Estimating the Worst-case Delay in FIFO Tandems Using Network Calculus

Luca Bisti, Luciano Lenzini, Enzo Mingozzi, Giovanni Stea
Dipartimento di Ingegneria dell'Informazione, University of Pisa, Italy
Via Diotisalvi, 2 56122 Pisa, Italy - Ph. +39 050 2217599
{l.bisti, l.lenzini, e.mingozzi, g.stea}@iet.unipi.it

ABSTRACT
This paper addresses the problem of estimating the worst-case end-to-end delay for a flow in a tandem of FIFO multiplexing nodes, following up our previous work [12]. We show that, contrary to the expectations, the state-of-the-art method for computing *delay bounds*, i.e. upper bounds on the worst-case delay, called the *Least Upper Delay Bound* (LUDB) methodology, may actually be larger than the worst-case delay even in simple cases. Thus, we first devise a method to compute improved delay bounds. Then, in order to assess how close the derived bounds are to the actual, still unknown, worst-case delays, we devise a method to compute *lower bounds* on the worst-case delay. Our analysis shows that the gap between the upper and lower bounds is quite small in many practical cases, which implicitly validates the upper bounds themselves.

Categories and Subject Descriptors
C.4 [**Computer systems organization**] Performance of systems – *design studies*, *performance attributes*.

General Terms
Algorithms, Performance, Design.

Keywords
Network Calculus, FIFO-multiplexing, Delay Bound.

1. INTRODUCTION
The ability to provision reliable real-time services in a scalable way is the key to the deployment of the next generation Internet. It is now commonly agreed that per-aggregate resource management is to be employed in order to be able to scale to large networks and large number of users. Two noticeable examples of architectures employing per-aggregate resource management are Differentiated Services (DiffServ [2]), and Multi-Protocol Label Switching (MPLS, [4]), both standardized by the IETF. In the former, flows traversing a domain are aggregated (or *multiplexed*) in a small number of classes or Behavior Aggregates (BA), whose forwarding treatment is standardized, and QoS is provisioned on a per-aggregate basis at each node. In the latter, flows are aggregated into Forwarding Equivalence Classes (FECs) and forwarding and routing are performed on a per-FEC basis.

ValueTools 2008, October 21-23, 2008, Athens, GREECE.

The performance evaluation of real-time services in networks employing per-aggregate resource management is however particularly challenging. Real-time services require in fact firm QoS guarantees, usually formulated by computing an end-to-end *delay bound*, i.e. an upper bound on the maximum end-to-end delay. Obviously enough, a delay bound is as good as it is *tight*, i.e. close to the actual maximum delay that can theoretically be experienced by a bit of the flow. We refer to the latter as the *worst-case delay* (WCD). However, while it is fairly simple to compute the WCD under per-flow resource management (see, for example, [3], Chapter 2), computing it in networks employing per-aggregate resource management appears to be considerably more complex. During the last decade, several results have appeared in the literature on this subject, all based on Network Calculus ([3], [5]-[8]), a theory for deterministic network performance analysis. The aim of these works is to compute *delay bounds* in *feed-forward* networks, which are known to be stable for any utilization below 100% [3]. For instance, recent works [16], present tools and techniques for computing end-to-end delay bounds for flows in feed-forward networks of *blind* multiplexing nodes. "Blind" means that no assumption is made regarding the flow multiplexing criterion: for instance, both a FIFO multiplexing scheme and a strict priority multiplexing scheme in which the *tagged flow* (i.e., the one being analyzed) is always multiplexed at the lowest priority fit this definition. Smaller bounds can be obtained by explicitly assuming that a FIFO multiplexing scheme is in place at the node. As regards FIFO multiplexing, some recent works [9]-[12] describe a methodology for computing per-flow delay bounds in tandem networks of rate-latency nodes traversed by leaky-bucket shaped flows. The method, called *Least Upper Delay Bound* (LUDB), is based on the well-known Network Calculus theorem that allows a parametric set of *per-flow* service curves to be inferred from *per-aggregate* service curves at a *single node*. It consists in i) applying the above theorem iteratively, so as to obtain a *parametric set of end-to-end service curves* for a flow, ii) computing a parametric expression for the delay bound, and iii) minimizing over the set of parameters so as to obtain, in fact, the least upper bound. End-to-end analysis and global minimization are the two points of strength of LUDB. As shown in [11]-[12], we are actually able to derive end-to-end service curves only for a particular class of tandems, called *nested tandems*, where the path traversed by a flow *a* is either entirely included into the path of another flow *b* or has a null intersection with it. Non-nested tandems, instead, have to be partitioned into a number of nested sub-tandems, which have to be analyzed separately. Then, *per sub-tandem* delay bounds are computed and summed up to obtain the end-to-end delay bound.
LUDB has been shown to yield *tighter* bounds with respect to both per-node analysis, and another end-to-end methodology, described in [13], which does not use global minimization. However, despite

LUDB being the state-of-the-art solution to the problem, it is still unknown, in the general case, whether the bounds thus computed are actually equal to the WCD. As far as *sink-tree* tandems are concerned, it was proved in [10] that the LUDB is actually equal to the WCD. However, whether this is true for generic tandems is still an open question.

In this paper, we first provide a negative answer to the above question. Contrary to the expectations, the LUDB method may actually yield *loose* bounds even in very simple tandems. We prove this by counterexample: we devise a method, called *Flow Extension,* that can be used in conjunction to the LUDB methodology, so as to compute *smaller* delay bounds, at least in some cases. This result is significant for two reasons: on one hand, it improves delay bound computation in many cases; on the other hand, its significance from a theoretical standpoint lies in proving that the NC theorem that is at the core of the LUDB method is not always sufficient to describe the worst-case behavior of FIFO networks.

This said, assessing how tight the computed bounds are becomes an important issue. Being unable so far to identify a provable worst-case scenario, we propose heuristics to approximate it. More specifically, we construct a set of scenarios where a flow experiences a *large* delay, which is itself a *lower bound* on the WCD, and we provide an algorithm to compute this lower bound. The interval between the lower and the upper bounds serves as an estimate of the tightness of the upper bound itself.

The rest of the paper is organized as follows: Section 2 reports the necessary Network Calculus background. In Section 3 we give a formal problem statement, and we describe the LUDB methodology in Section 4. In Section 5 we prove that the LUDB may actually be larger than the WCD, also describing how to compute smaller bounds than the LUDB. In Section 6, we present an algorithm for computing a lower bound on the WCD, and we describe the tool that allows one to compute the upper and the lower bounds in Section 7, along with two non-trivial case studies. Finally, conclusions are reported in Section 8.

2. NETWORK CALCULUS BACKGROUND

Network Calculus is a theory for deterministic network analysis [3],[5]-[8]. The concept of service curve is introduced in Network Calculus as a general means to model a network element in terms of input and output flow relationships, i.e., how the element transforms an arriving stream of packets into a departing stream. To this aim, data flows are described by means of the cumulative function $R(t)$, defined as the number of bits seen on the flow in time interval $[0,t]$. Function $R(t)$ is wide-sense increasing, i.e. $R(s) \le R(t)$ if and only if $s \le t$. Henceforth, we only consider wide-sense increasing functions. Specifically, let $A(t)$ and $D(t)$ be the *Cumulative Arrival* and *Cumulative Departure* functions characterizing the same data flow before entering a network element, and after having departed, respectively. Then, the network element can be modeled by the *service curve* $\beta(t)$ if

$$D(t) \ge \inf_{0 \le s \le t} \{A(t-s) + \beta(s)\} \qquad (1)$$

for any $t \ge 0$. The flow is said to be guaranteed the (minimum) *service curve* β. A service curve may be seen as the Cumulative Departure Function of an initially empty system which is fed an infinite burst of traffic at time zero. The infimum on the right side of (1), as a function of t, is called the min-plus convolution of A and β, and is denoted by $(A \otimes \beta)(t)$. Min-plus convolution has several important properties, including being commutative and associative. Furthermore, convolution of concave curves is equal to their minimum. Several network elements, such as delay elements, links, and regulators, can be modeled by corresponding service curves. For example, network elements which have a transit delay bounded by φ can be described by the following service curve:

$$\delta_\varphi(t) = \begin{cases} +\infty & t \ge \varphi \\ 0 & t < \varphi \end{cases}$$

More interestingly, it has been shown that many packet schedulers can be modeled by a family of simple service curves called the *rate-latency* service curves, defined as follows:

$$\beta_{\theta,R}(t) = R \cdot [t-\theta]^+$$

for some $\theta \ge 0$ (the latency) and $R \ge 0$ (the rate). Notation $[x]^+$ denotes $\max\{0,x\}$. Assume that some queues are managed by a scheduler, which provides them with rate-latency service curves. Then the service of each queue has a bounded lag (equal to its latency) with respect to when that queue is served at a constant-rate in a single-queue system. A fundamental result of Network Calculus is that the service curve of a feed-forward sequence of network elements traversed by a data flow is obtained by convolving the service curves of each of the network elements.

