the queueing in our system occurs at the first server, which, for example, makes SIS and LIFO equivalent strategies (as we also observed in simulations). For all factor combinations and all queue management strategies 100 simulation runs were performed; the results below report the average of the observed worst-case delays over these 100 replications.

Factor	Value
n	$1, \ldots, 10$
$C^{(i)}$	$200\,\frac{p}{s}$
$t_{on}^{(i)}$	$0.02\,s$
$t_{off}^{(i)}$	$i = n : 0.02\,s$, else $t_{off}^{(i)} = t_{off}^{(i+1)} - 0.002\,s$
λ	$20\,\frac{1}{s}$
μ	$20/3\,\frac{1}{s}$
P	$100\,\frac{p}{s}$
r	$10, 20, \ldots, 90\,\frac{p}{s}$
b	$20\,p$

Table 1. Factor Values for the Simulation Experiments.

5.2 Results

Different Queue Management Strategies. In Figure 3, the observed worst-case delays for all queue management strategies as well as the analytical worst-case delay bounds under FIFO and non-FIFO processing are displayed (for a utilization of 50%), from left to right in ascending order. We can observe that the non-FIFO bound is never violated. In contrast, the FIFO bound is actually violated by all queue management strategies (apart from FIFO, of course, which is slightly smaller); in particular, it is already violated by RND(2). We also see that SIS is closest to the non-FIFO bound and performs worse than LIFO. Clearly, the RND(w) delays deteriorate with increasing window size, and for $w = 8$ we are roughly at two third of the non-FIFO bound.

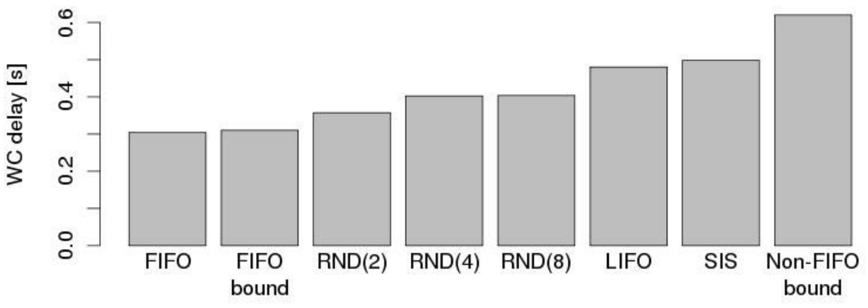

Fig. 3. Worst-case delays for different queue management strategies.

Varying the Utilization. The effect of varying the utilization of the system is shown in Figure 4. Here, we only show the results for SIS and RND(2) as well as the analytical bounds. As we can observe SIS exhibits the same growth behaviour as the non-FIFO bound and stays close up to high utilizations, though the gap tends to become larger. RND(2) is pretty insensitive to the utilization, but violates the FIFO bound for all utilizations.

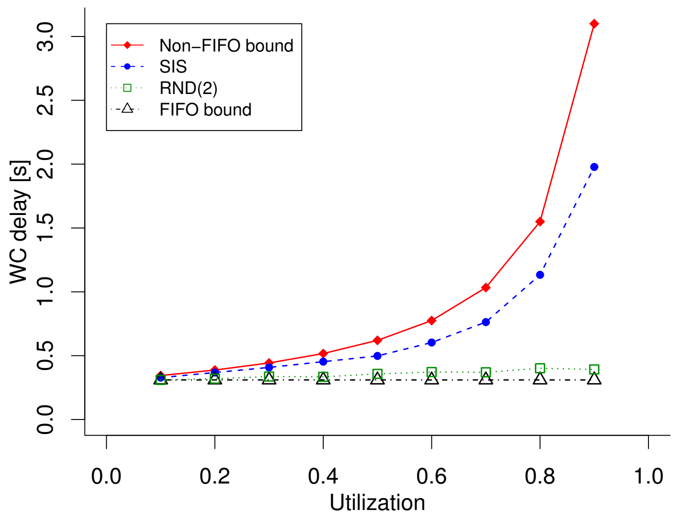

Fig. 4. The effect of utilization on worst-case delays.

Varying the Number of Nodes. In Figure 5, we can observe the effect of increasing the number of nodes to be traversed (at a utilization of 60%). Again, we only show the results for SIS and RND(2) as well as the analytical bounds. We can perceive that the good scaling behaviour of the analytical bounds is validated by the simulations. Again, SIS stays close to the non-FIFO bound whereas RND(2) stays close to FIFO. Interestingly, for low number of nodes we can observe that RND(2) can also stay below the FIFO bound.

