End-to-end Delay Bounds in FIFO-multiplexing Tandems

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.lenzini, e.mingozzi, g.stea}@iet.unipi.it

ABSTRACT

In this paper we address the problem of finding good end-to-end delay bounds for single leaky-bucket shaped flows subject to FIFO multiplexing in tandems of rate-latency nodes. More specifically, we focus on a methodology, called the Least Upper Delay Bound (LUDB) method, which is based on Network Calculus. The latter has already been used for computing delay bounds in tandems in which the path of the various flows are subject to particular restrictions. In this paper we generalize it to tandems traversed by flows following arbitrary paths. We show that such methodology yields better bounds than those obtained through both per-node analysis and comparable methods proposed in the literature.

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 "holy grail" of the future Internet is to support the provisioning of reliable real-time services on a wide scale. In order to achieve this goal, per-aggregate resource management is nowadays regarded as a mandatory choice. 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 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. Real-time services, however, require firm QoS guarantees, such as a bound on the end-to-end delay experienced by the packets of a flow. While it is well known how to compute the worst-case delay for a flow under per-flow resource management (see, for example, [3], Chapter 2), a general methodology for computing worst-case delays in networks employing per-aggregate resource management has not been devised yet. It goes without saying that knowledge of a worst-case delay would be beneficial for a large variety of purposes, e.g.: devising effective admission control procedures; computing the maximum network utilization (or the amount of overprovisioning required) under pre-specified delay constraints; defining aggregation schemes which are suitable for real-time traffic, etc. In short, it would lay a sound theoretical foundation for real-time traffic engineering practices.

In some recent works [12], [13], a methodology was proposed for computing *delay bounds*, i.e. upper bounds on the worst-case delay, for single flows in FIFO-multiplexing tandems, under reasonable assumptions on the nodes and flows behavior. The methodology is based on Network Calculus ([3], [8]-[11]), a theory for deterministic network performance analysis, and it allows one to relate the delay bound of a single flow traversing the tandem to the amount of resources provisioned for the aggregate and to the traffic at the ingress nodes. Such methodology is based on a known Network Calculus theorem that allows one to infer *per-flow* service curves from *per-aggregate* service curves at a *single node*. It basically consists in i) applying that theorem iteratively so as to obtain a *set of end-to-end service curves* for a flow, and ii) computing the *least upper delay bound*, i.e. the minimum among all the bounds which can be obtained from each end-to-end service curve. For that reason, henceforth we call it the LUDB methodology. In [13], the LUDB methodology has been applied to a *sink-tree* network (also called an *accumulation* or *multipoint-to-point* network, [5]-[7]), i.e. a network partitioned into a set of logical trees, rooted at egress nodes, so that a) bandwidth and buffer are provisioned per-tree at each node, and b) all traffic directed towards an egress node is routed through the related tree. It was proven in [13] that the delay bound thus computed is the *actual* worst-case delay. In [12], it is applied to a tandem network, in which the tagged flow is aggregated with a different interfering flow at each node. A closed-form delay bound has been computed, whose tightness however has not been assessed.

In this paper we generalize the LUDB methodology: more specifically, we first show that, in order for it to be applied directly, restrictions on how the path of the various flows are interleaved must be imposed: in fact, it can only be directly applied to *nested tandems*, i.e. tandems in which 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. Analyzing generic, non-nested tandems, instead, requires applying the LUDB methodology iteratively in order to compute *partial*, i.e. per sub-tandem, delay bounds, which are then summed up to compute the end-to-end delay bound. We then present an algorithm for partitioning a tandem into nested sub-tandems, and show how to apply the LUDB methodology to the sub-tandems, taking care so as to use the best possible arrival curves at each iteration.

Valuetools'07, October 23-25, 2007, Nantes, France.

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. We describe the LUDB methodology for nested tandems in Section 4, and generalize it to non-nested tandems in Section 5. In Section 6 we present a case study showing the effectiveness of the proposed approach. The related work is reviewed in Section 7. Finally, we report conclusions in Section 8.

2. NETWORK CALCULUS BACKGROUND

Network Calculus is a theory for deterministic network analysis [3],[8]-[11]. 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 $s \le t$. 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

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

for any $t \ge 0$. The flow is said to be guaranteed the (minimum) service curve β. 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 $\max\{0,x\}$. 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

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

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

$$d = \sum_{i=1}^{N} \theta^i + \frac{\sigma}{\underset{1 \le i \le N}{\wedge} \{R^i\}} \quad (3)$$

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

Furthermore, the *output arrival curve* after a node, i.e. the arrival curve that envelopes the flow at the exit of that node, can be computed as the *min plus deconvolution* of the flow's arrival curve and of the node's service curve, denoted as $(\alpha \odot \beta)(t)$.