Guaranteeing performance bounds to traffic flows requires that the arrivals be somewhat constrained. In Network Calculus this feature is modeled by introducing the concept of *arrival curve*. A wide-sense increasing function α is said to be an arrival curve (or, equivalently, an envelope) for a flow characterized by a cumulative arrival function A if it is:

$$A(t) - A(\tau) \le \alpha(t-\tau), \text{ for all } \tau \le t.$$

As an example, a flow regulated by a *leaky-bucket* shaper, with *sustainable rate* ρ and *burst size* σ, is constrained by the *affine* arrival curve

$$\gamma_{\sigma,\rho}(t) = (\sigma + \rho \cdot t) \cdot 1_{\{t>0\}}.$$

Function $1_{\{expr\}}$ is equal to 1 if *expr* is true, and 0 otherwise.

By combining together arrival and service curve characterizations of data traffic and network elements, respectively, it is possible to derive relevant performance bounds. Specifically, end-to-end delay bounds can be derived. In fact, assume that an element (or network of elements) is characterized by a service curve β and that a flow traversing that node is constrained by the arrival curve α. Then, if the node serves the bits of this flow in FIFO order, the delay is bounded by the horizontal deviation

$$h(\alpha,\beta) \triangleq \sup_{t \ge 0} \left[\inf \{ d \ge 0 : \alpha(t-d) \le \beta(t) \} \right] \qquad (2)$$

Intuitively, h is the amount of time the curve α must be shifted forward in time so that it lies below β. From (2) it follows that $\beta_1 \le \beta_2 \Rightarrow h(\alpha,\beta_1) \ge h(\alpha,\beta_2)$. Notation $\beta_1 \le \beta_2$ means that $\forall t \ \beta_1(t) \le \beta_2(t)$.

A well-known result related to a tandem of N rate-latency nodes β_{θ^i,R^i}, $1 \le i \le N$, traversed by a $\gamma_{\sigma,\rho}$ constrained flow follows from (2), i.e., the end-to-end delay bound is given by

$$d = \sum_{i=1}^{N} \theta^i + \sigma \Big/ \bigwedge_{1 \le i \le N} \{R^i\} \qquad (3)$$

if $\rho \le R^i$ for any i. Notation $\wedge$ denotes the minimum operation.

2.1 FIFO multiplexing

Regarding FIFO multiplexing, a fundamental result, first derived in [7], is reported in [3], Chapter 6. It allows one to compute a service

curve for a single flow based on the aggregate service curve and on the arrival curve of the interfering flow as follows:

Theorem 2.1 (FIFO Minimum Service Curves [3]).

Consider a lossless node serving two flows, 1 and 2, in FIFO order. Assume that packet arrivals are instantaneous. Assume that the node guarantees a minimum service curve β to the aggregate of the two flows. Assume that flow 2 has α_2 as an arrival curve. Define the family of functions:

$$\beta_1^{eq}(t,\tau) = \left[\beta(t) - \alpha_2(t-\tau)\right]^+ \cdot 1_{\{t>\tau\}}$$

For any $\tau \geq 0$ such that $\beta_1^{eq}(t,\tau)$ is wide-sense increasing, then flow 1 is guaranteed the (equivalent) service curve $\beta_1^{eq}(t,\tau)$.

Theorem 2.1 describes an *infinity* of equivalent service curves, each instance of which (obtained by selecting a specific value for the τ parameter), is a service curve for flow 1, provided it is wide-sense increasing. For ease of notation, we write $E(\beta,\alpha,\tau)(t)$ to denote the equivalent service curve obtained from applying Theorem 2.1 to a service curve $\beta(t)$, by subtracting from it arrival curve $\alpha(t-\tau)$. Hereafter, we omit repeating that curves are functions of time (and, possibly, of other parameters such as τ) whenever doing so does not generate ambiguity.

As an example, if the node is rate-latency, i.e. $\beta(t) = \beta_{\theta,R}(t)$, and flow 2 is leaky-bucket shaped, i.e. $\alpha_2(t) = \gamma_{\rho_2,\sigma_2}(t)$, then Theorem 2.1 yields the following set of equivalent service curves for flow 1 ([3], [9]).

$$E(\beta,\alpha_2,\tau)(t) = (R-\rho_2)\left[t - \left(\theta + \frac{\sigma_2 + \rho_2(\theta-\tau)}{R-\rho_2}\right)\right]^+ 1_{\{t>\tau\}} \quad (4)$$

The curves are also shown in Figure 1, from which the following two observations can be made:

a) $E(\beta,\alpha_2,\tau)$ is not necessarily a rate-latency curve. More specifically, it can be either a rate-latency curve (if $\tau \leq \theta + \sigma_2/R$) or a different kind of curve, namely an affine curve shifted to the right (if $\tau > \theta + \sigma_2/R$).

b) not all the curves obtained from Theorem 2.1 are actually relevant. For instance, all the curves obtained for $\tau < \theta + \sigma_2/R$ lie entirely below the one obtained for $\tau = \theta + \sigma_2/R$, and are therefore useless for computing performance bounds.

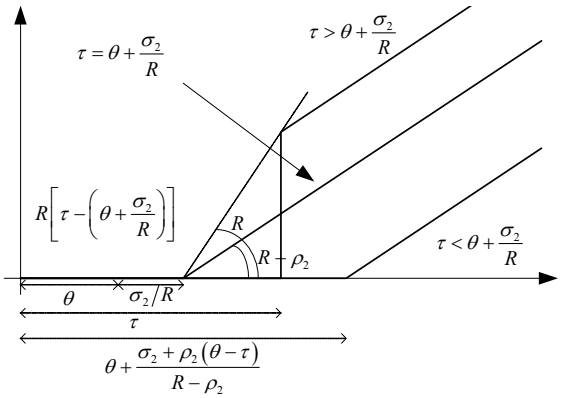

Figure 1. The set of equivalent service curves for flow 1

It has been proved in [10]-[12] (to which the interested reader is referred for more details and proofs) that *pseudoaffine* curves effectively describe the service received by single flows in FIFO multiplexing rate-latency nodes. We call a pseudoaffine curve one which can be described as:

$$\pi = \delta_D \otimes \left[\bigotimes_{1\leq x\leq n} \gamma_{\sigma_x,\rho_x}\right] \quad (5)$$

i.e., as a multiple affine curve shifted to the right. Note that, since affine curves are concave, (5) is equivalent to:

$$\pi = \delta_D \otimes \left[\bigwedge_{1\leq x\leq n} \gamma_{\sigma_x,\rho_x}\right]$$

We denote as *offset* the non negative term D, and as *leaky-bucket stages* the affine curves between square brackets. We denote with ρ_π^* (*long-term rate*) the smallest sustainable rate among the leaky-bucket stages belonging to the pseudoaffine curve π, i.e. $\rho_\pi^* = \min_{x=1,n}(\rho_x)$. We denote with Π the family of pseudoaffine curves. A rate-latency service curve is in fact pseudoaffine, since it can be expressed as $\beta_{\theta,R} = \delta_\theta \otimes \gamma_{0,R}$. A three-stage pseudoaffine curve is shown in Figure 2.

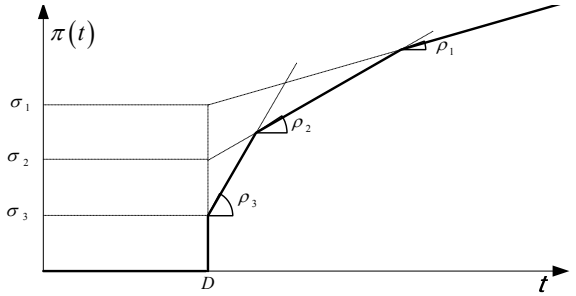

Figure 2. Example of a three-stage pseudoaffine curve

The alert reader will notice that, for any value of τ, all the curves obtained from (4) are pseudoaffine. Although more general than rate-latency curves, pseudoaffine curves are still fairly easy to manage from a computational standpoint: it can be easily shown that the convolution of two pseudoaffine curves is a pseudoaffine curve, whose offset is the sum of the offsets of the operands, and whose leaky-bucket stages are the leaky-bucket stages of both operands. Furthermore, Theorem 2.1 can be specialized for the case of pseudoaffine service curves and leaky-bucket arrival curves as follows:

***Corollary 2.2* ([10]):**

Let π be a pseudoaffine service curve, with offset D and n leaky-bucket stages γ_{σ_x,ρ_x}, $1 \leq x \leq n$, and let $\alpha = \gamma_{\sigma,\rho}$, with $\rho_\pi^* \geq \rho$. If a node guarantees a minimum service curve π to the aggregate of the two flows, and flow 2 has α as an arrival curve, then the family of functions $\left\{\overline{E}(\pi,\alpha,s), s \geq 0\right\}$, with:

$$\overline{E}(\pi,\alpha,s) = \delta_{D+\bigvee_{1\leq i\leq n}\left[\frac{\sigma-\sigma_i}{\rho_i}\right]^+ + s} \otimes \left[\bigotimes_{1\leq x\leq n} \gamma_{\rho_x\left\{s+\bigvee_{1\leq i\leq n}\left[\frac{\sigma-\sigma_i}{\rho_i}\right]^+ - \frac{\sigma-\sigma_x}{\rho_x}\right\}^+,\rho_x-\rho}\right],$$

or, equivalently,

$$\overline{E}(\pi,\alpha,s) = \delta_{h(\alpha,\pi)+s} \otimes \left[\bigotimes_{1\leq x\leq n} \gamma_{\rho_x\{s+h(\alpha,\pi)-D\}^+ - (\sigma-\sigma_x),\rho_x-\rho}\right] \quad (6)$$

are pseudoaffine equivalent service curves for flow 1.

