Performance Bounds in Feed-Forward Networks under Blind Multiplexing

Jens B. Schmitt, Frank A. Zdarsky, and Ivan Martinovic
disco | Distributed Computer Systems Lab
University of Kaiserslautern, Germany

Technical Report No. 349/06

April 2006 / Revised in July 2006

Abstract

Bounding performance characteristics in communication networks is an important and interesting issue. In this study we assume uncertainty about the way different flows in a network are multiplexed, we even drop the common FIFO assumption. Under so-called blind multiplexing we derive new bounds for the tractable, yet non-trivial case of feed-forward networks. This is accomplished for pragmatic, but general traffic and server models using network calculus. In particular, we derive an end-to-end service curve for a flow of interest under blind multiplexing, establishing what we call the *pay multiplexing only once principle*. We specify the algorithms necessary to apply this result in a network of blind multiplexing nodes. Since these algorithms may have prohibitive computational costs, we present strategies to reduce the computational effort in a controlled manner such that the quality of the bounds is affected as little as possible. Finally we present some numerical results from a network calculus tool we developed and compare our bounds against the best known bounds for networks of blind multiplexing nodes.

Keywords: Network calculus, aggregate scheduling, deterministic guarantees.

1 Introduction

Bounding performance characteristics in communication networks is a fundamental issue and has important applications in network design and control. Network calculus, which is a set of relatively new developments that provide deep insights into flow problems encountered in networks of queues [18], provides a deterministic framework for worst-case analysis of delay and backlog bounds. The basic network calculus results pioneered by [9], [10] in the early 1990s, however, implicitly assume some form of per-flow treatment inside the network in that they only apply to tandems of nodes. Large-scale packet-switched networks as the Internet operate on large aggregates of traffic and are far-off from supporting any per-flow state operations, as for example, sophisticated per-flow scheduling. Nevertheless, as [18] pointed out in 2001: "The state of the art for aggregate multiplexing is surprisingly poor." Since then considerable efforts have been made to address issues related to bounding performance characteristics in networks of aggregate multiplexing: [7] gives a delay bound for general FIFO networks; [15] extends this work to packetized flows; [27] extends it further to nodes that are allowed to time-stamp packets (similar to what had been proposed previously in [12] as a damper); [26] reproduces the bound found in [7] and uses it for flow population-independent admission control; [13], [14] treat the case of feed-forward FIFO networks; [19] investigates the optimal delay bound under FIFO multiplexing; [25] gives additive delay bounds for tree topologies.

All of the above work assumes FIFO nodes. However in practice, as nicely argued in [17], many devices cannot be accurately described by a FIFO model because packets arriving at the output queue from different input ports may experience different delays when traversing a node. This is due to the fact that many networking devices like routers are implemented using input-output buffered crossbars and/or multistage interconnections between input and output ports. Hence, packet reordering on the aggregate level is a frequent event (not so on the flow level) and should not be neglected in modelling. Therefore, in this work we drop the FIFO assumption and make essentially no assumptions on the way aggregates are multiplexed at servers, i.e. we assume blind multiplexing [18]:

Definition 1: (Blind Multiplexing) If a node multiplexes several flows and the arbitration discipline between the flows accessing the service of the node is assumed to be unknown we call it a blind multiplexing node.

Another issue that arises during the investigation of aggregate scheduling is stability, i.e. whether a finite delay bound exists [4], [2]. For general networks of arbitrary topology it is still very much an open research problem under which circumstances a bound on the delay exists at all [18]. In [7] a sufficient condition for stability in general networks is given. Yet, for larger networks this puts a heavy constraint on the utilization of the network since the maximum allowable utilization is inversely proportional to the network diameter. At the other end of the spectrum of topologies, we have tandem networks for which network calculus has become famous to deliver tight bounds for any utilization ≤ 1 (mainly building upon the celebrated concatenation theorem [18]). We concentrate on

the middle ground between these two extremes: *feed-forward networks*.

DEFINITION 2: (Feed-Forward Network) A network is feed-forward if it is possible to find a numbering of its links such that for any flow through the network the numbering of its traversed links is an increasing sequence.

It is well-known and easy to show that feed-forward networks are stable for any utilization ≤ 1 [18]. While many networks are obviously not feed-forward, many important instances are. Among these:

- switched networks that typically use spanning trees for routing [20],
- wireless sensor networks that can often be modelled as a single-sink or multiple-sink topology that is feed-forward, in fact wireless sensor networks have even been proposed as an application field of network calculus [22],
- MPLS multipoint-to-point label-switched paths which also have been tackled using network calculus in [25].

Furthermore, there are very effective techniques to make a general network feed-forward. One of the advanced techniques (in comparison to a simple spanning tree) is the so-called turn-prohibition algorithm [23]. The idea of this algorithm is to prohibit a minimum number of so-called turns (pairs of consecutive edges), but not to delete the edges themselves as a spanning tree algorithm would. Turn-prohibition achieves considerably higher throughput than spanning trees and remains fairly close to the maximum achievable throughput of general networks of medium size [23].

So we investigate how to compute performance bounds in feed-forward networks of nodes with blind multiplexing under pretty general assumptions about traffic and server models. To the best of our knowledge this has not been done before.

2 Network Calculus Background

In this section, the necessary background material for the network calculus applied in this report is presented. Furthermore, some additions as they are needed throughout the report are given.

2.1 Network Calculus Basics

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 *minimum service curve* was introduced in [11], [21], [5], [16], [1] and the concept of *maximum service curve* in [12]. 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 referred to also as pay bursts only once phenomenon in [18]. A detailed

treatment of min-plus algebra and of network calculus can be found in [3] and [6], [18], respectively.

As network calculus is built around the notion of cumulative functions for input and output flows of data, the set 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. This is not a major restriction as there are transformations from discrete to continuous models which do not affect the accuracy of the results significantly [18].

DEFINITION 3: (Min-plus Convolution and Deconvolution) The min-plus convolution respectively deconvolution of two functions $f$and g are defined to be

$$(f \otimes g)(t) = \begin{cases} \inf_{0 \leq s \leq t} \{f(t-s) + g(s)\} & t \geq 0 \\ 0 & t < 0 \end{cases}$$

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

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 a linear operator on that dioid and is not even closed in $\mathcal{F}$. These algebraic characteristics result in a number of rules that apply to those operators. The rules most important to our investigation are summarized in the following theorem:

THEOREM 1: (Properties of $\otimes$) Let $f, g, h \in \mathcal{F}$

1. Isotonicity of $\otimes$: if $f \leq h$ then $f \otimes g \leq h \otimes g$
2. Commutativity of $\otimes$: $f \otimes g = g \otimes f$
3. Associativity of $\otimes$: $(f \otimes g) \otimes h = f \otimes (g \otimes h)$
4. Convolution of concave function: if f and g are concave functions and $f(0) = g(0) = 0$ then $f \otimes g = f \wedge g$.
5. Convolution of piecewise linear convex functions: if f and g are piecewise linear convex functions, $f \otimes g$ is obtained by putting end-to-end the different linear pieces of f and g, sorted by increasing slopes.
6. Composition of $\oslash$: $(f \oslash g) \oslash h = f \oslash (g \otimes h)$
7. Distributivity of $\otimes$ with respect to $\wedge$: $(f \wedge g) \otimes h = (f \otimes h) \wedge (g \otimes h)$
8. Duality between $\otimes$and $\oslash$: $f \oslash g \leq h \iff f \leq g \otimes h$

The proof of this list of properties can be found in [18]. Note that this list is by far not exhaustive. Let us turn now to the performance characteristics of flows that can be bounded by network calculus means:

DEFINITION 4: (Backlog and 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 first-in-first-out delivery, the delay for a bit input at time t is defined as

$$d(t) = \inf \{\tau \geq 0 : F(t) \leq F^*(t + \tau)\}$$

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

DEFINITION 5: (Arrival Curve) Given a flow with input function R a function $\alpha \in \mathcal{F}$ is an arrival curve for R if and only if

$$\forall t, s \geq 0, s \leq t : R(t) - R(t - s) \leq \alpha(s)$$

$$\Leftrightarrow R \leq R \otimes \alpha \Leftrightarrow \alpha \geq R \oslash R$$

Note that an arrival curve which is not sub-additive can be improved by its sub-additive closure [18]. As a further remark, remember that any concave function is sub-additive. The next corollary is a simple implication of the definition of arrival curves and the isotonicity of the min-plus convolution.