2.1 FIFO multiplexing

Regarding FIFO multiplexing, a fundamental result, first derived in [10], is reported in [3], Chapter 6. Assume that two flows are FIFO multiplexed into the same network element, characterized by service curve β. Assume that α_2 is an arrival curve for flow 2. Then, the service received by flow 1 can be determined by computing its *equivalent* service curve $\beta_1^{eq}(t,\tau)$, 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 $\tau \ge 0$ 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.

In order to manipulate service curves under FIFO multiplexing we need some advanced results related to the composition of equivalent service curves. In some recent works ([13], [23]) a class of curves, namely *pseudoaffine* (or *quasi-concave*) curves, has been proved to effectively describe the service received by single flows in FIFO

multiplexing rate-latency nodes. We report some properties here, first shown in [13][1], that allow one to handle those curves efficiently. We call a *pseudoaffine* curve one which can be described as:

$$\pi = \delta_D \otimes \left[\bigotimes_{1 \le x \le n} \gamma_{\sigma_x, \rho_x} \right] \quad (4)$$

i.e., as a multiple affine curve shifted to the right. Note that, since affine curves are concave, (4) is equivalent to:

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

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. $\rho_\pi^* = \min_{x=1,\dots,n} (\rho_x)$. 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 1.

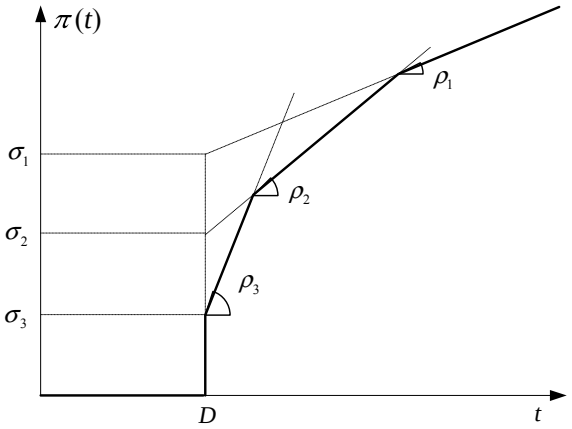

Figure 1 - Example of a three-stage pseudoaffine curve

A pseudoaffine curve π is said to be *non-redundant* if and only if there exists no leaky-bucket stage j such that:

$$\pi' = \delta_D \otimes \left[\bigotimes_{\substack{1 \le x \le n \\ x \ne j}} \gamma_{\sigma_x, \rho_x} \right] = \pi$$

i.e., no leaky-bucket stage can be removed without altering the service curve. In case a pseudoaffine curve is redundant, it is always possible to find a non-redundant equivalent description. We now list some properties related to pseudoaffine curves. The first two properties are related to how pseudoaffine curves are transformed by common operations, like convolution and application of Theorem 2.1.

Property 2.2 (closeness with respect to convolution):

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. Therefore, Π is closed with respect to the convolution operator $\otimes$.

Property 2.3 (equivalent service curve):

Let $\pi \in \Pi$ and $\alpha = \gamma_{\sigma,\rho}$. If $\rho_\pi^* \ge \rho$, then: $\forall \tau \ge 0$, $E(\pi, \alpha, \tau) \in \Pi$ and it is wide sense increasing. Furthermore, it is $\rho^*_{E(\pi,\alpha,\tau)} = \rho_\pi^* - \rho$.

Note that Property 2.3 cannot be generalized to the case of a more general arrival curve α. For example, if α is a piecewise linear (or multiple affine) curve (for instance, $\alpha = \gamma_{\sigma_1,\rho_1} \otimes \gamma_{\sigma_2,\rho_2}$, with $\rho_2 > \rho_\pi^* \ge \rho_1$), $E(\pi, \alpha, \tau)$ is not wide sense increasing for some $\tau \ge 0$.

We now show how to compute the delay bound and output arrival curve for a leaky-bucket shaped flow which is guaranteed a pseudoaffine service curve.