It can be proved that the set $S \triangleq \left\{\overline{E}(\pi,\alpha,s), s \geq 0\right\}$ is a proper *subset* of $\mathrm{T} = \left\{E(\beta,\alpha,\tau), \tau \geq 0\right\}$, i.e. it does not include some equivalent service curve that would be computed through Theorem 2.1. However, it does include those equivalent service curves which are relevant for computing delay bounds. More specifically, for each curve $x \in \mathrm{T} \setminus S$, there exists a curve $y \in S$ such that $y \geq x$. Therefore, all the performance bounds that can be found by

applying Theorem 2.1 can also be found by applying Corollary 2.2. With reference to the example of Figure 1, Corollary 2.2 yields:

$$\overline{E}(\beta,\alpha_2,s)=\delta_{s+\theta+\frac{\sigma_2}{R}}\otimes\gamma_{R\cdot s,R-\rho_2} \tag{7}$$

i.e., all the equivalent service curves obtained from Theorem 2.1 with $\tau\geq\theta+\sigma_2/R$. Note that (7) is much more compact than (4).

3. SYSTEM MODEL

We analyze a tandem of N *nodes*, connected by links. The tandem is traversed by *flows*, i.e. distinguishable streams of traffic. We are interested in computing a tight end-to-end delay bound for a specific flow, i.e. the *tagged flow tf*, which traverses the whole tandem from node 1 to N. At each node, *FIFO multiplexing* is in place, meaning that all flows traversing the node are buffered in a single queue First-Come-First-Served. Furthermore, the aggregate of the flows traversing a node is guaranteed a minimum service, in the form of a *rate-latency* service curve, with rate R^k and latency θ^k, $1\leq k\leq N$. In the above framework, a flow can be identified by the couple (i,j), $1\leq i\leq j\leq N$, where i and j are the first and last node of the tandem at which the flow is multiplexed with the aggregate. We model a flow as a stream of *fluid*, i.e. we assume that it is feasible to inject and service an arbitrarily small amount of traffic at a node, and we leave packetization issues for further study. We assume that flows are constrained by a σ,ρ *leaky-bucket* arrival curve at their ingress node. Leaky-bucket curves are additive, i.e. the aggregate of two leaky-bucket shaped flows is a leaky-bucket shaped flow whose arrival curve is the sum of the two. Hence, without any loss of generality, we assume that at most *one* flow exists along a path (i,j) and we identify it using the path (i,j) as a subscript. It was also proved that, in order to compute the *end-to-end delay bound*, all flows traversing path $(1,N)$ can be considered as if they were one flow, i.e. the tagged flow.

Based on how the paths of its flows are interleaved, we classify tandems as being either *nested* or *non nested*. In a *nested* tandem, flows are either *nested* into one another, or they have null intersection. This means that no two flows (i,j), (h,k) exist for which $i<h\leq j<k$. Said in other words, let us consider two flows (i,j), (h,k), with $(i,j)\neq(h,k)$ and $i\leq h$. Then either $j<h$, or $k\leq j$. In the first case, the two flows span a disjoint set of nodes. In the second case, we say that (h,k) is *nested within* (i,j), and we write $(h,k)\subset(i,j)$. For example, Figure 3 represents a nested tandem of three nodes. Flow $(3,3)$ is nested within flow $(2,3)$. Moreover, flows $(1,1)$, $(3,3)$ and $(2,3)$ are nested within the tagged flow $(1,3)$.On the other hand, a tandem is non-nested if it does not verify the above definition, as the one shown in Figure 4, below. In that case, we say that flow $(1,2)$ *intersects* flow $(2,3)$.

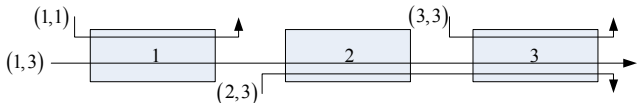

Figure 3. A nested tandem

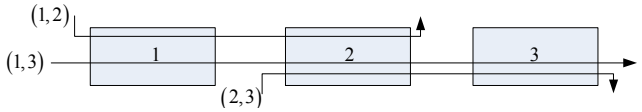

Figure 4. A non-nested tandem

Finally, as far as rate provisioning is concerned, we assume that a node's rate is no less than the sum of the sustainable rates of the flows traversing it, i.e. for every node $1\leq h\leq N$,

$$\sum_{(i,j):i\leq h\leq j}\rho_{(i,j)}\leq R^h \tag{8}$$

This allows a node's rate to be utilized up to 100%, thus being a necessary condition for stability. Moreover, we assume that the buffers are large enough to guarantee that traffic is never dropped.

4. THE LEAST UPPER DELAY BOUND METHODOLOGY

In this paragraph, we describe the *Least Upper Delay Bound* (LUDB) methodology. We first explain it on nested tandems, and extend it to non-nested tandems later on. At a first level of approximation, LUDB consists in computing *all* the service curves for the tagged flow: we start from the aggregate service curves at each node, we apply Corollary 2.2 iteratively in order to remove one flow $(i,j)\neq(1,N)$ from the tandem, and we convolve the service curves of nodes traversed by the same set of flows. Every time Corollary 2.2 is used, a new free parameter $s_{(i,j)}$ is introduced. Therefore, we compute in fact an *infinity* of service curves. From each of these we can compute a delay bound for the tagged flow, hence the minimum among all the delay bounds is the *least upper delay bound*.

For instance, let us consider again the three-node nested tandem shown in Figure 3. Figure 5 shows how to compute the set of end-to-end service curves for the tagged flow (1,3). We start from the aggregate service curves at each node, and we apply Corollary 2.2, starting from nodes 1 and 3. Then we convolve the service curves obtained for nodes 2 and 3, which are now traversed by the same aggregate of flows (1,3) and (2,3). We remove flow (2,3) by applying once more Corollary 2.2, and we obtain the set of end-to-end service curves for the tagged flow through convolution. The service curves $\pi^{\{1,3\}}\left(s_{(1,1)},s_{(3,3)},s_{(2,3)}\right)$ depend on three parameters, $s_{(1,1)}$, $s_{(2,3)}$, $s_{(3,3)}$, and they are pseudoaffine for each instance of them.

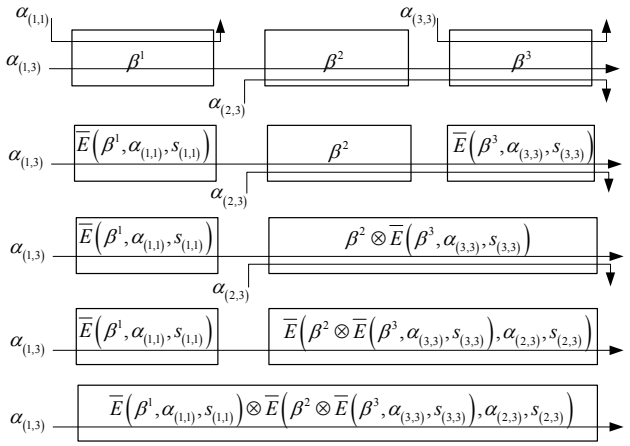

Figure 5. An example of application of the LUDB methodology

The best end-to-end delay bound that we can compute through this method, i.e. the LUDB, is the minimum among all the bounds that can be computed for each instance of the free parameters. For the above example, the problem can be formulated as follows:

$$\begin{cases}\min h\left(\alpha_{(1,3)},\pi^{\{1,3\}}\left(s_{(1,1)},s_{(3,3)},s_{(2,3)}\right)\right)\\ s_{(1,1)},s_{(3,3)},s_{(2,3)}\geq 0\end{cases} \tag{9}$$

Now, since $\pi^{\{1,3\}}(\ )$ is pseudoaffine and $\alpha_{(1,3)}$ is an affine curve,

problem (9) is an optimization problem with a *piecewise linear* objective function of x variables and x linear constraints, x being the number of distinguished *flows* in the tandem minus one, $x < 2N$, i.e. it is a *piecewise-linear programming* (P-LP) problem. In [11], this has been proved to be true for all nested tandems.