THEOREM 2: (Knowledge of Several Arrival Curves) If α_1 and α_2 are arrival curves for a flow F then $\alpha_1 \otimes \alpha_2$ is also an arrival curve for F.

DEFINITION 6: (Minimum and Maximum Service Curve) 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 minimum service curve β respectively a maximum service curve $\bar{\beta}$ if and only if

$$F^* \geq F \otimes \beta \quad \text{respectively} \quad F^* \leq F \otimes \bar{\beta}$$

The minimum service curve is more commonly used (therefore the minimum is often dropped from its name), since the maximum service curve is a weaker concept. Nevertheless, the maximum service curve will prove itself valuable for our purposes in improving the output bound of interfering flows. There is also a *strict* (minimum) service curve definition which is less general than the minimum service curve but sometimes is required to allow an analysis.

DEFINITION 7: (Strict Minimum Service Curve) Let $\beta \in \mathcal{F}$. We say that system $\mathcal{S}$ offers a *strict* minimum service curve β to a flow if, during any backlogged period of duration u, i.e. for any t for which $\forall s, 0 \leq s < u : b(t - s) > 0$, the output of the flow is at least equal to $\beta(u)$.

Note that any strict service curve is also a service curve. Using those concepts it is possible to derive basic performance bounds on backlog, delay and output:

THEOREM 3: (Performance Bounds) Consider a system $\mathcal{S}$ that offers a minimum service curve β and a maximum service curve $\bar{\beta}$. Assume a flow F traversing the system has an arrival curve α. Then we obtain the following performance bounds:
Backlog: $\forall t : b(t) \leq (\alpha \oslash \beta)(0) =: v(\alpha, \beta)$
Delay: $\forall t : d(t) \leq \inf\{t \geq 0 : (\alpha \oslash \beta)(-t) \leq 0\} =: h(\alpha, \beta)$
Output (arrival curve α^* for F^*): $\alpha^* \leq \left(\alpha \otimes \bar{\beta}\right) \oslash \beta$

Note that, if the maximum service cannot be bounded, i.e. $\forall t > 0 : \bar{\beta}(t) = \infty$ (which is the neutral element for the min-plus convolution), we obtain for the output bound $\alpha^* \leq \alpha \oslash \beta$. One of the strongest results of network calculus (albeit being a simple consequence of the associativity of $\otimes$) is the concatenation theorem that enables us to investigate tandems of systems as a single system:

THEOREM 4: (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 minimum service curve β_i and a maximum service curve $\bar{\beta}_i$, $i = 1, 2$ to the flow. Then the concatenation of the two systems offers a minimum service curve $\beta_1 \otimes \beta_2$ and a maximum service $\bar{\beta}_1 \otimes \bar{\beta}_2$ to the flow.

So far we have only covered the tandem network case, the next result factors in the existence of other interfering flows. In particular, it states the minimum service curve available to a flow at a single node under cross-traffic from other flows at that node.

THEOREM 5: (Blind Multiplexing Nodal Service Curves) Consider a node blindly multiplexing two flows 1 and 2. Assume that the node guarantees a *strict* minimum service curve β and a maximum service $\bar{\beta}$ to the aggregate of the two flows. Assume that flow 2 has α_2 as an arrival curve. Then

$$\beta_1 = [\beta - \alpha_2]^+$$

is a service curve for flow 1 if $\beta_1 \in \mathcal{F}$. $\bar{\beta}$ remains the maximum service curve also for flow 1 alone. Here, the $[.]^+$operator is defined as $[x]^+ = x \vee 0$, where $\vee$ denotes the maximum operator.

Note that we require the minimum service curve to be strict. In [18] an example is given showing that the theorem otherwise would not hold. It is *this theorem* that *we will generalize* to a feed-forward network case in Section 4 in such a way that the costs of multiplexing with interfering flows are only paid once, which forms one of the major contributions of this report. As already mentioned there is very profound work on aggregate scheduling in *general* networks, the most well known result is probably given by [7], later on reproduced and extended by [27], [15]. As the actual delay bound is derived based on a FIFO assumption we only extract the sufficient existence condition for a delay bound which is also valid under blind multiplexing:

THEOREM 6: (Blind Multiplexing in General Networks) Assume a general network of diameter h with any node n providing a sustained rate R_n and n having bounds on the rates of all incoming traffic C_n. Let us define the node utilization for node n as $u_n = \frac{\sum_{n \in f} r_f}{R_n}$, with r_f denoting the sustained rate of a flow f and $n \in f$ means that flow f traverses node n. Then a delay bound

exists if

$$\bigvee_n u_n < \bigwedge_n \sigma_{\{C_n > R_n\}} \left(\frac{C_n}{(C_n - R_n)(h-1) + R_n} \right)$$

with $\sigma_{\{cond\}}(x)$ equals x if *cond* is true and 1 otherwise.

2.2 Some Additions

In this subsection, we give some basic properties furtherly helpful in the network calculus derivations of Section 4.

LEMMA 1: (Distributivity of $[.]^+$ with respect to $\vee$) Let f, g be two real-valued functions then it applies that

$$[f \vee g]^+ = [f]^+ \vee [g]^+$$

PROOF: Assume there is a t for which $[(f \vee g)(t)]^+ \neq [f(t)]^+ \vee [g(t)]^+$. Either $f(t) > 0$ or $g(t) > 0$ (otherwise both sides are 0), also both cannot be > 0 (otherwise both sides are equal as the $[.]^+$ would have no effect). Without loss of generality assume $f(t) > 0$ and $g(t) \leq 0$, yet then $[(f \vee g)(t)]^+ = [f(t)]^+ = [f(t)]^+ \vee [g(t)]^+$ which contradicts the assumption and thus proves the statement of the lemma.

Note however, the min-plus convolution does not distribute with the minimum operator.

□

LEMMA 2: (Lower bound on $[.]^+$) Let f, g be two real-valued functions then it applies that

$$[f+g]^+(t) \geq (f+g)(t) \cdot 1_{\{f(t) \geq 0, g(t) \geq 0\}}$$

where we use the indicator function $1_{\{cond\}}$, which is 1 if *cond* is true and 0 otherwise.

PROOF: Assume the statement does not hold, that means: $\exists t : [f+g]^+(t) < (f+g)(t) \cdot 1_{\{f(t) \geq 0, g(t) \geq 0\}}$. Now $f(t)$ and $g(t)$ cannot both be ≥ 0 since otherwise both sides of the inequation are the same. If, however, one of them is < 0 then the right hand side of the inequation is 0 and thus $\leq$ to the left hand side. Thus the statement must hold.

□

LEMMA 3 (Commutativity of inf and $\vee$) Let $f_i : \mathbb{R}^n \to \mathbb{R}$, $i = 1, \ldots, n$. Let $S \subset \mathbb{R}^n$ then

$$\inf_{\overrightarrow{x} \in S} \left\{ \bigvee_{i=1}^n f_i(\overrightarrow{x}) \right\} = \bigvee_{i=1}^n \inf_{\overrightarrow{x} \in S} f_i(\overrightarrow{x})$$

PROOF: On the one hand ("$\leq$")

$$\forall \overrightarrow{x_0} \in S : \inf_{\overrightarrow{x} \in S} \left\{ \bigvee_{i=1}^n f_i(\overrightarrow{x}) \right\} \leq \bigvee_{i=1}^n f_i(\overrightarrow{x_0})$$

$$\Rightarrow \inf_{\overrightarrow{x} \in S} \left\{ \bigvee_{i=1}^{n} f_i\left(\overrightarrow{x}\right) \right\} \leq \left\{ \bigvee_{i=1}^{n} \inf_{\overrightarrow{x_0} \in S} f_i\left(\overrightarrow{x_0}\right) \right\}$$

on the other hand ("$\geq$")

$$\forall \overrightarrow{x_0} \in S : \bigvee_{i=1}^{n} f_i\left(\overrightarrow{x_0}\right) \geq \bigvee_{i=1}^{n} \inf_{\overrightarrow{x} \in S} f_i\left(\overrightarrow{x}\right)$$

$$\Rightarrow \inf_{\overrightarrow{x_0} \in S} \left\{ \bigvee_{i=1}^{n} f_i\left(\overrightarrow{x_0}\right) \right\} \geq \bigvee_{i=1}^{n} \inf_{\overrightarrow{x} \in S} f_i\left(\overrightarrow{x}\right)$$

which proves the statement of the lemma.