Randomized Processing Order. In order to investigate the fairly realistic RND(w) strategies more deeply, we provide their simulation results along with the analytical bounds for 10 nodes under a varying utilization in Figure 6. Apart from the values of $w = 2, 4, 8$, we also implemented a strategy that always selects randomly from *all* queued packets, effectively realizing RND(∞). As we can observe, the observed worst-case delays of the RND(w) strategies fall between the FIFO and non-FIFO bounds. The higher values of w exhibit the same growth behaviour as the non-FIFO bound. It is interesting to note that RND(8) stays very close to RND(∞) and (due to random effects) can even be larger.

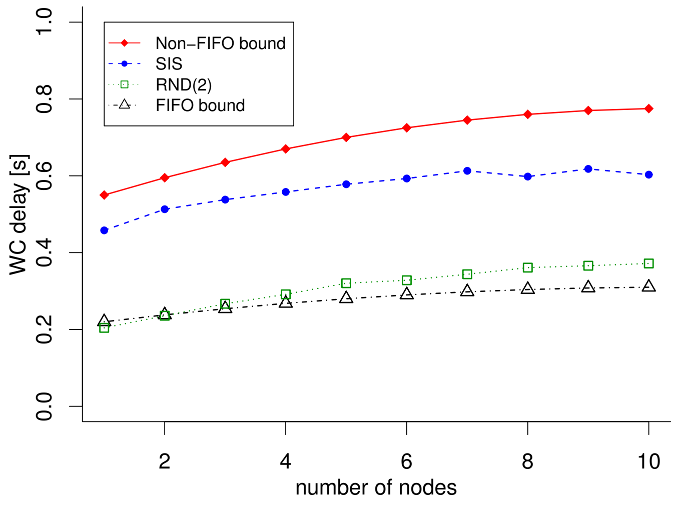

Fig. 5. The effect of number of nodes on worst-case delays.

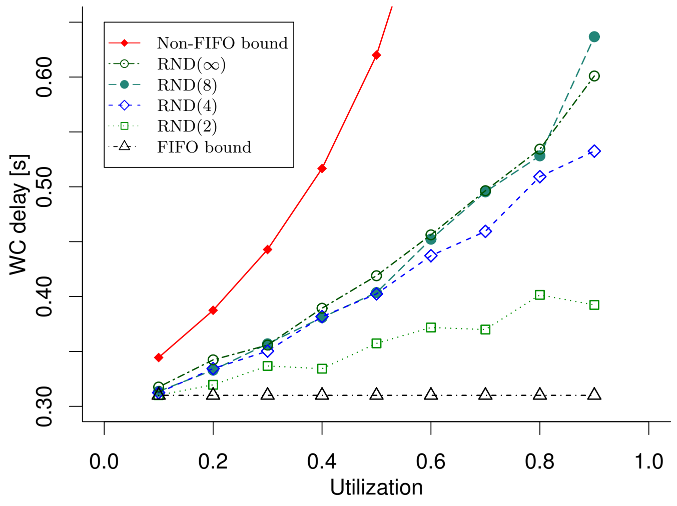

Fig. 6. The RND(w) strategies in comparison.

6 A Note on "Pay bursts only once does not hold for non-FIFO guaranteed rate nodes" [24]

In Section 4, 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 [24]. 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 server model, guaranteed rate (GR) nodes, some schedulers can be abstracted as both, GR or S^3C, and thus it is important to resolve this contradiction, as for such nodes only one statement can be true.

6.1 The Result from [24]

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 by limiting the deviation of its departure time from a hypothetical *FIFO* constant rate server. To that 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^{th} packet arriving at the node, $A(p^j)$ its arrival time, and R the server's allocated rate for the flow under analysis:

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

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 a GR node does not need to be FIFO, though its definition restricts its non-FIFO behavior. Possible reorderings of packets are restricted to packets leaving *early* with respect to their deadline. For this reason, a delay bound can be derived for a GR node even if it is not assumed to be FIFO. In fact, for a single node, the delay bound is the same as if FIFO were assumed [22]. However, in order to concatenate several GR nodes one draws upon their equivalence with SC in the *FIFO* case [22]. Clearly, this does not help in the non-FIFO case. Therefore, in [24], an additive delay bound based on the per-node delay bound for non-FIFO GR nodes is derived. More specifically, under a token-bucket arrival curve $\gamma_{r,b}$ and assuming a propagation delay τ^i between nodes i and $i+1$ as well as a maximum packet size l_{max}, the bound in [24] for a tandem of n GR nodes with equal service rate R is given as

$$d^{[24]} = n\frac{b}{R} + \frac{rl_{max}n\,(n-1)}{2R^2} + \frac{r}{R}\sum_{k=1}^{n}\sum_{i=1}^{k-1} e^i + \sum_{i=1}^{n}\left(e^i + \tau^i\right).$$

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. [24] claims that the bound is *tight* in general and thus concludes that "pay bursts only once does not hold for non-FIFO GR nodes".