The LUDB methodology cannot be applied directly to non-nested tandems, such as the one shown in Figure 4. In fact, in that case, there are no two consecutive nodes traversed by the same set of flows, since two flows intersect each other. In [11], it was observed that a non-nested tandem can always be *cut* into at most $\lceil N/2 \rceil$ *disjoint nested sub-tandems*. Therefore, one can use LUDB to compute partial, per sub-tandem delay bounds, and an end-to-end delay bound can be then computed by summing up the partial delay bounds. For instance, the tandem of Figure 4 can be cut in two different ways, i.e. placing the cut *before* or *after* node 2. This way, two different end-to-end delay bounds can be computed, call them V^a and V^b, both using LUDB for the sub-tandems. More specifically, cutting the tandem *after* node 2 yields the following results (see [11] for the computations):

If $R^1 + \rho_{(2,3)} < R^2$,

$$V_1^a = \theta^1 \cdot \left[1 + \frac{\rho_{(1,3)}}{R^3} \cdot \left(1 + \frac{\rho_{(2,3)}}{R^2}\right)\right] + \theta^2 \cdot \left(1 + \frac{\rho_{(2,3)} + \rho_{(1,3)}}{R^3}\right) + \theta^3 + \frac{\sigma_{(1,2)}}{R^3}\left(\frac{\rho_{(2,3)}}{R^2} + \frac{R^3 + \rho_{(1,3)}}{R^1}\right) + \frac{\sigma_{(1,3)}}{R^3} \cdot \left(1 + \frac{\rho_{(2,3)}}{R^2}\right) + \frac{\sigma_{(1,3)}}{R^1} + \frac{\sigma_{(2,3)}}{R^2} \cdot \left(1 + \frac{R^2 + \rho_{(1,3)}}{R^3}\right) \quad (10)$$

Otherwise,

$$V_2^a = \theta^1 \cdot \left[1 + \frac{\rho_{(1,3)}}{R^3} \cdot \left(1 + \frac{\rho_{(2,3)}}{R^2}\right)\right] + \theta^2 \cdot \left(1 + \frac{\rho_{(2,3)} + \rho_{(1,3)}}{R^3}\right) + \theta^3 + \frac{\sigma_{(1,2)}}{R^2} \cdot \left[\frac{\rho_{(2,3)}}{R^3} + \left(1 + \frac{\rho_{(1,3)}}{R^3}\right) \cdot \left(1 + \frac{\rho_{(2,3)}}{R^1}\right)\right] + \frac{\sigma_{(1,3)}}{R^2} \cdot \left(1 + \frac{\rho_{(2,3)}}{R^2} + \frac{R^2 + \rho_{(2,3)}}{R^3}\right) + \frac{\sigma_{(2,3)}}{R^2} \cdot \left(1 + \frac{R^2 + \rho_{(1,3)}}{R^3}\right) \quad (11)$$

On the other hand, cutting the tandem *before* node 2 yields the following results:

If $R^3 + \rho_{(1,2)} < R^2$,

$$V_1^b = \theta^1 \cdot \left(1 + \frac{\rho_{(1,2)}}{R^2} + \frac{\rho_{(1,3)}}{R^3}\right) + \theta^2 + \theta^3 + \frac{\sigma_{(1,2)}}{R^1}\left(1 + \frac{\rho_{(1,3)}}{R^3}\right) + \frac{\sigma_{(1,2)}}{R^2} + \frac{\sigma_{(1,3)}}{R^1}\left(1 + \frac{\rho_{(1,2)}}{R^2}\right) + \frac{\sigma_{(1,3)}}{R^3} + \frac{\sigma_{(2,3)}}{R^3} \quad (12)$$

Otherwise,

$$V_2^b = \theta^1 \cdot \left[1 + \frac{\rho_{(1,2)}}{R^2} + \frac{\rho_{(1,3)}}{R^2} \cdot \left(1 + \frac{\rho_{(2,3)}}{R^3}\right)\right] + \theta^2 + \theta^3 + \frac{\sigma_{(1,2)}}{R^1} \cdot \left[1 + \frac{\rho_{(1,3)}}{R^2} \cdot \left(1 + \frac{\rho_{(2,3)}}{R^3}\right)\right] + \frac{\sigma_{(1,2)}}{R^2} + \frac{\sigma_{(1,3)}}{R^1} \cdot \left(1 + \frac{\rho_{(1,2)}}{R^2}\right) + \frac{\sigma_{(1,3)}}{R^2} \cdot \left(1 + \frac{\rho_{(2,3)}}{R^3}\right) + \frac{\sigma_{(2,3)}}{R^2} \cdot \left(1 + \frac{\rho_{(2,3)}}{R^3}\right) \quad (13)$$

Now, $V = V^a \wedge V^b$ is an end-to-end delay bound for the tagged flow. One can see through straightforward algebraic manipulations that both V^a and V^b can actually be the minimum, depending on the actual values of the nodes and flows parameters.

4.1 Tightness of the LUDB

Assessing whether LUDB yields tight bounds is made particularly challenging by the fact that a method for computing the WCD in FIFO networks is still missing. In a previous work of ours, [10], LUDB was applied to *sink-tree* tandems, such as the one shown in Figure 6 which are in fact nested tandems. In sink-tree tandems, all flows are of the kind (j, N), $1 \le j \le N$, i.e. they are nested into one another progressively. For this class of tandems, we showed that the LUDB (which can be computed in a closed form) is actually *equal* to the WCD. The proof was obtained by constructing a scenario where a bit of the tagged flow experiences a delay equal to the LUDB itself.

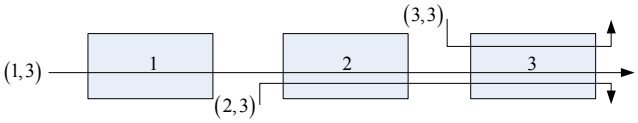

Figure 6. A sink-tree tandem

As far as non-nested tandems are concerned, in [11], we showed that this method yields better results compared to per-node analysis. However, we also observed that breaking the end-to-end analysis, i.e. computing and summing *partial* delay bounds, is likely to lead to loose end-to-end delay bounds. In fact, this entails assuming independent worst-case scenarios at each sub-tandem, which cannot take place simultaneously. The alert reader will notice that a similar argument has been used in the past to prove that the *pay burst only once* principle holds for single flows in per-flow scheduling networks (see e.g. [3] for some discussion on this topic). Broadly speaking, cutting a tandem into sub-tandems is certainly not as bad as cutting it into single nodes (as it is done in per-node analysis), but mostly because you need *less* cuts to obtain the same task (one instead of two, in the above example).

One question that remains open is whether, at least when end-to-end analysis is possible (i.e., in nested tandems), the LUDB is *always* equal to the WCD. In the next section, we show that this is not the case.

5. A COUNTEREXAMPLE

Hereafter, we show by counterexample that the LUDB may be larger than the WCD, even when end-to-end analysis is possible. Since we do not know how to compute the WCD, we can only prove this assertion by computing a *smaller* delay bound than the LUDB. The procedure is the following: consider a tandem T , and call W its WCD. Now, assume you are able to build tandem $\overline{T}$, such that its WCD $\overline{W}$ is *no smaller than* W , i.e. $\overline{W} \ge W$. Now, any delay bound (e.g., the LUDB) is no smaller than the WCD by definition. Thus, if V and $\overline{V}$ are delay bounds for T and $\overline{T}$, then it is $\overline{V} \ge W$, i.e. $\overline{V}$ is obviously a delay bound for T . However, if we find cases when $\overline{V} < V$, we can prove that $V > W$.

The property that allows us to build such a tandem $\overline{T}$ from a given tandem T is called *Flow Extension (FE)*. We first explain it, and then exploit it to construct simple counterexamples.

Hereafter, we denote as $A_{(i,j)}^k(t)$ the Cumulative Arrival Function (CAF) for flow (i,j) at node k , and with $D_{(i,j)}^k(t)$ the Cumulative Departure Function (CDF) for flow (i,j) at node k . Further-

more, we denote with $A^k(t)$ and $D^k(t)$ the *total* CAF and CDF at node k.

Define a *scenario* g for an N-node tandem as:

1) a set of CAFs for all the flows $(i,j) \subseteq (1,N)$ at their entry node, $A^i_{(i,j)}(t)$;
2) a set of "node behaviors", i.e. the way each node i, $1 \le i \le N$, transforms its CAF $A^i(t)$ into its CDF $D^i(t)$, according to the related service curve inequality $D^i(t) \ge \left[A^i \otimes \beta^i\right](t)$. As for the latter, we can describe a node behavior by means of a non-negative *lead function* $L^i(t)$, which is such that $D^i(t) = \left[A^i \otimes \beta^i\right](t) + L^i(t)$. Note that $L^i(t)$ is not necessarily wide-sense increasing.