LEMMA 4: (Alternative Definition of Strict Service Curve) The following two statements are equivalent: (i) a system $\mathcal{S}$ offers β as a strict service curve to a flow F, (ii) $\forall t \geq 0 : \exists u \geq 0 : F(t) \geq F(t-u) + \beta(u)$ and $t-u$ is the start of the last backlogged period for F. Here we assume that for any point in time t where the system is not backlogged, the start of the last backlogged period is defined as t itself (i.e. $u = 0$).

PROOF: $(i) \Leftrightarrow$ during any backlogged period of duration $u \geq 0$ the system offers at least service $\beta(u)$ to the flow F.
$\Leftrightarrow$ for any time t for which the last backlogged period started at $t-u$ the output of the flow F^* satisfies: $F^*(t) - F^*(t-u) \geq \beta(u)$
$\Leftrightarrow$ for any time t for which the last backlogged period started at $t-u$ the output of the flow F^* satisfies: $F^*(t) \geq F(t-u) + \beta(u)$
$\Leftrightarrow (ii)$

□

3 Network Calculus with Piecewise Linear Curves

To be able to compute performance bounds in networks of blind multiplexing nodes, we need to apply basic network calculus operations like min-plus convolution and deconvolution on the arrival and service curves we encounter in a given scenario. This cannot be done on general functions with arbitrary characteristics but must, of course, eventually be done with specific instances of arrival and service curves. However, a class of functions which is both tractable as well as general enough to express all common candidates is that of piecewise linear functions. Hence, in this section we present for general piecewise linear arrival and service curves basic network calculus operations upon which further on the network analysis will be built in Section 5.

3.1 A Catalog of Useful Functions

In the definition of piecewise linear arrival and service curves the following catalog of functions from $\mathcal{F}$ is helpful:

DEFINITION 8: Auxiliary functions from $\mathcal{F}$

Burst delay functions: $\delta_T(t) = \begin{cases} +\infty & t > T \\ 0 & t \leq T \end{cases}$

Affine function (token bucket): $\gamma_{r,b}(t) = \begin{cases} rt + b & t > 0 \\ 0 & t \leq 0 \end{cases}$

Rate latency function: $\beta_{R,T}(t) = \begin{cases} R(t-T) & t > T \\ 0 & t \leq T \end{cases}$

From these functions general piecewise linear function can be constructed using the $\bigvee$ and $\bigwedge$ as we will encounter in the next subsection. Note that, as mentioned in [18], it applies that $\forall f \in \mathcal{F} : (f \otimes \delta_T)(t) = f(t - T)$. Hence, a convolution with the burst delay function δ_T results in a shift along the x-axis according to the value of T.

3.2 Choice of Arrival and Service Curves

As already discussed focusing on piecewise linear functions as arrival and service curves is no practical restriction, since it is still general enough to cover virtually any realistic case of traffic and server models. In particular many investigations make stronger assumptions: for example treating only the case of token buckets as arrival curve and rate latency functions as service curve is a popular simplification of matters.

3.2.1 Arrival Curve

Let us assume a piecewise linear concave arrival curve:

$$\alpha = \bigwedge_{i=1}^{n} \gamma_{r_i,b_i}$$

On the one hand it is possible to already use such a function as a traffic source description in order to be able to closely approximate a source's worst case behaviour. On the other hand, looking at the network analysis we cannot, as we will see when we factor in the maximum service curve, avoid to model arrival curves *inside* the network as general piecewise linear concave functions. This is due to the fact that they result from the addition of multiple flows induced by the multiplexing of the latter. To assume concavity is no major restriction because, as discussed in [18], non-concave functions (unless they are not sub-additive to be accurate) can be improved by pure knowledge of themselves, thus they cannot be tight.

3.2.2 Minimum Service Curve

Let us assume a piecewise linear convex minimum service curve:

$$\beta = \bigvee_{j=1}^{m} \beta_{R_j,T_j}$$

A piecewise linear convex minimum service curve results from deriving the service curve for a flow of interest at a node that blindly multiplexes this flow with other flows which, as an aggregate, have a piecewise linear concave arrival curve. Again, it might also be useful to model a node's service curve using several linear segments. To assume convexity is not a major restriction, as it might for instance apply if a node also has other duties. Though, in contrast to the arrival curve which is not sensible for non-concave functions (non-sub-additive to be exact), there are potentially sensible non-convex service curves as for example presented in [24], although this is rather uncommon.

3.2.3 Maximum Service Curve

Let us assume a piecewise linear *almost concave* maximum service curve:

$$\bar{\beta} = \left(\bigwedge_{k=1}^{l} \gamma_{\tilde{r}_k, \tilde{b}_k} \right) \otimes \delta_L$$

By *almost concave* we mean that the curve is only concave for values of $t > L$ and is 0 for values $t \leq L$. If $L > 0$ this models a node that has a certain minimum latency. The piecewise concavity models again the fact that a node may also have other duties. The concavity in contrast to the convexity for the minimum service curve is due to the fact that a best case instead of a worst case perspective has to be taken.

Before discussing the required network calculus operations on these curves, we first make an observation how the output bound of Theorem 3 can be improved under maximum service curves as they are assumed here.

3.3 Tightening the Output Bound

Realising that the maximum service curve induces arrival constraints for the output we can improve the output bound from Theorem 3 by the following lemma. (We prove it slightly more general–concave instead of just piecewise linear concave–than necessary for our purposes.)

LEMMA 5: (Improvement of Output Bound) Under the same assumptions as in Theorem 3, let additionally L be such that $L = \sup\{t \geq 0 : \bar{\beta}(t) = 0\}$ (L can be regarded as minimum/fixed delay for the system), if $\bar{\beta} \otimes \delta_{-L}$ is concave then the output bound for any flow R can be calculated as

$$\alpha^* = \left(\left(\alpha \otimes \bar{\beta} \right) \oslash \beta \right) \otimes \left(\bar{\beta} \otimes \delta_{-L} \right)$$

PROOF: A bound on the system output at time t can be computed as

$$\left(\bar{\beta} \oslash \bar{\beta} \right)(t) = \sup_{s \geq 0} \left\{ \bar{\beta}(t+s) - \bar{\beta}(s) \right\}$$

Since we have assumed $\bar{\beta}(t)$ to be zero for $t \leq L$ and to be concave for $t > L$ this means that the *sup* is taken on at $s = L$ such that

$$\left(\bar{\beta} \oslash \bar{\beta} \right)(t) = \bar{\beta}(t+L) = \left(\bar{\beta} \otimes \delta_{-L} \right)(t)$$

Hence, as $\bar{\beta} \otimes \delta_{-L}$ is a bound on the output of the system it is also an arrival curve for any output function R^* of the system. On the other hand, we have $\left(\alpha \otimes \bar{\beta}\right) \oslash \beta$ as an arrival curve for any R^* from Theorem 3. Applying Theorem 2 we obtain $\left(\left(\alpha \otimes \bar{\beta}\right) \oslash \beta\right) \otimes \left(\bar{\beta} \otimes \delta_{-L}\right)$ as an output bound.