6.2 The Bound From [24] Is Not Tight

A counter-example on the tightness of the bound in [24] can be constructed by assuming constant rate servers and choosing the following parameters: $n = 2$, $R = 1$, $e^i = 0$, $r = 0$, $b = 2$, $\tau^i = 0$, and all packets being of unit length. Then $d^{[24]} = 4$. However, it can be seen 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 $d^{[24]} = n\frac{b}{R} = 2n$. This shows that the delay bound from [24] 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 with $r < R$, it is not. Thus they restrict generality by that seemingly innocent assumption. In contrast, using S^3C allows to calculate the tight delay bound: We use $\beta^{stair}(t) := \lfloor t \rfloor$ as S^3C; for the concatenation of n nodes we have $\bigotimes_{i=1}^{n} \beta^{stair} = \beta^{stair} \otimes \delta_{n-1}$; thus we obtain

$$d^{S^3C} = i(\gamma_{0,2}, \beta^{stair} \otimes \delta_{n-1}) = n + 1.$$

A more comprehensive example for the problem regarding the tightness of [24] is shown in Figure 7. With this example it shall be demonstrated more clearly why the condition that $r = R$ is necessary for the bound to be tight. This example recreates the scenario from the tightness proof in the original paper, but drops the assumption that $r = R$. Instead, it uses $r = 0.5$ and $R = 1$. Other parameters are set as follows: $e_i = 2$, $\tau_i = 1$, $b = 4$, and unit length packets. According to the original bound, this results in $d^{[24]} = 25.5[tu]$; however, the example of a worst-case sample path shown in the figure demonstrates that a packet can at most spend $22tu$ in the system. The construction of that sample path follows the same principle as used in [24]: 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 $\left(n\frac{b}{r}\right)$; however, in an unsaturated system, the *GR* guarantee 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^{th} packet at the i^{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

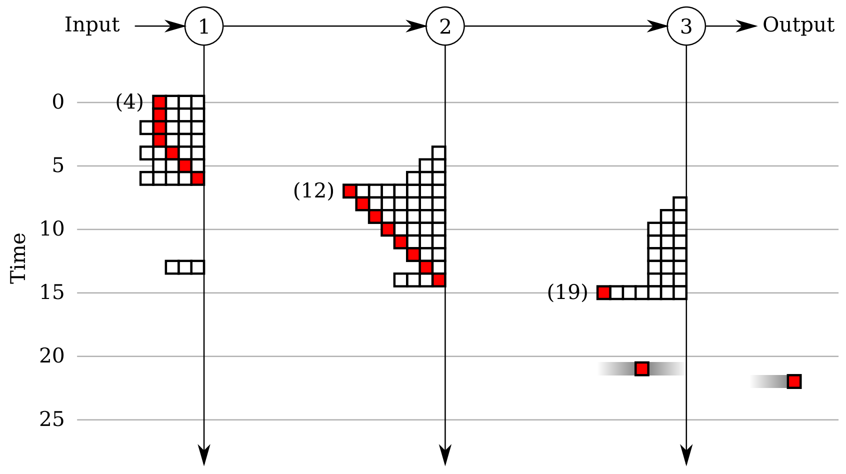

Fig. 7. Demonstration of the untightness of the delay bound presented in [24]. The buffers of three *GR* nodes are shown over time. Numbers in brackets are the watched packet's (shaded) *GRC*. Buffer contents are only shown as long as they are relevant to the sample path.

to send 3 packets. 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 *GRC*s 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 $\tau_3 = 1$, 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.

6.3 Comparison with S^3C-based Bounds

In principle, comparing the bound from [24] with an S^3C-based one is like a comparison between apples and oranges. On the one hand, a *GR* node is less stringent with respect to giving a guarantee about the work being done in a specific time interval as it can work ahead quite a lot and then be lazy for an extended period of time (similar to *SC*). On the other hand, a *GR* node has the great advantage over an S^3C node that it allows for much less reordering between work units as these are assigned deadlines when entering the *GR* node. Nevertheless, we provide a comparison between the two bounding methods for the following sample scenario: Assume we are given a tandem of PGPS servers (each with a single buffer) guaranteeing a rate R with latency $T = e_i$; such nodes can be abstracted as *GR* as well as S^3C nodes (because they are S^2C with c.w.r.). Then Figure 8 provides a numerical example (along with the parameter settings) comparing the delay bounds obtained from the S^3C-based method and [24]. Up to a very high utilization, S^3C yields less conservative bounds than

[24]. Only at a utilization of 95% the increased potential for reordering due to large backlogs outweighs the benefits of the concatenation-based S^3C bound. Whether a concatenation-based (and thus probably tight) bound can be found under the non-FIFO GR model remains an open problem.