In order for a scenario to be *feasible*, each CAF has to be compatible with the related arrival curve constraint, $A^i_{(i,j)}(t) - A^i_{(i,j)}(s) \le \alpha_{(i,j)}(t-s)$. Furthermore, each lead function has to verify $L^i(t) \le A^i(t) - \left[A^i \otimes \beta^i\right](t)$ in order for node i to have a causal behavior.

Theorem 5.1 (Flow Extension, FE)

Let T be a tandem of N nodes, in which there is a flow $(j, N-1)$. Call $\overline{T}$ the tandem obtained from T by "extending" flow $(j, N-1)$, i.e. by substituting it with flow (j,N), all else being equal. Call d and $\overline{d}$ the WCD for the tagged flow in T and $\overline{T}$. Then, it is $\overline{d} \ge d$.

Proof

Call Γ the set of all feasible scenarios in a tandem. Throughout this proof, we express the fact that a quantity depends on scenario $g \in \Gamma$ by using the conditional notation $|_g$, i.e. $A^i_{(i,j)}(t)|_g$ denotes the CAF of flow (i,j) at node i under scenario g.

Call $d^i(t)|_g$ the delay experienced at node i by a bit of the tagged flow entering a generic N-node tandem at time t in scenario g. The WCD d is defined as follows:

$$d = \max_{g \in \Gamma} \left\{ \max_{t \ge 0} \left[\sum_{i=1}^{N} d^i(t)|_g \right] \right\} \tag{14}$$

Call $\Phi \subset \Gamma$ the subset of scenarios where $L^N(t) = 0$, i.e. those for which node N is *lazy*. We first show that at least one worst-case scenario is included in Φ, i.e.:

$$d = \max_{g \in \Phi} \left\{ \max_{t \ge 0} \left[\sum_{i=1}^{N} d^i(t)|_g \right] \right\} \tag{15}$$

Assume by contradiction that:

$$d > \max_{g \in \Phi} \left\{ \max_{t \ge 0} \left[\sum_{i=1}^{N} d^i(t)|_g \right] \right\} \tag{16}$$

and call $x \in \Gamma \setminus \Phi$ the scenario where d is achieved. Consider now the scenario $y \in \Phi$, which only differs from x because $L^N(t) = 0$. It is obviously $d^i(t)|_y = d^i(t)|_x$, $1 \le i \le N-1$, and $A^N(t)|_y = A^N(t)|_x$. However, if node N is lazy in y and not in x, it is $D^N(t)|_y \le D^N(t)|_x$, hence $D^N_{(1,N)}(t)|_y \le D^N_{(1,N)}(t)|_x$ since the node is FIFO, and $d^N(t)|_y \ge d^N(t)|_x$. Thus, we have found a scenario $y \in \Phi$ in which a delay at least no smaller than d is achieved, which contradicts (16).

Having said this, we move to comparing T and $\overline{T}$, limiting ourselves to the subset of scenarios in which the last node is lazy. Whenever needed, we use the same symbol to denote the same quantities in T and $\overline{T}$, adding a bar to the latter ones in order to distinguish them. Consider now a generic scenario $g \in \Gamma$ for tandem T, and define the *corresponding* scenario $\overline{g}$ in $\overline{T}$ as the one with the same set of CAFs at the entry nodes of all flows, and the same set of lead function at all nodes. Clearly, if the scenario is feasible in T, it is also feasible in $\overline{T}$, since flows and nodes are subject to the same constraints. However, in tandem $\overline{T}$, flow $(j, N-1)$ is *extended* up to node N. This is exactly like adding, as an input to node N, a "virtual" flow $(N-1, N)$, with $A^N_{(N-1,N)}(t) = D^{N-1}_{(j,N-1)}(t)$.

For a scenario $g \in \Phi$ in T, the corresponding scenario $\overline{g} \in \overline{\Phi}$ is such that:

$$\overline{d^i}(t)\big|_{\overline{g}} \ge d^i(t)\big|_g, \ 1 \le i \le N. \tag{17}$$

In fact, equality holds in (17) for $1 \le i \le N-1$, since the two scenarios are the same up to node $N-1$ included. However, the input at node N in $\overline{T}$ is $\overline{A^N}(t) = A^N(t) + A^N_{(N-1,N)}(t)$, where $A^N_{(N-1,N)}(t)$ is a wide-sense increasing function. Now, since node N is lazy and FIFO, the delay of each bit in $A^N(t)$ cannot be lower than in T, thus $d^N(t)|_{\overline{g}} \ge d^N(t)|_g$.

Now, for any scenario $g \in \Gamma_T$ there exists a scenario $\overline{g} \in \Gamma_{\overline{T}}$ in which the end-to-end delay of a bit of the tagged flow entering at time t in tandem $\overline{T}$ is larger than (or equal to) the one in tandem T. Therefore, the same inequality also holds between the respective WCDs, i.e. $\overline{d} \ge d$. □

We now show how to exploit FE to compute *smaller* bounds than the LUDB.

Example 5.2

Consider the two-node tandem T shown in Figure 7, left.

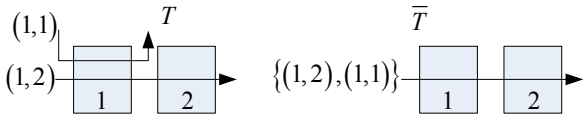

Figure 7. Two simple tandems. The one on the right is obtained by applying FE to the one on the left.

Build the corresponding tandem $\overline{T}$ according to FE (shown in the same figure on the right), for which it is $\overline{W} \ge W$. Consider now what *delay bound* we can compute through LUDB in both tandems. In T, it is the following:

$$V = \begin{cases} \sum_{i=1}^{2} \theta^i + \dfrac{\sigma_{(1,1)}}{R^1} + \dfrac{\sigma_{(1,2)}}{R^2} & R^2 + \rho_{(1,1)} < R^1 \\ \sum_{i=1}^{2} \theta^i + \dfrac{\sigma_{(1,1)}}{R^1} + \dfrac{\sigma_{(1,2)}}{R^1 \cdot \dfrac{R^2}{R^2 + \rho_{(1,1)}}} & R^2 + \rho_{(1,1)} \ge R^1 \end{cases} \tag{18}$$

provided that the following provisioning inequalities hold:

$$R^1 \ge \rho_{(1,1)} + \rho_{(1,2)}, \ R^2 \ge \rho_{(1,2)} \tag{19}$$

Otherwise it is infinite.

On the other hand, the LUDB for the tagged flow in $\overline{T}$ is:

$$\overline{V} = \sum_{i=1}^{2} \theta^i + \frac{\sigma_{(1,1)} + \sigma_{(1,2)}}{R^1 \wedge R^2}, \tag{20}$$

provided that the following provisioning inequalities holds:

$$R^1 \ge \rho_{(1,1)} + \rho_{(1,2)}, \ R^2 \ge \rho_{(1,1)} + \rho_{(1,2)}, \tag{21}$$

otherwise it is infinite. Note that the second inequality in (21), related to node 2, is more constraining than the corresponding one in (19).

Now, $\wedge(V, \overline{V})$ is a delay bound in T. However, it is easy to see that $\overline{V} < V$ in some cases. Table 1 reports the comparison between V and $\overline{V}$ in the five different regions in which the rate inequali-

ties included in expressions (18)-(21) divide the plan R^1OR^2 (also shown in Figure 8).

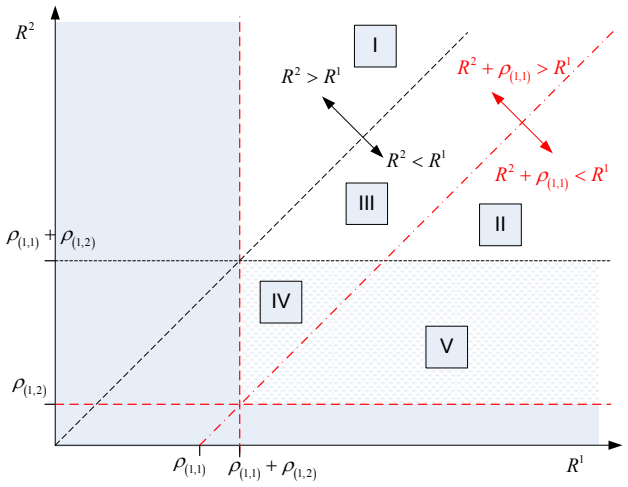

Figure 8. Regions of the plan R^1OR^2 and related inequalities.