□

3.4 Network Calculus Operations

We now have everything set to examine the network calculus operations we require as basic building blocks for a network analysis. First of all computing output bounds of the flows in the network is an important operation as it allows to separate flows of interest from interfering flows by using Theorem 5 respectively the results we present in Theorem 7. Lemma 4 gives us the general rule to compute the output bound:

$$\alpha^* = \left(\left(\alpha \otimes \bar{\beta}\right) \oslash \beta\right) \otimes \left(\bar{\beta} \otimes \delta_{-L}\right)$$

Hence, starting from the innermost operation, the convolution of the arrival curve and the maximum service must be determined first:

$$\alpha \otimes \bar{\beta} = \alpha \otimes \left(\left(\bigwedge_{k=1}^{l} \gamma_{\tilde{r}_k, \tilde{b}_k}\right) \otimes \delta_L\right) = \left(\alpha \otimes \bigwedge_{k=1}^{l} \gamma_{\tilde{r}_k, \tilde{b}_k}\right) \otimes \delta_L$$

$$= \left(\alpha \wedge \left(\bar{\beta} \otimes \delta_{-L}\right)\right) \otimes \delta_L =: \sigma$$

Here we used first the associativity of $\otimes$, then rule 4 from Theorem 1. While σ might look complex, it is easy to compute: first shift the maximum service curve to the left by its latency then take the minimum with the concave arrival curve and shift the result to the right by the latency of the maximum service curve. Next, the deconvolution of the resulting almost concave function σ with the minimum service curve is calculated as:

$$\sigma \oslash \beta = \left(\sigma \otimes \delta_{-X}\right) - \beta\left(X\right) =: \zeta$$

where $X = \sup_{t \geq 0} \left\{ \frac{d\sigma}{dt}\left(t\right) \geq \frac{d\beta}{dt}\left(t\right) \right\}$. X must be at one of the inflexion points of σ and β. Note that ζ is concave since always $X \geq L$. And finally ζ is convolved with the right-shifted maximum service curve using rule 4 from Theorem 1:

$$\zeta \otimes \left(\bar{\beta} \otimes \delta_{-L}\right) = \zeta \wedge \left(\bar{\beta} \otimes \delta_{-L}\right)$$

So, in terms of the initial curves we receive as an overall result for the output bound:

$$\alpha^* = \left(\left(\left(\left(\alpha \wedge \left(\bar{\beta} \otimes \delta_{-L}\right)\right) \otimes \delta_L\right) \otimes \delta_X\right) - \beta\left(X\right)\right) \wedge \left(\bar{\beta} \otimes \delta_{-L}\right)$$

Equipped with this a single node analysis can be accomplished. However, another basic operation consists of using the concatenation theorem to collapse

systems in sequence into one large system by calculating their min-plus convolution. So the convolution of piecewise linear minimum and maximum service curves must be computed. For the minimum service curves we can draw upon rule 5 from Theorem 1 as they are piecewise linear convex functions $\in \mathcal{F}$. For maximum service curves the next lemma gives us the computation of the min-plus convolution of two almost concave function:

LEMMA 6: (Min-Plus Convolution of Almost Concave Functions) Consider two almost concave piecewise linear functions $\bar{\beta}_1$ and $\bar{\beta}_2$ with latencies L_1 and L_2, then their min-plus convolution can be computed as follows

$$\bar{\beta}_1 \otimes \bar{\beta}_2 = \left(\bar{\beta}_1 \otimes \delta_{-L_1}\right) \wedge \left(\bar{\beta}_2 \otimes \delta_{-L_2}\right) \otimes \delta_{L_1+L_2}$$

PROOF: Again we make use of min-plus algebra

$$\begin{aligned}\bar{\beta}_1 \otimes \bar{\beta}_2 &= \left(\left(\bar{\beta}_1 \otimes \delta_{-L_1}\right) \otimes \delta_{L_1}\right) \otimes \left(\left(\bar{\beta}_2 \otimes \delta_{-L_2}\right) \otimes \delta_{L_2}\right) \\ &= \left(\left(\bar{\beta}_1 \otimes \delta_{-L_1}\right) \otimes \left(\bar{\beta}_2 \otimes \delta_{-L_2}\right)\right) \otimes \left(\delta_{L_1} \otimes \delta_{L_2}\right) \\ &= \left(\left(\bar{\beta}_1 \otimes \delta_{-L_1}\right) \wedge \left(\bar{\beta}_2 \otimes \delta_{-L_2}\right)\right) \otimes \delta_{L_1+L_2}\end{aligned}$$

Here we used associativity and commutativity of $\otimes$ as well as again rule 4 from Theorem 1.

□

4 The End-to-End Service Curve under Blind Multiplexing

Before we come to the actual algorithms for network analysis applying the basic operations that were just presented, we investigate different alternatives to derive the end-to-end service curve for a flow of interest through a network of blind multiplexing nodes. One possibility is to derive the end-to-end service curve based on the concatenation theorem and the result for single node blind multiplexing in Theorem 5. This evident method is mentioned in [18]. For example, if a scenario as depicted in Figure 1 is to be analysed for flow 1, a straightforward end-to-end service curve for flow 1 would be determined as follows (using the notation in Figure 1):

$$\beta_1^{SF} = [\beta_1 - \alpha_2 - \alpha_3]^+ \otimes [\beta_2 - \alpha_2^* - \alpha_3^*]^+ \otimes [\beta_3 - \alpha_3^{**}]^+$$

Yet, another way to analyse the system is to concatenate node 1 and 2, subtract flow 2 and thus receive the service curve for flow 1 and 3 together, concatenate this with node 3 and subtract flow 3, essentially making optimal use of the sub-path sharing between the interfering flows:

$$\beta_1^{PS} = \Big[[(\beta_1 \otimes \beta_2) - \alpha_2]^+ \otimes \beta_3 - \alpha_3\Big]^+$$

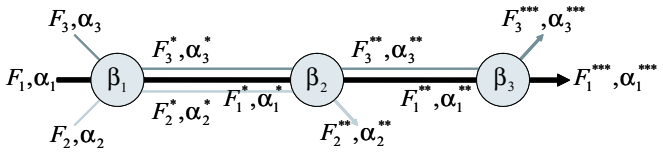

Figure 1: Nested interfering flows scenario.

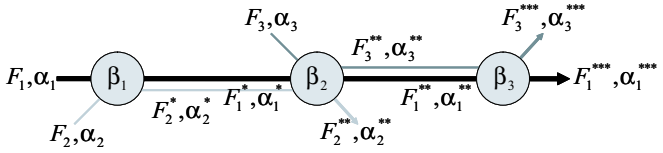

Figure 2: Overlapping interfering flows scenario.

If, for example, $\beta_i = \beta_{3,0}$, $i = 1, 2, 3$ and $\alpha_2 = \alpha_3 = \gamma_{1,1}$ we obtain: $\beta_1^{SF} = \beta_{1,12\frac{1}{2}}$ and $\beta_1^{PS} = \beta_{1,2}$. Hence, exploiting the sub-path sharing between the interfering flows, we obtain an end-to-end service curve with considerably lower latency. Put in other words, we have to pay for the blind multiplexing of the flows only once, which is why we also call this phenomenon the pay multiplexing only once (PMOO) principle, in analogy to the well-known pay bursts only once principle [18]. Basically the same observation was made in [13], [14] for *FIFO multiplexing* under the special case of token bucket arrival curves and rate-latency service curves. We derive the end-to-end service curve for *blind multiplexing* exploiting the PMOO principle under *general piecewise linear concave arrival curves* and *general piecewise linear convex service curves*.

The difficulty in obtaining the end-to-end service under blind multiplexing for a flow of interest lies in situations as depicted in Figure 2. Here, in contrast to the scenario of nested interfering flows as in Figure 1, we have a scenario of overlapping interfering flows, i.e. flows 2 and 3 which interfere with flow 1, our flow of interest, that share some servers with each other but each also traverses servers the other does not traverse. For such a scenario the end-to-end service curve cannot be derived as easily as demonstrated before but requires to look deeper into the input and output relationships of the flows. The following theorem states how to calculate the end-to-end service curve under blind multiplexing exploiting the PMOO principle for the canonical example of overlapping interfering flows in Figure 1.

THEOREM 7 (End-to-End Minimum Service Curve under Blind Multiplexing – Pay Multiplexing Only Once Principle)
Consider a scenario as shown in Figure 2: a flow of interest F_1 interfered by two overlapping other flows F_2 and F_3. F_2 and F_3 have arrival curves α_2 and α_3. The three servers each offer a *strict* minimum service curve $\beta_i, i = 1, 2, 3$. The

output flows of each of the servers are denoted as in Figure 2. If $\alpha_2 = \bigwedge_{i=1}^{n} \gamma_{r_i,b_i}$ and $\alpha_3 = \bigwedge_{j=1}^{m} \gamma_{\hat{r}_j,\hat{b}_j}$ are piecewise linear concave arrival curves then

$$\phi = \bigvee_{i=1}^{n} \bigvee_{j=1}^{m} \left[(\beta_1 - \tilde{\gamma}_{r_i,b_i}) \otimes \left(\beta_2 - \gamma_{r_i,0} - \gamma_{\hat{r}_j,0}\right) \otimes \left(\beta_3 - \tilde{\gamma}_{\hat{r}_j,\hat{b}_j}\right) \right]^+$$

constitutes a *strict* end-to-end service curve for the flow of interest, in particular $F_1^{***} \geq F_1 \otimes \phi$. Here we use a new notation: $\tilde{\gamma}_{r,b}(t) = \begin{cases} \gamma_{r,b}(t) & t \neq 0 \\ b & t = 0 \end{cases}$.