Property 2.4 (delay bound):

Let π be a pseudoaffine curve, with offset D and n leaky-bucket stages γ_{σ_x,ρ_x}, $1 \le x \le n$, and let $\alpha = \gamma_{\sigma,\rho}$. If $\rho_\pi^* \ge \rho$, then: $h(\alpha, \pi) = \pi^{-1}(\sigma)$, where $\pi^{-1}(\bullet)$ denotes the pseudo-inverse of $\pi(\bullet)$, defined in [3]. Moreover:

$$h(\alpha, \pi) = \pi^{-1}(\sigma) = D + \left[\bigvee_{1 \le x \le n} \frac{\sigma - \sigma_x}{\rho_x} \right]^+$$

Property 2.5 (deconvolution):

Let π be a pseudoaffine curve, with offset D and n leaky-bucket stages γ_{σ_x,ρ_x}, $1 \le x \le n$, and let $\alpha = \gamma_{\sigma,\rho}$. If $\rho_\pi^* \ge \rho$, then:

$$\alpha \odot \pi = \alpha \odot \delta_D = \gamma_{\sigma + \rho \cdot D, \rho}$$

The following property (see [13]) has an important practical implication, as it allows one to constrain the set of service curves obtained by applying Theorem 2.1 to a pseudoaffine service curve:

Property 2.6 (isotonicity):

Let $\pi_1, \pi_2 \in \Pi$ with $\pi_1 \le \pi_2$, and let $\alpha_1 \ge \alpha_2$. Then $E(\pi_1, \alpha_1, \tau) \le E(\pi_2, \alpha_2, \tau)$. Moreover, if $0 \le \tau_1 \le \tau_2 \le h(\alpha, \pi_1)$, $E(\pi_1, \alpha, \tau_1) \le E(\pi_1, \alpha, \tau_2)$.

Define $\mathrm{M} = \{E(\pi, \alpha, \tau), \tau \ge 0\}$ and $\mathrm{M}' = \{E(\pi, \alpha, \tau), \tau \ge h(\alpha, \pi)\}$. Property 2.6 states that $\forall \mu \in \mathrm{M}/\mathrm{M}'$, $\exists \varpi \in \mathrm{M}' : \varpi \ge \mu$, which is to say that service curves belonging to M/M' are not relevant for computing performance bounds (e.g. delay and output arrival curves). Accordingly, when Theorem 2.1 is applied to a pseudoaffine service curve so as to compute $E(\pi, \alpha, \tau)$, the resulting set of curves can be reduced by imposing the constraint $\tau \ge h(\alpha, \pi)$. By doing this we obtain a set of curves that can easily be expressed in a parametric way, as the following corollary shows ([13]).

Corollary 2.7:

Let π be a pseudoaffine service curve, with offset D and n leaky-bucket stages γ_{σ_x,ρ_x}, $1 \le x \le n$, and let $\alpha = \gamma_{\sigma,\rho}$, with $\rho_\pi^* \ge \rho$. Then, $\{E(\pi, \alpha, \tau), \tau \ge h(\alpha, \pi)\} \equiv \{\overline{E}(\pi, \alpha, s), s \ge 0\}$, with:

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

or, equivalently,

$$\overline{E}(\pi, \alpha, s) = \delta_{h(\alpha,\pi)+s} \otimes \left[\bigotimes_{1 \le x \le n} \gamma_{\rho_x \{ s + h(\alpha,\pi) - D \} - (\sigma - \sigma_x), \rho_x - \rho} \right] \quad (5)$$

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. Furthermore, we are interested in computing a tight output arrival curve for that flow at the end of the tandem.

At each node, *FIFO multiplexing* is in place, meaning that all flows traversing the node are buffered in a single queue First-Come-First-

[1] Actually, Property 2.5 was not shown in [13], being actually a new – though minor – result. We mention it here for the sake of readability. Its proof is straightforward, thus it is omitted.

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, $1 \le k \le N$. In the above framework, a flow can be identified by the couple (i, j), $1 \le i \le j \le N$, 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 $(i, j) \neq (1, N)$ and we identify it using the path (i, j) as a subscript. As far as the *tagged* flow, which traverses path $(1, N)$, is concerned, we will see later on in this paper that we might or might not need to distinguish it from the other flows traversing the same path, depending on which bound we want to compute. More specifically, we will show 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. On the other hand, in order to compute *output arrival curves*, the tagged flow has to be considered *separately* from the rest of the flows traversing path $(1, N)$. The latter, however, can be considered as one flow, which we denote as rf. Henceforth, we denote with $(1, N)$ the aggregate of the flows, therein including the tagged flow, which traverse all the tandem, and we use subscripts tf and rf whenever we need to detail further.