In region I, $\overline{V} < V$. Thus, the following set of inequalities hold:

$$V \geq W, \quad \overline{V} \geq \overline{W}, \quad \overline{W} \geq W, \quad \overline{V} < V \qquad (22)$$

An immediate consequence of (22) is that $V > W$, i.e. *the LUDB is not the WCD in that case*.

Furthermore, note that in region III, the rate inequalities are not sufficient to decide whether $\overline{V} < V$ or $\overline{V} \geq V$: in fact, both can occur depending on the values of the parameters. Again, this means that the LUDB is not necessarily the WCD in that region too. □

Now, when LUDB is applied to a nested tandem, the entire set of all the "good" end-to-end service curves that can be computed using Theorem 2.1 and convolution is explored, and a global minimum is computed. This means that no better bounds can be computed by relying on Theorem 2.1 alone. However – quite surprisingly – this is proved not to be sufficient for computing the WCD. A likely cause for this is that not all the necessary information is retained in the equivalent service curves computed through Theorem 2.1.

Consider, for instance, a single rate-latency node traversed by two leaky-bucket shaped flows, as in the example shown in Figure 1, and assume that the arrival curve of the two flows are $\alpha_i = \gamma_{\sigma_i,\rho_i}$, $1 \leq i \leq 2$. The LUDB for flow 1 is computed as the solution of the following trivial optimization problem:

$$d = \min_{s \geq 0} \left\{ s + \theta + \frac{\sigma_2}{R} + \left\lceil \frac{\sigma_1 - R \cdot s}{R - \rho_2} \right\rceil^+ \right\}$$

The minimum is achieved when $s = \sigma/R$, and it is equal to $V = \theta + (\sigma_1 + \sigma_2)/R$. This is also the WCD for flow 1, since it is attained by its σ_1^{th} bit in the following worst-case scenario:

a) both flows are *greedy:* $A_i(t) = \alpha_i(t)$, i.e. their CAFs are equal to their respective arrival curves. However, the burst of flow 2 arrives *just before* that of flow 1.

b) the node is lazy.

Call $D_1(t)$ the CDF for flow 1 obtained in the above scenario, shown in Figure 9 as a thicker dashed line. Let us compare it to the curves $D_1'(t,s)$ obtained by convolving the greedy CAF of flow 1 with each equivalent service curve derived through Corollary 2.2, therein including the "optimum" one. These are shown as thinner lines in the same figure, for various values of s, and they represent *lower bounds* to any CDF that can be obtained from that CAF, by definition of (equivalent) service curve. However, one can easily see that $\nexists s : D_1(t) = D_1'(t,s)$. This seems to suggest that the $D_1'(t,s)$ might not be tight lower bounds themselves. This, in turn, would imply that each equivalent service curve alone cannot describe the behavior of a FIFO node with the necessary accuracy.

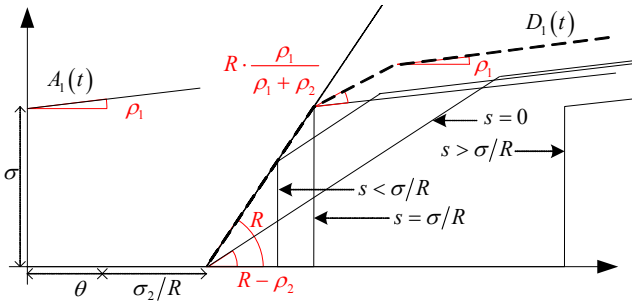

Figure 9. CDFs obtained using equivalent service curves

Table 1. Different regions of the plan R_1OR_2 and related end-to-end delay bounds

Region	Rate Inequalities	V	$\overline{V}$	Comparison
I	$R^1 \geq \rho_{(1,1)} + \rho_{(1,2)}$, $R^2 > R^1$	$\sum_{i=1}^{2} \theta^i + \frac{\sigma_{(1,1)}}{R^1} + \frac{\sigma_{(1,2)}}{R^1 \cdot \frac{R^2}{R^2 + \rho_{(1,1)}}}$	$\sum_{i=1}^{2} \theta^i + \frac{\sigma_{(1,1)} + \sigma_{(1,2)}}{R^1}$	$\overline{V} < V$
II	$R^2 \geq \rho_{(1,1)} + \rho_{(1,2)}$, $R^2 + \rho_{(1,1)} < R^1$	$\sum_{i=1}^{2} \theta^i + \frac{\sigma_{(1,1)}}{R^1} + \frac{\sigma_{(1,2)}}{R^2}$	$\sum_{i=1}^{2} \theta^i + \frac{\sigma_{(1,1)} + \sigma_{(1,2)}}{R^2}$	$V \leq \overline{V}$
III	$R^2 \geq \rho_{(1,1)} + \rho_{(1,2)}$, $R^2 < R^1$, $R^2 + \rho_{(1,1)} \geq R^1$	$\sum_{i=1}^{2} \theta^i + \frac{\sigma_{(1,1)}}{R^1} + \frac{\sigma_{(1,2)}}{R^1 \cdot \frac{R^2}{R^2 + \rho_{(1,1)}}}$	$\sum_{i=1}^{2} \theta^i + \frac{\sigma_{(1,1)} + \sigma_{(1,2)}}{R^2}$	It depends
IV	$R^1 \geq \rho_{(1,1)} + \rho_{(1,2)}$, $R^2 < \rho_{(1,1)} + \rho_{(1,2)}$, $R^2 + \rho_{(1,1)} \geq R^1$	$\sum_{i=1}^{2} \theta^i + \frac{\sigma_{(1,1)}}{R^1} + \frac{\sigma_{(1,2)}}{R^1 \cdot \frac{R^2}{R^2 + \rho_{(1,1)}}}$	∞	$V < \overline{V}$
V	$R^2 \geq \rho_{(1,2)}$, $R^2 < \rho_{(1,1)} + \rho_{(1,2)}$ $R^2 + \rho_{(1,1)} \geq R^1$	$\sum_{i=1}^{2} \theta^i + \frac{\sigma_{(1,1)}}{R^1} + \frac{\sigma_{(1,2)}}{R^2}$	∞	$V < \overline{V}$

5.1 Practical applications of Flow Extension

Beside being useful to prove the limitations of Theorem 2.1, FE can also be exploited to compute improved delay bounds. However, its practical usefulness is limited for at least two reasons. The first one is represented by the topology restrictions required in order to apply Theorem 5.1 (i.e., that there is a flow in the tandem that leaves at node $N-1$). Second, in order for it to be of any practical use, it requires that the last node be *overprovisioned*. With reference to the previous example, we can observe that, if $R^2 \in \left[\rho_{(1,2)}, \rho_{(1,1)} + \rho_{(1,2)}\right[$, i.e. in regions IV and V, the WCD in tandem $\overline{T}$ is infinite, and thus FE is useless in this case.

This said, we can still find some useful generalization of Theorem 5.1. The first one is that, given a tandem T, and a set of *extensible* flows $S \equiv \{(j, N-1) \subset (1, N)\}$, FE can in fact be applied by extending any (non empty) *subset* of flows $s \subseteq S$. Thus we can build up to $2^{|S|}-1$ different tandems $\overline{T}$, for each one of which a delay bound can be computed, possibly improving on the LUDB in T for some value of the nodes and flows parameters. However, the more flows are in s, the more constraining the provisioning inequalities at node N must be, in order for the related bound in $\overline{T}$ to be finite. More specifically, the required inequality is the following:

$$\sum_{(i,N)\subset(1,N)} \rho_{(i,N)} + \sum_{(i,N-1)\in s} \rho_{(i,N-1)} \le R^N \qquad (23)$$

Thus, the amount of overprovisioning at node N may act as a constraint on the number of *effective* ways in which FE can be applied (which can therefore be smaller than $2^{|S|}-1$ in practice).

The second generalization is that FE can be applied *more than once* to the same tandem, while obviously tightening the provisioning inequalities at each iteration. For instance, in the tandem shown in Figure 10, above, FE can be applied a first time by extending flow $(1,2)$. After convolving the service curves of node 2 and 3, it can then be applied again, extending flow $(1,1)$ up to node 3.

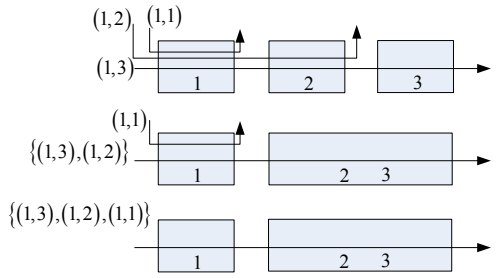

Figure 10. Nested tandem and related FE transformations

Hereafter, we report another example for FE, this time related to a non-nested tandem.