PROOF: Since β_3 is a strict service curve we know from Lemma 4 that $\forall t \geq 0 : \exists u \geq 0 :$

$$\left(F_1^{***} + F_3^{***}\right)(t) \geq \left(F_1^{**} + F_3^{**}\right)(t-u) + \beta_3(u)$$

and $t-u$ is the start of the last backlogged period at node 3.

$$\Longrightarrow F_1^{***}(t) - F_1^{**}(t-u) \geq \beta_3(u) - \left(F_3^{***}(t) - F_3^{**}(t-u)\right) \tag{1}$$

Along the same lines we can show for node 2 that $\forall (t-u) \geq 0 : \exists s \geq 0 :$
$\Longrightarrow F_1^{**}(t-u) - F_1^{*}(t-u-s) \geq \beta_2(s)$

$$-\left(F_2^{**}(t-u) - F_2^{*}(t-u-s)\right) - \left(F_3^{**}(t-u) - F_3^{*}(t-u-s)\right) \tag{2}$$

and $t-u-s$ is the start of the last backlogged period at node 2 and the composed service at node 3 and 2.

And for node 1 that $\forall (t-u-s) \geq 0 : \exists r \geq 0 :$
$\Longrightarrow F_1^{*}(t-u-s) - F_1^{*}(t-u-s-r) \geq \beta_1(r)$

$$-\left(F_2^{**}(t-u-s) - F_2^{*}(t-u-s-r)\right) \tag{3}$$

and $t-u-s-r$ is the start of the last backlogged period at node 1 and and the composed service at node 1,2, and 3.

If we add up (1), (2), and (3) we obtain $\forall t \geq 0 : \exists u, s, r \geq 0 :$

$$F_1^{***}(t) - F_1(t-u-s-r) \geq \beta_1(r) + \beta_2(s) + \beta_3(u)$$

$$-\left(F_2^{**}(t-u) - F_2(t-u-s-r)\right) - \left(F_3^{***}(t) - F_3(t-u-s)\right)$$

and $t-u-s-r$ is the start of the last backlogged period of the composed service at node 1, 2, and 3. Now using the causality of the systems under observation and the arrival curves for F_2 and F_3 we arrive at

$$F_1^{***}(t) - F_1(t-u-s-r) \geq \beta_1(r) + \beta_2(s) + \beta_3(u)$$

$$-\alpha_2(s+r) - \alpha_3(s+u)$$

$$
\begin{aligned}
&\geq \inf_{\substack{r',s',u'\geq 0 \\ r'+s'+u'=s+u+r}} \{\beta_1(r')+\beta_2(s')+\beta_3(u') \\
&\qquad -\alpha_2(s'+r')-\alpha_3(s'+u')\} \\
&= \inf_{\substack{r',s',u'\geq 0 \\ r'+s'+u'=s+u+r}} \{\beta_1(r')+\beta_2(s')+\beta_3(u') \\
&\qquad -\bigwedge_{i=1}^{n}\gamma_{r_i,b_i}(s'+r')-\bigwedge_{j=1}^{m}\gamma_{\hat{r}_j,\hat{b}_j}(s'+u')\Big\} \\
&= \inf_{\substack{r',s',u'\geq 0 \\ r'+s'+u'=s+u+r}} \Big\{\bigvee_{i=1}^{n}\bigvee_{j=1}^{m}(\beta_1(r')+\beta_2(s')+\beta_3(u')) \\
&\qquad -\gamma_{r_i,b_i}(s'+r')-\gamma_{\hat{r}_j,\hat{b}_j}(s'+u')\Big\} \\
&= \bigvee_{i=1}^{n}\bigvee_{j=1}^{m}\inf_{\substack{r',s',u'\geq 0 \\ r'+s'+u'=s+u+r}} \{\beta_1(r')+\beta_2(s')+\beta_3(u') \\
&\qquad -\gamma_{r_i,b_i}(s'+r')-\gamma_{\hat{r}_j,\hat{b}_j}(s'+u')\Big\} \qquad (4)
\end{aligned}
$$

The last equation is due to Lemma 3. Now we need to bring in the special shape of the arrival curves being additively composable. In particular we have

$$
\begin{aligned}
\gamma_{r_i,b_i}(s'+r') &= \begin{cases} \gamma_{r_i,0}(s')+\tilde{\gamma}_{r_i,b_i}(r') & s'+r'>0 \\ 0 & s'+r'=0 \end{cases} \\
\gamma_{\hat{r}_j,\hat{b}_j}(s'+u') &= \begin{cases} \gamma_{\hat{r}_j,0}(s')+\tilde{\gamma}_{\hat{r}_j,\hat{b}_j}(u') & s'+u'>0 \\ 0 & s'+u'=0 \end{cases}
\end{aligned}
$$

Note that we have different *equivalent* choices for the additive separation of the token buckets. To return to (4) we need to distinguish four cases:
Case 1: $s'+r'>0$ and $s'+u'>0 \Rightarrow s+r>0$ and $s+u>0 \Rightarrow$ either $s>0$ or $r,u>0$. This corresponds to having a non-zero backlog at least at two of the three nodes or at node 2, thus

$$
\begin{aligned}
&F_1^{***}(t)-F_1(t-u-s-r) \\
&\geq \bigvee_{i=1}^{n}\bigvee_{j=1}^{m}\inf_{\substack{r',s',u'\geq 0 \\ r'+s'+u'=s+u+r}} \{\beta_1(r')+\beta_2(s')+\beta_3(u') \\
&\qquad -\gamma_{r_i,0}(s')-\tilde{\gamma}_{r_i,b_i}(r')-\gamma_{\hat{r}_j,0}(s')+\tilde{\gamma}_{\hat{r}_j,\hat{b}_j}(u')\Big\}
\end{aligned}
$$

$$\otimes \left(\beta_3 - \tilde{\gamma}_{\hat{r}_j,\hat{b}_j}\right)\Big)\Big)(u+s+r)$$
$$= \left(\bigvee_{i=1}^{n}\bigvee_{j=1}^{m}\phi_{i,j}\right)(u+s+r)$$

with $\phi_{i,j} = (\beta_1 - \tilde{\gamma}_{r_i,b_i}) \otimes \left(\beta_2 - \gamma_{r_i,0} - \gamma_{\hat{r}_j,0}\right) \otimes \left(\beta_3 - \tilde{\gamma}_{\hat{r}_j,\hat{b}_j}\right)$ for $i = 1,\ldots,n$ and $j = 1,\ldots,m$.

Case 2: $s'+r' = 0$ and $s'+u' = 0 \Rightarrow s+r = 0$ and $s+u = 0 \Rightarrow s = r = u = 0$. This corresponds to having a non-zero backlog for the composed service at time $t = t-u-s-r$, thus

$$F_1^{***}(t) - F_1(t-u-s-r) = 0 \geq \left(\bigvee_{i=1}^{n}\bigvee_{j=1}^{m}\phi_{i,j}\right)(0)$$
$$= \bigvee_{i=1}^{n}\bigvee_{j=1}^{m} - \left(b_i + \hat{b}_j\right)$$

Case 3: $s' + r' > 0$ and $s' + u' = 0 \Rightarrow s + r > 0$ and $s + u = 0 \Rightarrow r > 0$ and $s = u = 0$. This corresponds to having a non-zero backlog only at node 1, thus

$$F_1^{***}(t) - F_1(t-u-s-r) \geq \bigvee_{i=1}^{n}\{\beta_1(r) + \tilde{\gamma}_{r_i,b_i}(r)\}$$
$$= \bigvee_{i=1}^{n}\{\beta_1(r) - r_i r - b_i\} \geq \left(\bigvee_{i=1}^{n}\bigvee_{j=1}^{m}\phi_{i,j}\right)(r)$$
$$= \bigvee_{i=1}^{n}\bigvee_{j=1}^{m}\left\{\beta_1(r) - r_i r - b_i - \hat{b}_j\right\}$$