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 $i < h \le j < k$. Said in other words, let us consider two flows (i, j), (h, k), with $(i, j) \neq (h, k)$ and $i \le h$. Then either $j < h$, or $k \le j$. 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). For example, Figure 2 represents a nested tandem of three nodes. Flows $(2,2)$ and $(3,3)$ are nested within flow $(2,3)$. Furthermore, flows $(1,1)$, $(2,2)$, $(3,3)$ and $(2,3)$ are nested within $(1,3)$, that is the tagged flow. Given a flow (i, j), we denote its *level of nesting* $l(i, j)$ as the number of flows (h, k) into which it is nested. For instance, with reference to Figure 2, it is $l(1,1) = l(2,3) = 2$, and $l(2,2) = l(3,3) = 3$. The level of nesting of the tagged flow is therefore equal to one. The *level of nesting of the tandem* is the maximum level of nesting of one of its flows, which can be easily recognized to be the maximum number of flows crossing a single node. Note that a tandem of N nodes has a level of nesting no greater than N, and that the maximum number of flows insisting on an N-node nested tandem is $2N - 1$.

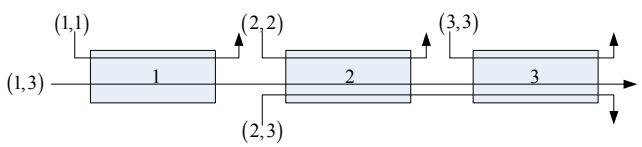

Figure 2 – an example of nested tandem

A particular case of n-level nested tandem is the one in which $\forall x, 1 \le x \le n, \exists! (i, j) : l(i, j) = x$, i.e. we have only *one* flow at each level of nesting. We call such a tandem a *fully nested tandem*. For instance, a sink-tree tandem, i.e. a tandem in which there are exactly N flows, whose path is (i, N), $1 \le i \le N$ (see Figure 3, above), is a fully nested tandem (whose level of nesting is N). On the other hand, a tandem is non-nested if it does not verify the above definition, as the one shown in Figure 3, below. In that case, we say that flow $(1,2)$ *intersects* flow $(2,3)$.

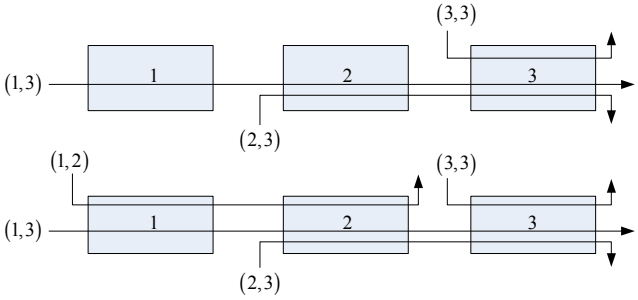

Figure 3 – a fully nested tandem (above) and a non nested tandem (below)

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 $1 \le h \le N$,

$$\sum_{(i,j):\, i \le h \le j} \rho_{(i,j)} \le R^h \qquad (6)$$

Note that this allows a node's rate to be utilized up to 100%, and it is therefore a necessary condition for stability. Moreover, we assume that the buffer of a node is large enough as to guarantee that traffic is never dropped.

We have devised a methodology, called the *Least Upper Delay Bound* (LUDB) methodology, that allows one to compute end-to-end delay bounds and output arrival curves for a flow traversing a *nested* tandem. However, the latter can also be extended for analyzing *non-nested* tandems. For this reason, we first focus on nested tandems in Section 4, and extend our analysis to non-nested tandems later on in Section 5.

4. LEAST UPPER DELAY BOUND METHODOLOGY

In this paragraph, we present the LUDB methodology. At a first level of approximation, the latter consists in computing *all* the service curves for the tagged flow: we start from the aggregate service curves at each node, and we apply Corollary 2.7 iteratively in order to remove one flow $(i, j) \not\equiv (1, N)$ from the tandem. With every iteration, a new free parameter $s_{(i,j)}$ is introduced. Therefore, we compute in fact a *multi-dimensional 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*.

Let us consider a nested tandem of N nodes, whose level of nesting is $n \ge 2$ (otherwise the problem is trivial). The algorithm for computing the delay bound for the tagged flow can be described as follows.

As a first step, we build the *nesting tree* of the tandem, which is in fact a simplified representation of the tandem. Let us define two sets:

$$S_{(h,k)} = \left\{ (i, j) : h \le i \le j \le k \text{ and } l(i, j) = l(h, k) + 1 \right\},$$

i.e. the set of flows which are nested right into (h, k), and:

$$C_{(h,k)} = \left\{ l : h \le l \le k \text{ and } \forall (i, j) \in S_{(h,k)}, l < i \text{ or } l > j \right\},$$

i.e. the set of nodes in path (h,k) that are not in the path of any flow in $S_{(h,k)}$. Note that, if $S_{(h,k)}=\varnothing$, then $C_{(h,k)}=\{h,h+1,...,k\}$.
For the sake of clarity, hereafter the nodes in the nesting tree are called *t-nodes*, in order to distinguish them to the nodes in the path of the tagged flow. In the nesting tree, there are two kind of t-nodes: non-leaf t-nodes represent a *flow*, and leaf t-nodes represent a *set of nodes* in the path. More specifically:

1. Each non-leaf t-node contains a flow (h,k). The root t-node contains $(1,N)$.
2. Each t-node whose content is (h,k) has all flows $(i,j)\in S_{(h,k)}$ as direct descendants. Furthermore, if $C_{(h,k)}\neq\varnothing$, (h,k) has *one* more direct descendant representing $C_{(h,k)}$ (which is a leaf t-node).

The level of nesting of a flow is the level of the corresponding t-node in the nesting tree. Accordingly, we henceforth write that $(i,j)\to(h,k)$ iff. $(i,j)\in S_{(h,k)}$, $S_{(h,k)}$ being the set of non-leaf *direct descendants* of (h,k), and that $(i,j)\to^{*}(h,k)$ to denote that (i,j) is a (possibly non-direct) descendant of (h,k). Figure 4 shows a tandem and the related nesting tree. Leaf t-nodes are shown as circles, while non-leaf nodes are ellipses. For instance, it is $(5,6)\to(4,6)$ and $(6,6)\to^{*}(1,6)$, whereas $(2,3)\not\to(4,6)$.
Once the nesting tree has been constructed, the set of end-to-end service curves for $(1,N)$ is computed by visiting the nesting tree from the leaves to the root as follows:

1. For each *leaf* t-node representing $C_{(h,k)}$ for some parent t-node (h,k), compute
$$\pi^{C_{(h,k)}}=\bigotimes_{j\in C_{(h,k)}}\beta^{j}$$
2. at a non-leaf t-node (h,k), compute a service curve as
$$\pi^{\{h,k\}}=\pi^{C_{(h,k)}}\otimes\left[\bigotimes_{(i,j)\in S_{(h,k)}}\overline{E}\left(\pi^{\{i,j\}},\alpha_{(i,j)},s_{(i,j)}\right)\right] \qquad (7)$$
i.e. as the convolution of:
   i) The (pseudoaffine) service curves obtained by applying Corollary 2.7 to the service curve computed at all child t-nodes;
   ii) The (rate-latency) service curve $\pi^{C_{(h,k)}}$ if $C_{(h,k)}\neq\varnothing$ (otherwise assume for completeness that $\pi^{C_{(h,k)}}=\delta_0=\beta_{0,+\infty}$).

The set of end-to-end service curves for $(1,N)$, call it $\pi^{\{1,N\}}$, is obtained by computing the service curve at the root t-node. The latter is a function of a number of free parameters $s_{(i,j)}:(i,j)\to^{*}(1,N)$. However, from Property 2.2 and Property 2.3 we obtain that, for each instance of these parameters, the resulting end-to-end service curve is pseudoaffine. Now, if the tagged flow traverses the whole tandem as part of an aggregate with other flows, then $\pi^{\{1,N\}}$ is a service curve for the *aggregate*. Call tf and rf the tagged flow and the aggregate of the rest of the flows, so that $\alpha_{tf}+\alpha_{rf}=\alpha_{(1,N)}$. The following theorem, whose proof is reported in [26], shows that we do not need to distinguish between them for the purpose of computing the delay bound.

Theorem 4.1

Let α_A and α_B be the leaky-bucket arrival curves for two flows A and B which are FIFO-multiplexed in a node. Assume that the node has a pseudoaffine service curve π with offset D and n leaky-bucket stages γ_{σ_x,ρ_x}, $1\le x\le n$, with $\rho_\pi^{*}\ge\rho_A+\rho_B$. Then:

$$h(\alpha_A+\alpha_B,\pi)=\min_{s\ge 0}\left\{h\left(\alpha_A,\overline{E}(\pi,\alpha_B,s)\right)\right\}$$

■

Based on the above result, the least upper end-to-end delay bound for the tagged flow is the following:

$$D=\min_{\substack{s_{(i,j)}\ge 0,\\ (i,j)\to^{*}(1,N)}}\left\{h\left(\alpha_{(1,N)},\pi^{\{1,N\}}\left(s_{(i,j)}:(i,j)\to^{*}(1,N)\right)\right)\right\} \qquad (8)$$

Now, since $\pi(\ )$ is pseudoaffine and $\alpha_{(1,N)}$ is an affine curve, problem (8) 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 (or, equivalently, the number of non-leaf t-nodes in the nesting tree) minus one, $x<2N$. Standard techniques exist to solve such *piecewise-linear programming* (P-LP) problems (see, for instance, [25]). Furthermore, closed-form solutions have been derived through ad-hoc methods for special cases of problem (8), i.e. for a 2-level nested tandem of arbitrary length [12], and for a sink-tree tandem [13].