Example 5.3

Consider the non-nested tandem of Figure 4. We apply FE to it, by extending flow $(1,2)$, and derive the following delay bound:

If $R^1 + \rho_{(2,3)} < \left(R^2 \wedge R^3\right)$, then:

$$\overline{V} = \sum_{i=1}^{3} \theta^i + \frac{\sigma_{(2,3)}}{R^2 \wedge R^3} + \frac{\sigma_{(1,3)} + \sigma_{(1,2)}}{R^1} \qquad (24)$$

Otherwise

$$\overline{V} = \sum_{i=1}^{3} \theta^i + \frac{\sigma_{(2,3)}}{R^2 \wedge R^3} + \frac{\sigma_{(1,3)} + \sigma_{(1,2)}}{\left(R^2 \wedge R^3\right) \cdot \dfrac{R^1}{R^1 + \rho_{(2,3)}}} \qquad (25)$$

Both (24) and (25) hold provided that $R^3 \ge \rho_{(1,3)} + \rho_{(2,3)} + \rho_{(1,2)}$.

Note that, unlike in V_a and V_b, in $\overline{V}$ each burst $\sigma_{(i,j)}$ appears exactly once. By comparing them with (10)-(13), it is easy to identify regions in which $\overline{V} < \left(V_a \wedge V_b\right)$. For instance, if $R^i = 3$, $\theta^i = 1$, $1 \le i \le 3$, and $\sigma_{(i,j)} = 3$, $\rho_{(i,j)} = 1$, for all flows, we obtain $\overline{V} = 20/3$, $V_2^a = 101/9$, $V_2^b = 92/9$, so that $\overline{V} \simeq 0.65 \cdot \left(V_a \wedge V_b\right)$. □

6. A LOWER BOUND ON THE WORST-CASE DELAY

In order to assess how tight the upped bound V that we can compute through LUDB or FE is, we compute a *lower* bound v on the worst-case delay. The interval $[v, V]$ includes the WCD by definition, and its width $V - v$ is a measure of the uncertainty on the worst-case delay itself. Such a method has already been used to assess the tightness of a network calculus bound in [18].

Now, any *attainable* end-to-end delay is by definition a lower bound on the worst-case delay, the latter being in fact the maximum attainable delay. Therefore, we heuristically design a scenario which leads to a "large" end-to-end delay for the tagged flow. The heuristics used are the same that were proved in [10] to actually represent *the* worst-case scenario for sink-tree tandems. While this does not imply that the same holds for generic tandems, it nonetheless provides a good motivation. We rely on the following three high-level heuristics:

a) All nodes are lazy, i.e. they delay each bit as long as they can
b) The tagged flow $(1, N)$ sends its whole burst $\sigma_{(1,N)}$ at time $t = 0$ and then stops. Therefore, the $\sigma_{(1,N)}$ th bit of the tagged flow experiences a larger delay than the other $\sigma_{(1,N)} - 1$.
c) Every cross flow (i, j) sends "as much traffic as possible", so as to delay the $\sigma_{(1,N)}$ th bit of the tagged flow.

We measure the delay experienced by the $\sigma_{(1,N)}$ th bit of the tagged flow under these hypotheses.

Let us take a closer look at hypothesis c) above. Call a^x, b^x the time instants when the *first* and the *last* bit of the tagged flow arrive at node x. For instance, it is $a^1 = b^1 = 0$, while $a^x < b^x$ for $x > 1$, since all nodes are lazy. Hypothesis c) implies that

$$A^i_{(i,j)}\left(b^i\right) - A^i_{(i,j)}\left(a^i\right) = \alpha_{(i,j)}\left(b^i - a^i\right) \qquad (26)$$

for each flow (i, j). However, there are infinite CAFs that verify (26). For instance, one is the *greedy* CAF, $A^i_{(i,j)}(t) = \alpha_{(i,j)}\left(t - a^i\right)$, while another one is $A^i_{(i,j)}(t) = F^i_{(i,j)}(t)$, with:

$$F^i_{(i,j)}(t) = \begin{cases} \rho_{(i,j)} \cdot \left(t - a^i\right)^+ & t < b^i \\ \rho_{(i,j)} \cdot \left(b^i - a^i\right) + \sigma_{(i,j)} & t = b^i \end{cases} \qquad (27)$$

which we call *delayed greedy* CAF, in which the flow sends its burst $\sigma_{(i,j)}$ *just before* the $\sigma_{(1,N)}$ th bit of the tagged flow arrives at node i, as shown in Figure 11.

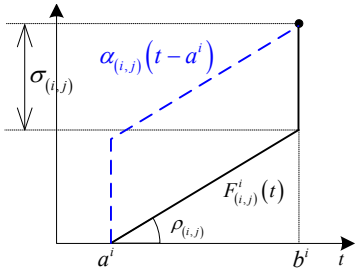

Figure 11. Cumulative arrival functions for flow *(i,j)*

Under the hypotheses of the system model, if all the CAFs for the

cross flows are either greedy or delayed greedy, both the *total* CAF and the CAF of the tagged flow at each node are piecewise linear, and therefore easy to handle algorithmically. We have designed and coded an algorithm that manipulates piecewise linear CAFs and rate-latency service curves, computing the respective CDFs under the lazy-node hypothesis (similar to what was done in [16] for blind-multiplexing networks). The algorithm takes into account the FIFO multiplexing and de-multiplexing of flows, thus allowing one to separate the contribution of each flow to the total CAF at a node.

It turns out that, depending on the values associated to the nodes and flows parameters, using either the greedy or the delayed greedy CAF for the cross flows actually leads to different delays, and it is not always possible to establish which of the two is larger without actually running the scenarios. Therefore, in order to compute the largest possible lower bound on the delay, one needs to test up to 2^M different scenarios, M being the number of cross-flows. However, the computations required for evaluating a single scenario are indeed lightweight, so that computing v is normally faster than computing the LUDB.

In order to give a first example of the effectiveness of the proposed algorithm, we apply it to the two examples of Section 5, in which improved bounds were computed through FE. The results are:

- Example 5.2: $v = \wedge\left(V, \overline{V}\right)$ in regions I, II, V, while in regions III and IV it is

$$v = \sum_{i=1}^{2} \theta^i + \frac{\sigma_{(1,1)}}{R^1} + \frac{\sigma_{(1,2)}}{R^2} < \wedge\left(V, \overline{V}\right)$$

- Example 5.3: when (24) holds, it is always $v = \overline{V}$, otherwise in (25) it is $v = \overline{V}$ only when $R^2 \le R^3$, otherwise it is:

$$v = \sum_{i=1}^{3} \theta^i + \frac{\sigma_{(2,3)}}{R^3} + \frac{\sigma_{(1,3)}}{R^3 \cdot \dfrac{R^1}{R^1 + \rho_{(2,3)}}} + \frac{\sigma_{(1,2)}}{R^2 \cdot \dfrac{R^1}{R^1 + \rho_{(2,3)}}} < \wedge\left(V, \overline{V}\right)$$

This further motivates us to think that, on one hand, FE is effective in complementing LUDB, and, on the other hand, that the heuristics behind the computation of the lower bound are effective as well. In the next section, we evaluate the LUDB and the lower bound in two non-trivial case studies.

7. NUMERICAL EVALUATION

We have developed a tool, called DEBORAH (DElay BOund Rating AlgoritHm), for computing both upper and lower delay bounds in tandems, which is available for download at [17]. The tool is written in C++, and it takes a text file as an input, which contains the rate and latency $\left(R^i, \theta^i\right)$ of each node $1 \le i \le N$ and the leaky-bucket parameters $\left(\sigma_{(i,j)}, \rho_{(i,j)}\right)$ for each flow. At the moment of writing, the tools only computes the LUDB for nested tandems, whereas the lower bound algorithm shown in Section 6 works for both nested and non-nested tandems. The LUDB is computed by separating the original P-LP problem into a number of simplexes, and solving each simplex separately. Each simplex has M variables, M being the number of $s_{(i,j)}$ variables associated to each cross flow. The number of constraints C is upper bounded by the following expression:

$$C \le M + \frac{(M+1) \cdot M}{2} - 1$$

The first M constraints are simply $s_{(i,j)} \ge 0$ for each cross flow. The other constraints are required to isolate each single piece of the piecewise linear objective function, and they are always no more than the sum of the first M naturals (much less on average).

A thorough analysis of the computational complexity of the LUDB computation is part of the ongoing work. As a first, preliminary observation, we report that the number of required simplexes grows fast with the number of flows and nodes in the tandem. Furthermore, the number of simplexes is influenced not only by the *number* of cross-flows, but also on the way they are nested into each other. In fact, it may range between M, achieved in a tandem with 1-hop persistent cross-flows, such as the one analyzed in [9], to $M!$, which is achieved in a sink-tree tandem (although the LUDB can actually be computed in a closed-form in both cases, without the aid of a software tool). However, in our experiments we found that a high percentage of those simplexes (roughly an average of 90%, in the cases we analyzed) are actually unfeasible, and can be easily identified as such, thus saving a considerable computation time.

Hereafter, we report computations related to two case studies.