Case 4: $s' + r' = 0$ and $s' + u' > 0 \Rightarrow s + r = 0$ and $s + u > 0 \Rightarrow u > 0$ and $s = r = 0$. This corresponds to having a backlog only at node 3, thus (analogous to *Case 3*)

$$F_1^{***}(t) - F_1(t-u-s-r) \geq \bigvee_{j=1}^{m}\left\{\beta_3(u) + \tilde{\gamma}_{\hat{r}_j,\hat{b}_j}(u)\right\}$$
$$\geq \left(\bigvee_{i=1}^{n}\bigvee_{j=1}^{m}\phi_{i,j}\right)(u) = \bigvee_{i=1}^{n}\bigvee_{j=1}^{m}\left\{\beta_3(u) - \hat{r}_j u - \hat{b}_j - b_i\right\}$$

Taking all four cases together we arrive at $\forall t \geq 0 : \exists u,s,r \geq 0 :$

$$\left(\bigvee_{i=1}^{n}\bigvee_{j=1}^{m}\phi_{i,j}\right)(u+s+r) \leq F_1^{***}(t) - F_1(t-u-s-r)$$

$$= F_1^{***}(t) - F_1^{***}(t-u-s-r)$$

The last equality is due to the fact that $t-u-s-r$ is the start of the last backlogged period for the service of the composed system. Since $F_1^{***} \in \mathcal{F}$ we obtain $\forall t \geq 0 : \exists u, s, r \geq 0 :$

$$F_1^{***}(t) - F_1(t-u-s-r) \geq \left[\left(\bigvee_{i=1}^{n} \bigvee_{j=1}^{m} \phi_{i,j} \right) (u+s+r) \right]^+$$

$$= \bigvee_{i=1}^{n} \bigvee_{j=1}^{m} \left[\phi_{i,j}(u+s+r) \right]^+ = \phi(u+s+r)$$

and $t-u-s-r$ is the start of the last backlogged period of the composed service system. Note that we used Lemma 1 for the last equality. According to Lemma 4 this establishes ϕ as a strict end-to-end service curve for flow 1.

□

As we are dealing with piecewise linear convex service curves we provide a corresponding application of Theorem 7 in the following corollary:

COROLLARY 1: (PMOO End-to-End Minimum Service Curve under Piecewise Linear Convex Nodal Service Curves) Under the assumptions of Theorem 7, suppose the three servers offer (strict) piecewise linear convex minimum service curves $\beta_1 = \bigvee_{i=1}^{n_i} \beta_{R_i,T_i}$, $\beta_2 = \bigvee_{j=1}^{n_j} \beta_{\hat{R}_j,\hat{T}_j}$, and $\beta_3 = \bigvee_{k=1}^{n_k} \beta_{\tilde{R}_k,\tilde{T}_k}$. With $\alpha_2 = \bigwedge_{l=1}^{n_l} \gamma_{r_l,b_l}$ and $\alpha_3 = \bigwedge_{m=1}^{n_m} \gamma_{\hat{r}_m,\hat{b}_m}$,

$$\phi = \bigvee_{i=1}^{n_i} \bigvee_{j=1}^{n_j} \bigvee_{k=1}^{n_k} \bigvee_{l=1}^{n_l} \bigvee_{m=1}^{n_m} \beta_{R_{i,j,k,l,m},T_{i,j,k,l,m}}$$

with

$$R_{i,j,k,l,m} = (R_i - r_l) \wedge \left(\hat{R}_j - r_l - \hat{r}_m \right) \wedge \left(\tilde{R}_k - \hat{r}_m \right)$$

and

$$T_{i,j,k,l,m} = T_i + \hat{T}_j + \tilde{T}_k$$

$$+ \frac{b_l + \hat{b}_m + r_l \left(T_i + \hat{T}_j \right) + \hat{r}_m \left(\hat{T}_j + \tilde{T}_k \right)}{(R_i - r_l) \wedge \left(\hat{R}_j - r_l - \hat{r}_m \right) \wedge \left(\tilde{R}_k - \hat{r}_m \right)}$$

constitutes a *strict* (piecewise linear convex) end-to-end service curve for the flow of interest, in particular $F_1^{***} \geq F_1 \otimes \phi$.

PROOF: Substituting the actual service curves into the result from Theorem 7 gives

$$\phi = \bigvee_{l=1}^{n_l} \bigvee_{m=1}^{n_m} \left[\left(\bigvee_{i=1}^{n_i} \beta_{R_i,T_i} - \tilde{\gamma}_{r_i,b_i} \right) \otimes \right.$$

$$\left(\bigvee_{j=1}^{n_j} \beta_{\hat{R}_j,\hat{T}_j} - \gamma_{r_i,0} - \gamma_{\hat{r}_j,0}\right) \otimes \left(\bigvee_{k=1}^{n_k} \beta_{\tilde{R}_k,\tilde{T}_k} - \tilde{\gamma}_{\hat{r}_j,\hat{b}_j}\right)\Bigg]^+$$

$$= \bigvee_{i=1}^{n_i} \bigvee_{j=1}^{n_j} \bigvee_{k=1}^{n_k} \bigvee_{l=1}^{n_l} \bigvee_{m=1}^{n_m} [(\beta_{R_i,T_i} - \tilde{\gamma}_{r_i,b_i}) \otimes$$

$$\left(\beta_{\hat{R}_j,\hat{T}_j} - \gamma_{r_i,0} - \gamma_{\hat{r}_j,0}\right) \otimes \left(\beta_{\tilde{R}_k,\tilde{T}_k} - \tilde{\gamma}_{\hat{r}_j,\hat{b}_j}\right)\Big]^+$$

$$= \bigvee_{i=1}^{n_i} \bigvee_{j=1}^{n_j} \bigvee_{k=1}^{n_k} \bigvee_{l=1}^{n_l} \bigvee_{m=1}^{n_m} \beta_{R_{i,j,k,l,m},T_{i,j,k,l,m}}$$

Here the first equation is due to Lemma 3 and the second equation is an application of Rule 5 from Theorem 1 (note that in Theorem 1 we required the functions to be $\in \mathcal{F}$, yet the construction rule also holds for the type of functions we have here).

□

The generalization to an arbitrary number of nodes is notationally complex but follows along the same line of argument as the three nodes with overlapping interference scenario, therefore it is left out here. It had of course to be taken into account for the implementation of the Network Calculator tool that will be described in Section 5. The derivation of the end-to-end maximum service curve is simple and given in the next lemma:

LEMMA 7: (E2E Maximum Service Curve under Blind Multiplexing) Consider a flow R that traverses a sequence of systems $\mathcal{S}_i$, $i = 1, \ldots, n$ where it is blindly multiplexed with an arbitrary number of flows. Assume each of the systems offers a nodal maximum service $\bar{\beta}_i$, ß $= 1, \ldots, n$ then flow R is offered

$$\bar{\beta} = \bigotimes_{i=1}^{n} \bar{\beta}_i$$

as a maximum service on its path.

It is obvious that in the best case R is not interfered at any of the systems $\mathcal{S}_i$ and thus receives the nodal maximum service curve $\bar{\beta}_i$ at each of its traversed systems such that the overall maximum service curve β follows directly from the basic concatenation result for maximum service curves in Theorem 4.

□

5 Network Analysis Algorithms

In this section, we now present the algorithms to analyse networks of blind multiplexing nodes. We focus on the delay bound computation. The backlog bound computation is analogous and just requires to compute the vertical deviation v where for the delay the horizontal deviation h is calculated.