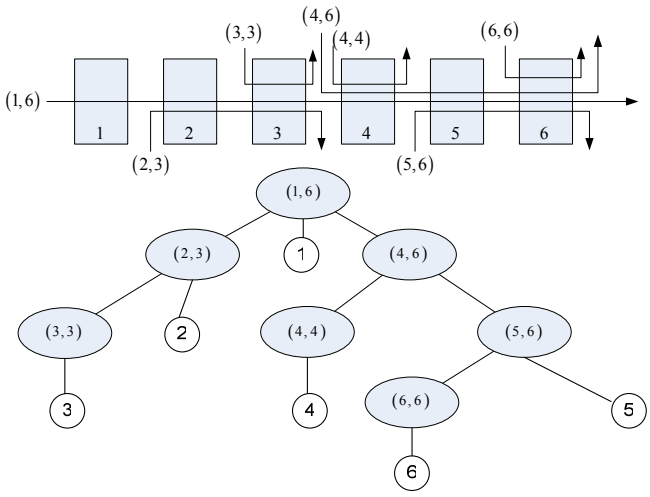

Figure 4 – a nested tandem and the related nesting tree

4.1 Computing the output arrival curve for the tagged flow

We now show how to compute the output arrival curve for a tagged flow at the exit of a tandem. We will see later on in Section 5 that this result is required when the LUDB methodology is to be applied to non-nested tandems.
Call $\pi^{tf}=\left\{\overline{E}\left(\pi^{\{1,N\}},\alpha_{rf},s_{rf}\right),s_{rf}\ge 0\right\}$ the set of end-to-end service curves for the *tagged flow* (if $tf\equiv(1,N)$, we assume that α_{rf} is null). In general, π^{tf} is computed through Corollary 2.7 as if rf was *nested within* the tagged flow, and it depends on an additional free parameter s_{rf}. A constraint on the arrival curve of the tagged flow after node N, call it α_{tf}^{N+1}, is thus the following (see [3], Chapter 6):

$$\alpha_{tf}^{N+1}(t)=\min_{\substack{s_{(i,j)}\ge 0,(i,j)\to^{*}(1,N)\\ s_{rf}\ge 0}}\left\{\alpha_{tf}\odot\overline{E}\left(\pi^{\{1,N\}},\alpha_{rf},s_{rf}\right)\right\}(t)$$

However, from Property 2.5, we have

$$\alpha_{tf}^{N+1}(t)=\min_{\substack{s_{(i,j)}\ge 0,(i,j)\to^{*}(1,N)\\ s_{rf}\ge 0}}\left\{\alpha_{tf}\odot\delta_{D^{tf}}\right\}(t) \qquad (9)$$

Where $D^{tf}=h\left(\alpha_{rf},\pi^{\{1,N\}}\right)+s_{rf}$, as shown in (5). Let σ_{tf} and ρ_{tf} be the leaky bucket parameters of α_{tf} at node 1. Since all the curves between curly brackets are *affine* and with the *same* slope, then there exists one such curve which is smaller than the others for every t. Therefore, $\alpha_{tf}^{N+1}(t)$ is a leaky bucket with $\rho_{tf}^{N+1}=\rho_{tf}$ and $\sigma_{tf}^{N+1}=\sigma_{tf}+\rho_{tf}\cdot D^{\min}$, where:

$$D^{\min} = \min_{\substack{s_{(i,j)} \geq 0, (i,j) \to^* (1,N) \\ s_{rf} \geq 0}} \left\lceil h\left(\alpha_{rf}, \pi^{\{1,N\}}\right) + s_{rf} \right\rceil \quad (10)$$

Now, since $h\left(\alpha_{rf}, \pi^{\{1,N\}}\right) \not\leq s_{rf}$, this is equivalent to

$$D^{\min} = \min_{s_{(i,j)} \geq 0, (i,j) \to^* (1,N)} \left\lceil h\left(\alpha_{rf}, \pi^{\{1,N\}}\right) \right\rceil \quad (11)$$