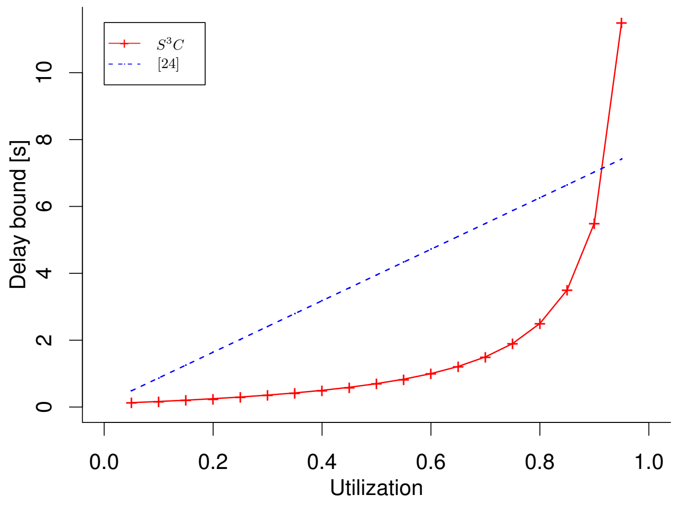

Fig. 8. Comparison of delay bounds from S^3C and [24] for a tandem of PGPS servers. The parameter settings are: $R = 20$, $T = e^i = 0.01$, $\tau^i = 0$, $r = \rho R$, $b = \frac{r}{2}$, unit packets for a tandem of $n = 10$ servers (i.e., $l_{max} = 1$), with ρ being the utilization.

7 Conclusion

It was our goal to extend the scope of network calculus towards non-FIFO systems, as it is theoretically interesting, methodologically consistent, and useful in real-world applications. It turned out that existing service curve definitions are not satisfying under non-FIFO processing of a flow: they are either too 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 efficient end-to-end analysis. By some numerical examples, we showed that the new analysis based on S^3C is clearly superior to existing methods; by simulations, we validated the bounds against actual system behaviour. Remarkably, S^3C allows to recover the pay bursts only once phenomenon for non-FIFO systems.

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. Rajeev Agrawal, R. L. Cruz, Clayton M. Okino, and Rajendran Rajan. A framework for adaptive service guarantees. In *Proceedings of Allerton Conf.*, pages 693–702, 1998.
3. 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.
4. 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.
5. J. C. R. Bennett and H. Zhang. WF^{2}Q: Worst-case fair weighted fair queueing. In *Proc. IEEE INFOCOM*, pages 120–128, March 1996.
6. J. M. Blanquer and B. Özden. Fair queuing for aggregated multiple links. *SIGCOMM Comput. Commun. Rev.*, 31(4):189–197, 2001.
7. A. Bouillard, L. Jouhet, and E. Thierry. Tight performance bounds in the worst-case analysis of feed-forward networks. In *Proc. IEEE INFOCOM*, pages 1–9, March 2010.
8. 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.
9. 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.
10. C.-S. Chang. *Performance Guarantees in Communication Networks.* Telecommunication Networks and Computer Systems. Springer-Verlag, 2000.
11. 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.
12. 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.
13. R. L. Cruz. A calculus for network delay, Part II: Network analysis. *IEEE Transactions on Information Theory*, 37(1):132–141, January 1991.
14. 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.
15. 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.
16. C. Guo. SRR: An O(1) time complexity packet scheduler for flows in multi-service packet networks. *IEEE/ACM Transactions on Networking*, 12(6):1144–1155, December 2004.
17. 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.
18. H. Kim and J.C. Hou. Network calculus based simulation: theorems, implementation, and evaluation. In *Proc. IEEE INFOCOM*, March 2004.
19. A. Koubaa, M. Alves, and E. Tovar. Modeling and worst-case dimensioning of cluster-tree wireless sensor networks. In *Proc. of RTSS'06*, pages 412–421. IEEE, 2006.

20. J.-Y. Le Boudec. Application of network calculus to guaranteed service networks. *IEEE Transactions on Information Theory*, 44(3):1087–1096, May 1998.
21. J.-Y. Le Boudec and A. Charny. Packet scale rate guarantee for non-fifo nodes. In *Proc. IEEE INFOCOM*, pages 23–26, June 2002.
22. 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.
23. 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.
24. 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.
25. 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.
26. J. Schmitt and U. Roedig. Sensor network calculus - a framework for worst case analysis. In *Proc. of DCOSS*, pages 141–154, June 2005.
27. 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.
28. 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.