Algorithm 1 Straightforward Network Analysis

ComputeDelayBound (flow of interest f)
  forall nodes $i \in$ path(f) starting at sink
    forall pred(i)
      α_{pred}+= ComputeOutputBound(pred(i),
        {flows to node i from pred(i)}\{f})
    $\beta_i^{eff} = [\beta_i - \alpha_{pred}]^+$
  $\beta^{SF} = \bigotimes_{i=1}^{n} \beta_i^{eff}$
  return $h\left(\alpha_f, \beta^{SF}\right)$

ComputeOutputBound(from node i, flows $\mathbb{F}$)
  forall pred(i)
    α_{pred}+ =ComputeOutputBound(pred(i),
        {flows to node i from pred(i)}$\cap\mathbb{F}$)
    α_{excl}+= ComputeOutputBound(pred(i),
        {flows to node i from pred(i)}\$\mathbb{F}$)
  $\beta_i^{eff} = [\beta_i - \alpha_{excl}]^+$
  return $\left(\left(\alpha_{pred} \otimes \bar{\beta}_i\right) \oslash \beta_i^{eff}\right) \otimes \left(\bar{\beta}_i \otimes \delta_{L_i}\right)$

5.1 Straightforward Network Analysis

This analysis is based on Theorem 5 and the way it is done is described on a high level in Algorithm 1. It consists mainly of two procedures: One to compute the delay bound for the flow of interest using the nodal service curve under blind multiplexing from Theorem 5 and convolving all nodal service curves to find the end-to-end service curve for the flow of interest which is then used to compute a delay bound for the flow. To be able to compute the nodal service curve, another procedure is required which computes the output bound for all interfering flows at a certain node. This is a recursive procedure which needs to take into account the effect of a general feed-forward network that even flows which never share a server with the flow of interest may affect the flow of interest transitively, by interfering with a flow which in turn interferes with the flow of interest.

5.2 PMOO Network Analysis

This analysis is based on Theorem 7 and the way it is done is described in Algorithm 2. It is more complicated than the straightforward analysis and consists of three procedures. The simplest is the one computes the delay bound and uses another procedure to compute the end-to-end service curve for the flow of interest according to Theorem 7. From the result it then calculates the delay bound for the flow of interest. The main complexity lies in the procedure to compute the PMOO service curve. Initially, the path of the flow of interest needs to be prepared to enable an efficient application of the PMOO result. At first, flows that join the flow of interest several times are eliminated by introducing

fresh flows for each later rejoin (with the arrival constraints of the original flow at that node in the network, of course). Next, parallel interfering flows, i.e. flows that have the same ingress and egress nodes, are merged together into a flow set which is treated as one flow for the computation of the service curve. The merging helps to keep the procedure as efficient as possible, since fewer flows have to be taken into account. However, care must be taken when computing the arrival constraints for each of the flow sets, since the flows contained in a set are generally not in parallel throughout the whole network. This is done by a separate procedure which we will discuss in a moment. After the arrival constraints of each of the flow sets have been computed, Theorem 7 is applied to compute the desired end-to-end PMOO service curve. This statement sounds harmless, however the computation of the maximum over all token buckets of the piecewise linear arrival curves of interfering flows and rate-latency functions of the piecewise linear service curve of the nodes on the path of the flow of interest can be a computationally intensive task. We come back to this issue in the next subsection. The last procedure serves to compute the output bound of a flow set behind a given node. Here, the PMOO result is reused as much as possible by first finding the sub-path shared by all flows in the flow set, respectively the node where the flow set splits. From that split point the procedure recurs on itself for each of the smaller flow sets after the split point. With the sum of those output bounds after the split point, the procedure now calls in indirect recursion the procedure for the computation of the PMOO service curve for the shared path of the flow set. Furthermore, the maximum service curve according to Lemma 6 is calculated. Together, minimum and maximum service curve are applied to calculate the output bound according to Lemma 4.

5.3 Computational Effort Reduction

As briefly pointed out in the preceding section, the computational effort for the PMOO analysis can be prohibitive as it grows exponentially with the number of interfering flows. This is due to the maximum computation from Theorem 7 that needs to be done for each combination of token buckets of the piecewise linear concave interfering flows and rate latency functions of the servers' piecewise linear convex service curves. Let us assume for this discussion that the nodes use simple rate latency functions such that those do not contribute to the combinatorial explosion. While this assumption will often hold in practice, it is possible, if not even very likely, for the arrival curves of the interfering flows to be composed of several token buckets due to the way the maximum service curve affects the burstiness of flows. Then the number of combinations for the computation of the PMOO service curve equals $\prod_{j=1}^{f} n_j$, where f is the number of interfering flows and n_j is the number of token buckets of which interfering flow j is composed. For instance, if all of the arrival curves of interfering flows consist of two token buckets when joining the flow of interest (not when they enter the network), then the number of combination is 2^f. The basic PMOO analysis algorithm already reduces the number of combinations by merging parallel flows. This does not affect the computation of the bound. However, we can

Algorithm 2 PMOO Network Analysis

ComputeDelayBound (flow of interest f)
  let p be the path of the flow of interest f
  β^{PMOO}= ComputePMOOServiceCurve(p, $\{f\}$)
  return $h\left(\alpha_f, \beta^{PMOO}\right)$

ComputePMOOServiceCurve(path p, flows $\mathbb{F}$)
  eliminate rejoining interfering flows
  merge parallel interfering flows into flowsets $\mathbb{F}_i$
  forall interfering flowsets $\mathbb{F}_i$
    forall pred($n_{\mathbb{F}_i}$) of the ingress node $n_{\mathbb{F}_i}$ of $\mathbb{F}_i$
      $\alpha_{\mathbb{F}_i}$+= ComputeOutputBound(pred($n_{\mathbb{F}_i}$), $\mathbb{F}_i$)
  calculate β^{PMOO} using $\alpha_{\mathbb{F}_i}$ according to Corollary 1
  return β^{PMOO}

ComputeOutputBound(from node i, flows $\mathbb{F}$)
  find shared path p of flows in $\mathbb{F}$ starting at node i
  call s the last node on path p ($\rightarrow$split point)
  forall pred(s)
    α_s+= ComputeOutputBound(pred(s),
      {flows to node s from pred(s)}$\cap\mathbb{F}$)
  β_p^{PMOO}= ComputePMOOServiceCurve(p, $\mathbb{F}$)
  calculate $\bar{\beta}_p^{PMOO}$ according to Lemma 6
  return $\left(\left(\alpha_{split} \otimes \bar{\beta}_p^{PMOO}\right) \oslash \beta_p^{PMOO}\right) \otimes \left(\bar{\beta}_p^{PMOO} \otimes \delta_{L_p}\right)$

go further in reducing the computational effort if we accept some degradation of the PMOO analysis.

There are two interesting techniques for directly reducing the computational effort of the maximum operation from Theorem 7 by decreasing the number of combinations to be computed (both can be invoked in procedure ComputeOutputBound in Algorithm 2 just before β^{PMOO} is calculated):

Flow Prolongation. The idea of this technique is that flows can be prolonged (of course, only for the analysis) such that they can be merged with other flows. Thus the number of combinations is reduced, for instance, from 2^f to 2^{f-1} for each flow that is prolonged. A flow prolongation incurs some cost as the flow consumes more resources at the servers it now traverses in addition depending on the latency of the servers and the rate of the flow. Often, this cost is small (and zero for a constant rate server). We therefore included that computational reduction technique into our PMOO analysis by accepting all flow prolongations that are below a certain "charge" starting with the shortest possible.

Arrival Curve Approximation. The idea of this technique is to reduce the complexity of the traffic descriptions of the interfering flows, i.e. to decrease the n_j, without compromising the arrival constraints of the interfering flows. Taken to the extreme we could represent each interfering flow with a

single token bucket, the one with the highest bucket depth and the lowest rate which would result in only a single combination that has to be computed. This would certainly be too much computational effort reduction at a potentially high degradation of the bounds. Therefore, we try to eliminate those linear segments from the arrival curves of interfering flows that have the least impact on their shape. To achieve, this we sort *all* segments in descending order of their y-range and only take a specified number of segments for the further calculation. Note that we do this globally over all segments of arrival curves of interfering flows and not per arrival curve to ensure that over-complex modelling is reduced at the right place. The target number of segments we keep is set to a selectable average number of segments per arrival curve times the number of interfering flows. This allows to reduce the computational effort in a controlled manner while degrading the bounds as little as possible.

Apart from these two techniques there is another option: restricting the recursion depth of the output bound computation. That means at a defined recursion depth we stop to use the PMOO-specific procedure to compute the output bound and use the straightforward analysis specific computation of the output bound instead. The latter avoids the combinatorial explosion from that recursion level on. For example, if the maximum recursion depth is set to 1, then we effectively do not use the PMOO result to compute the output bound but just use it to compute the service curve for the flow of interest.