Thus, computing the output arrival curve for the tagged flow is equivalent to computing the LUDB. More specifically, if $\alpha_{rf} \neq 0$ (i.e., $tf \not\equiv (1,N)$), the two problems have the same number of variables $s_{(i,j)} : (i,j) \to^* (1,N)$. Otherwise, if $tf \equiv (1,N)$ and $\alpha_{rf} = 0$, it can be easily shown that (see [26] for the computations)

$$D^{\min} = \sum_{x \in S_{(1,N)}} \left\{ \min_{s_{(i,j)} \geq 0, (i,j) \to^* x} \left\lceil h\left(\alpha_x, \pi^x\right) \right\rceil \right\} + \theta^{C_{(1,N)}} \quad (12)$$

i.e. $D^{\min}$ is the sum of the LUDB delay bounds of the flows in $S_{(1,N)}$, plus possibly the latency of the leaf t-node $C_{(1,N)}$. Thus, solving (12) implies solving several *separate* P-LP problems, one for each non-leaf direct descendant of tf. This is generally easier than solving (8). For instance, for the nested tandem of Figure 4 (assuming $tf \equiv (1,N)$), computing (10) requires computing the LUDB of flows $(2,3)$ and $(4,6)$ in their respective sub-trees. The latter depend on one and three variables respectively. On the other hand, computing (8) for the tagged flow entails solving a P-LP problem with six variables.

Hereafter, we show how to compute the end-to-end delay bounds in non-nested tandems.

5. ANALYZING NON NESTED TANDEMS

The LUDB methodology cannot be applied directly to non-nested tandems. In fact, the nesting tree for the tagged flow is built under the assumption that the service curves of any two sibling t-nodes can be computed independently, whereas in a non-nested tandem this is not true anymore. While this makes it impossible to compute a set of end-to-end service curves for the tagged flow, we can still find a way to compute *partial* service curves for it in *disjoint sub-tandems*. From those, partial, per sub-tandem delay bounds can be computed. Then, an end-to-end delay bound can be computed by summing up the partial delay bounds.

More formally, given a tandem $T = \{i : 1 \leq i \leq N\}$ of N nodes, we partition it into m disjoint sub-tandems $T_i = \{k : c_{i-1} \leq k \leq c_i - 1\}$, $1 \leq i \leq m$, with $c_m = N+1$, $c_i < c_{i+1}$ (and $c_0 = 1$ for ease of notation). Visually speaking, this can be done by *cutting* the tandem right before each node c_i, which is accordingly called a *cut*. Furthermore, we call a *set of cuts* a set of nodes $SC = \{c_i : 1 \leq i \leq m\}$ such that $\bigcup_{1 \leq i \leq m} T_i \equiv T$ and $T_i \cap T_j = \varnothing$ for $i \neq j$. Now, if T is a *non-nested* tandem, we can always find a set of cuts that partition it into *nested* sub-tandems, where the (partial) delay bounds for the tagged flow can actually be computed through the LUDB methodology. Such a partitioning requires at most $\lceil N/2 \rceil$ cuts. In fact, any two-node tandem is by definition a nested tandem.

We therefore need to describe i) how to compute a set of cuts given a non-nested tandem, and ii) how to compute the per sub-tandem delay and output bounds given a set of cuts. Before delving deeper into the above two issues, we report a clarifying example.

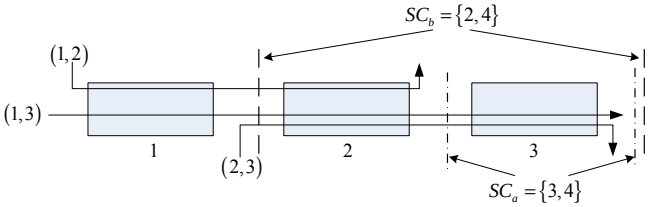

Figure 5 – A three-node non-nested tandem and two different sets of cuts

Example 5.1

Consider the simplest possible non-nested tandem, shown in Figure 5. Call $\sigma_{(i,j)}$ and $\rho_{(i,j)}$ the leaky bucket parameters of the arrival curve $\alpha^i_{(i,j)}$ of flow (i,j), where $(i,j) \in \{(1,3),(2,3),(1,2)\}$, and let θ^i and R^i be the latency and rate of node i, $1 \leq i \leq 3$. Assume also that (6) holds at each node.

We can compute a delay bound for flow $(1,3)$ by cutting the tandem according to the set of cuts $SC_a = \{3,4\}$. The algorithm is as follows:

a) Compute $d^{\{1,2\}}$, i.e. the LUDB through sub-tandem $T_1 = \{1,2\}$.
b) Compute the arrival curves for flow $(1,3)$ and $(2,3)$ at the cut, i.e. at node 3, $\alpha^3_{(1,3)}$, $\alpha^3_{(2,3)}$.
c) Compute the delay bound through sub-tandem $T_2 = \{3\}$, $d^{\{3\}}$, as $h\left(\alpha^3_{(1,3)} + \alpha^3_{(2,3)}, \beta^3\right)$. Note that, by Theorem 4.1, flows $(1,3)$ and $(2,3)$ can be aggregated when computing $d^{\{3\}}$.
d) Compute the end-to-end delay bound as $V^a = d^{\{1,2\}} + d^{\{3\}}$.