***Case study* 1**

We analyze a relatively complex nested tandem of 15 nodes and 17 flows, shown in Figure 12, whose nodes and flows parameters are reported in Table 2 and Table 3.

For this case study, the DEBORAH tool outputs $V = 217.386$. The LUDB was computed by solving 19440 simplexes, 17604 of which turned out to be unfeasible. The overall time taken was 6.92 seconds on a 3.0GHz single-core processor, nearly 85% of which spent within the simplex solver itself. As a cross-check, we also tried FE, by extending flow (12,14) (note that flow (14,14) could not be extended due to the rate constraints at node 15). This yielded $\overline{V} = 232.541 > V$, in similar computation time. The lower bound computation algorithm yielded $v = 190.483$, in less than 5 s. The gap is $(V - v)/V \cong 12.4\%$, which further confirms that the LUDB is a good estimate of the WCD. □

Table 2. Node parameters

node	R	θ
1	70	0.3
2	10	0.2
3	40	0.1
4	40	0.1
5	60	0.2
6	55	0.1
7	7	0.2
8	60	0.3
9	60	0.1
10	10	0.2
11	30	0.1
12	40	0.3
13	45	0.1
14	45	0.3
15	7	0.2

Table 3. Flow parameters

flow	σ	ρ
1,15	200	1
1,1	100	60
1,3	200	3
1,8	100	2
3,3	200	30
4,4	400	30
4,7	300	2
5,5	300	50
6,6	200	45
8,8	100	55
9,9	300	55
9,11	200	2
11,11	200	20
12,12	100	30
12,14	100	3
13,13	200	40
14,14	100	40

***Case study* 2**

We analyze a tandem of N nodes, traversed by the tagged flow $(1, N)$ and by one-hop persistent cross-flows (i, i), $1 \le i < N$, shown in Figure 13. We assume that all flows have the same leaky-bucket arrival curve, with $\sigma = 5$ and $\rho = 4$. All nodes have the same rate-latency service curve, with $\theta = 1$ and $R = 2\rho/U$, U ranging from 20% to 100%. The LUDB expression for that tandem is available in a closed form, and it is equal to (see [9], Theorem 2):

$$V = N \cdot \left[\theta + \frac{U \cdot \sigma}{\rho} \cdot \left(\frac{1}{2} + \frac{1}{2-U} \right) \right]$$

Figure 14 shows the gap between the LUDB and the lower bound as a function of the number of nodes and for various values of U. As the figure shows, the gap increases with both N and U. However, it tends to reach a limit value as N grows higher. While the exact quota of the gap depends on the actual parameter values, the same behavior is always observed.

Note that we can apply FE to the above tandem when $U < 100\%$. For instance, when $N = 8$, $U = 20\%$ all the cross flows can be extended to the last node, thus yielding a sink-tree tandem with a tagged flow $(1,N) \equiv (2\sigma, 2\rho)$ and cross-flows $(i,N) \equiv (\sigma, \rho)$, $2 \le i \le 8$, for which the LUDB can be computed in a closed form applying the formula in [10]. As shown in Table 4, this reduces the gap of about 40%. □

Table 4. Gap between the LUDB and the lower bound for with N=8 and U=20%

tandem	LUDB	Lower Bound	Gap
original	10.111	9.125	9.75%
with FE	9.673		5.66%

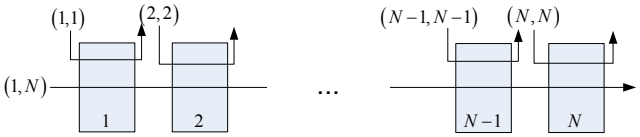

Figure 13 – Case-study nested tandem

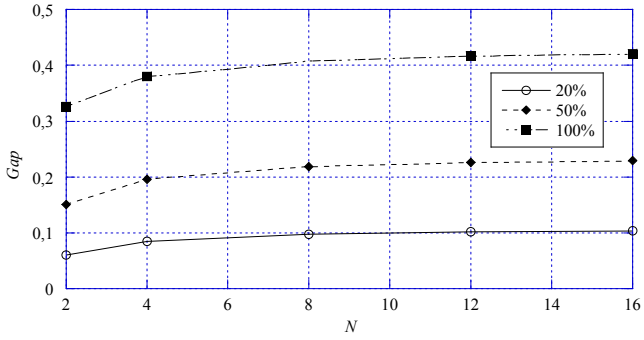

Figure 14. Gap between the LUDB and the lower bound in the case-study tandem

8. CONCLUSIONS AND FUTURE WORK

In this paper we have shown that the current Network Calculus theorems related to FIFO multiplexing are not sufficient for computing the worst-case delay in tandem networks. The best delay bound that can be computed, i.e. the *least upper delay bound*, can sometimes be improved upon, even in very simple cases. We have shown this introducing a method – called Flow Extension, that allows one to compute delay bounds by exploiting topological properties of tandems. We have then addressed the question of how close the upper bounds are to the (unknown) worst-case delay. We have devised an algorithm that computes *lower* bounds on the worst-case delay. Our preliminary analysis shows that, at least for nested tandems, the upper and lower bounds appear to be reasonably close.

This work is being extended at the time of writing. Specifically, we are currently extending DEBORAH so as to take into account non nested tandems. Furthermore, we are evaluating the tightness of the LUDB in a broader set of scenarios. Finally, we are developing heuristics to approximate the LUDB in very large nested tandems, where solving a too large number of simplexes would not be computationally affordable.

9. REFERENCES

[1] R. Braden, D. Clark, S. Shenker, "Integrated Services in the Internet Architecture: an Overview", IETF RFC 1633, June 1994.

[2] S. Blake, *et al.*, "An Architecture for Differentiated Services," IETF RFC 2475, 1998.

[3] J.-Y. Le Boudec, P. Thiran, Network Calculus, Springer-Verlag LNCS vol. 2050, 2001.

[4] E. Rosen, A. Viswanathan, R. Callon, "Multiprotocol Label Switching Architecture", IETF RFC 3031, January 2001

[5] R.L. Cruz. "A calculus for network delay, part i: Network elements in isolation". IEEE Transactions on Information Theory, Vol. 37, No. 1, March 1991, pp. 114-131.

[6] R.L. Cruz. "A calculus for network delay, part ii: Network analysis". IEEE Transactions on Information Theory, Vol. 37, No. 1, March 1991, pp. 132–141.

[7] R. Agrawal, R. L. Cruz, C. Okino, R. Rajan, "Performance Bounds for Flow Control Protocols," IEEE/ACM Transactions on Networking, Vol. 7, No. 3, June 1999, pp. 310-323.

[8] C. S. Chang, Performance Guarantees in Communication Networks, Springer-Verlag, New York, 2000.

[9] L. Lenzini, E. Mingozzi, G. Stea, "Delay Bounds for FIFO Aggregates: a Case Study", Elsevier Computer Communications Vol. 28 Issue 3, February 2005 pp. 287–299.

[10] L. Lenzini, L. Martorini, E. Mingozzi, G. Stea, "Tight End-to-end Per-flow Delay Bounds in FIFO Multiplexing Sink-tree Networks", Performance Evaluation, Vol. 63, October 2006, pp. 956-987.

[11] L. Lenzini, E. Mingozzi, G. Stea, "End-to-end Delay Bounds in FIFO-multiplexing Tandems", VALUETOOLS'07, Nantes (FR), October 23-25, 2007

[12] L. Lenzini, E. Mingozzi, G. Stea, "A Methodology for Computing End-to-end Delay Bounds in FIFO-multiplexing Tandems", to appear on Performance Evaluation

[13] M. Fidler, V. Sander, "A Parameter Based Admission Control for Differentiated Services Networks", Elsevier Computer Networks, Vol. 44, No 1, January 2004, pp. 463-479.

[14] R. L. Cruz. "Sced+: Efficient management of quality of service guarantees", proc. of IEEE Infocom'98, San Francisco (USA), 29 March-April 1998, pp. 625-634.

[15] J. C. R. Bennett, K. Benson, A. Charny, W. F. Courtney, J.-Y. Le Boudec, "Delay Jitter Bounds and Packet Scale Rate Guarantee for Expedited Forwarding", IEEE/ACM Trans. on Networking, Vol. 10, No. 4, August 2002, pp. 529-540.

[16] J. B. Schmitt, F. A. Zdarsky, "The DISCO Network Calculator - A Toolbox for Worst Case Analysis", Proc. of VALUETOOLS '06, Pisa, Italy. ACM, October 2006.

[17] Website of the Computer Networking Group at the University of Pisa, http://www.info.iet.unipi.it/cng/.

[18] G. Urvoy-Keller, G. Hèbuterne, Y. Dallery, "Traffic Engineering in a Multipoint-to-point network.", *IEEE JSAC,* Vol. 20, No. 4, May 2002, pp. 834-849