6 Numerical Experiments

To validate and evaluate the proposed methods for computing performance bound in blind multiplexing networks we have implemented a toolbox called `DISCO Network Calculator`. This toolbox is more generally usable than just for the analysis of networks of blind multiplexing nodes. For instance, we have also implemented FIFO service curves for general piecewise concave arrival curves and piecewise linear convex service curves. The `DISCO Network Calculator`, which is written in Java^{TM}, is publicly available[1] and may hopefully be of use for other researchers interested in network calculus.

6.1 Experimental Design

In order to create a representative network we used the BRITE topology generator with the following recommended parameter settings: 2-level-top-down topology type with 10 ASes with 10 routers each, Waxman's model with $\alpha = 0.15, \beta = 0.2$, and random node placement with incremental growth type. The diameter of the generated topology is 11 hops. Hosts are created randomly at access routers: 30% random routers in each AS get a uniform random number of hosts between 1 and 5.

This topology is then transformed into a feed-forward network using the turn-prohibition algorithm. The turn-prohibition algorithm resulted in a change

[1] http://disco.informatik.uni-kl.de/content/Downloads

of 13% of the shortest path routes with an average change of 1.9 hops and a standard deviation of 1.2. So, altogether the routes were changed only by 0.25 hops on average. This can be considered a success for the turn-prohibition algorithm, because it was able to transform this general network with a diameter of 11 into a feed-forward network. Thus it is now suitable to be analysed by the methods presented in Section ?? for all possible network utilizations ≤ 1, whereas if it had been analysed as a general network using, e.g., the bound from [7] it could have been analysed only for utilizations below 10% (according to Theorem 2.1), so we are in a much better position now without paying a high cost in excessive route prolongation.

Now, for each source a flow is generated towards a randomly assigned sink. Each flow's arrival curve is drawn from a set $\mathbb{S}_\alpha$ of candidate arrival curves. Specifically, for this investigation we restricted the set members to token buckets. The minimum service curves were restricted to be rate-latency functions. These provide 1 of 3 rates: C, $2C$, or $3C$, where C is chosen such that the most loaded link can carry its load (plus a margin of 10%) and for the other links the rate is chosen closest to their carried load. The latency is chosen to be 10ms network-wide. If a maximum service curve is used its latency is set to zero and its rate equal to the rate of the corresponding nodal minimum service curve.

6.2 PMOO Analysis vs. Straightforward Analysis

One of the goals of the more comprehensive this study in was to find out how the straightforward analysis and the PMOO analysis compare to each other. We perform this comparison for two different traffic scenarios, one with low burstiness ($\mathbb{S}_\alpha = \{\gamma_{r,0.05[s]r}\}$, with $r = 10$ Mbit/s) and the other one with significantly higher burstiness ($\mathbb{S}_\alpha = \{\gamma_{r,0.05[s]r}, \gamma_{r,0.25[s]r}, \gamma_{r,0.5[s]r}\}$, with $r = 10$ Mbit/s). Furthermore, we want to isolate the effects by (1) the service curve computation using the PMOO principle, (2) the output bound computation using the PMOO principle, and (3) the maximum service curve in its improved version according to Lemma 3.3. We use all available sinks for flow generation and compute 10 replications with different seeds. The results can be found in Figure 3. First we take a look at the effect of the PMOO service curve alone: It is considerably better than its straightforward counterpart for both low and high burstiness of the traffic. As expected, the burstier traffic incurs higher absolute delays, but the percentage of improvement is roughly the same for both types of flows: $\approx 66 \pm 6.5\%$ at a 95% level of confidence. When the PMOO output bound is used this gives a further significant improvement of $\approx 13\%$ for the PMOO analysis (for both traffic types). When the maximum service curve is used this reduces the bounds by another $\approx 16\%$. The maximum service curve also improves the straighforward analysis by $\approx 35\%$ (low) and $\approx 45\%$ (high) and thus keeps its promise to be valuable for a good delay bound analysis although it leads to more complex models.

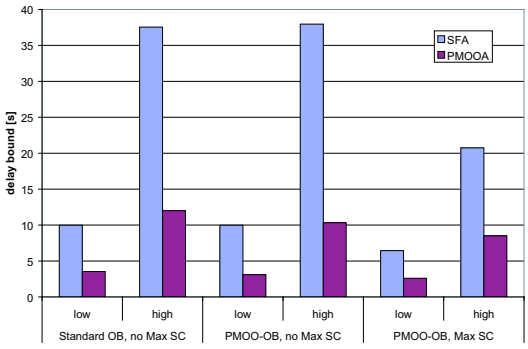

Figure 3: PMOO analysis vs. straighforward analysis in turn-prohibited networks.

7 Conclusion

In this report, we have presented a comprehensive set of methods to address the issue of computing performance bounds in feed-forward networks under blind multiplexing assumptions. Starting from the derivation of the basic operations for piecewise linear arrival and service curves required for the network analysis, we derived an interesting result on a novel way to compute the end-to-end service curve for a flow of interest under blind multiplexing, again for the pretty general case of piecewise linear arrival and service curves. Equipped with these results we presented algorithms for the network analysis for a straightforward technique based on existing results and a technique based on our new results. We demonstrated that in realistic environments the new bounds can be significantly better, reducing the existing bounds by at least 50% in our scenarios. A drawback of these bounds is their potentially high computational effort if the model sophistication is high, i.e. either sources have already complicated traffic descriptions or maximum service curves generate complex traffic flows inside the network. While often in theory such complexities are avoided due to their inelegance this may lead to much looser bounds than necessary. In fact, we are quite happy with our approach of allowing that complexity during the modelling phase and only during the computation of the bounds the complexity is reduced in a way to minimally impact the quality of the bounds.

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] M. Andrews. Instability of fifo in session-oriented networks. In *Proc. SODA*, pages 440–447, March 2000.

[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., West Sussex, Great Britain, 1992.

[4] C.-S. Chang. Stability, queue length and delay of deterministic and stochastic queueing networks. *IEEE Transactions on Automatic Control*, 39(5):913–931, May 1994.

[5] 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.

[6] C.-S. Chang. *Performance Guarantees in Communication Networks.* Telecommunication Networks and Computer Systems. Springer-Verlag, London, Great Britain, 2000.

[7] A. Charny and J.-Y. Le Boudec. Delay bounds in a network with aggregate scheduling. In *Proc. QofIS*, pages 1–13, September 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] R. L. Cruz. SCED+: Efficient management of quality of service guarantees. In *Proc. IEEE INFOCOM*, volume 2, pages 625–634, March 1998.

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

[14] M. Fidler and V. Sander. A parameter based admission control for differentiated services networks. *Computer Networks*, 44(4):463–479, 2004.

[15] Y. Jiang. Delay bounds for a network of guaranteed rate servers with fifo aggregation. *Computer Networks*, 40(6):683–694, 2002.

[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] L. Lenzini, E. Mingozzi, and G. Stea. Delay bounds for fifo aggregates: A case study. In *Proc. QofIS*, pages 31–40, February 2003.

[20] Radia Perlman. *Interconnections (2^{nd} ed.): bridges, routers, switches, and internetworking protocols*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2000.

[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. DCOSS*, pages 141–154, June 2005.

[23] D. Starobinski, M. Karpovsky, and L. A. Zakrevski. Application of network calculus to general topologies using turn-prohibition. *IEEE/ACM Transactions on Networking*, 11(3):411–421, 2003.

[24] I. Stoica, H. Zhang, and T. S. E. Ng. A hierarchical fair service curve algorithm for link-sharing, real-time and priority services. In *In Proc. ACM SIGCOMM*, pages 249–262, September 1997.

[25] G. Urvoy-Keller, G. Hébuterne, and Y. Dallery. Traffic engineering in a multipoint-to-point network. *IEEE Journal on Selected Areas in Communications*, 20(4):834–849, May 2002.

[26] S. Wang, D. Xuan, R. Bettati, and W. Zhao. Providing absolute differentiated services for real-time application in static-priority scheduling networks. In *Proc. IEEE INFOCOM*, pages 669–678, April 2001.

[27] Z.-L. Zhang, Z. Duan, and Y. T. Hou. Fundamental trade-offs in aggregate packet scheduling. In *Proc. ICNP*, pages 129–137, November 2001.