The result is (detailed computations are reported in [26]):

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

$$\begin{aligned} V^a_1 = {} & \theta^1 \cdot \left[1 + \frac{\rho_{(1,3)}}{R^3} \cdot \left(1 + \frac{\rho_{(2,3)}}{R^2}\right)\right] + \theta^2 \cdot \left(1 + \frac{\rho_{(2,3)} + \rho_{(1,3)}}{R^3}\right) + \theta^3 \\ & + \frac{\sigma_{(1,2)}}{R^3}\left(\frac{\rho_{(2,3)}}{R^2} + \frac{R^3 + \rho_{(1,3)}}{R^1}\right) \\ & + \frac{\sigma_{(1,3)}}{R^3} \cdot \left(1 + \frac{\rho_{(2,3)}}{R^2}\right) + \frac{\sigma_{(1,3)}}{R^1} + \frac{\sigma_{(2,3)}}{R^2} \cdot \left(1 + \frac{R^2 + \rho_{(1,3)}}{R^3}\right) \end{aligned} \quad (13)$$

Otherwise,

$$\begin{aligned} V^a_2 = {} & \theta^1 \cdot \left[1 + \frac{\rho_{(1,3)}}{R^3} \cdot \left(1 + \frac{\rho_{(2,3)}}{R^2}\right)\right] + \theta^2 \cdot \left(1 + \frac{\rho_{(2,3)} + \rho_{(1,3)}}{R^3}\right) + \theta^3 \\ & + \frac{\sigma_{(1,2)}}{R^2} \cdot \left[\frac{\rho_{(2,3)}}{R^3} + \left(1 + \frac{\rho_{(1,3)}}{R^3}\right) \cdot \left(1 + \frac{\rho_{(2,3)}}{R^1}\right)\right] \\ & + \frac{\sigma_{(1,3)}}{R^2} \cdot \left(1 + \frac{\rho_{(2,3)}}{R^2} + \frac{R^2 + \rho_{(2,3)}}{R^3}\right) + \frac{\sigma_{(2,3)}}{R^2} \cdot \left(1 + \frac{R^2 + \rho_{(1,3)}}{R^3}\right) \end{aligned} \quad (14)$$

Similarly, we can cut the tandem according to the set $SC_b = \{2,4\}$. In this case, the delay bound is computed as follows:

a) Compute the delay bound at sub-tandem $T_1 = \{1\}$ as $d^{\{1\}} = h\left(\alpha^1_{(1,3)} + \alpha^1_{(1,2)}, \beta^1\right)$.
b) Compute the arrival curves for flows $(1,3)$ and $(1,2)$ at node 2, $\alpha^2_{(1,3)}$, $\alpha^2_{(1,2)}$.
c) Compute the LUDB $d^{\{2,3\}}$ through sub-tandem $T_2 = \{2,3\}$. Note that, by Theorem 4.1, flows $(1,3)$ and $(2,3)$ can be aggregated when computing $d^{\{2,3\}}$.
d) Compute the end-to-end delay bound as $V^b = d^{\{1\}} + d^{\{2,3\}}$.

The result is (detailed computations are reported in [26]):

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

$$V_1^b = \theta^1 \cdot \left(1 + \frac{\rho_{(1,2)}}{R^2} + \frac{\rho_{(1,3)}}{R^3}\right) + \theta^2 + \theta^3 + \frac{\sigma_{(1,2)}}{R^1}\left(1 + \frac{\rho_{(1,3)}}{R^3}\right) + \frac{\sigma_{(1,2)}}{R^2} + \frac{\sigma_{(1,3)}}{R^1}\left(1 + \frac{\rho_{(1,2)}}{R^2}\right) + \frac{\sigma_{(1,3)}}{R^3} + \frac{\sigma_{(2,3)}}{R^3} \quad (15)$$

Otherwise,

$$V_2^b = \theta^1 \cdot \left[1 + \frac{\rho_{(1,2)}}{R^2} + \frac{\rho_{(1,3)}}{R^2} \cdot \left(1 + \frac{\rho_{(2,3)}}{R^3}\right)\right] + \theta^2 + \theta^3 + \frac{\sigma_{(1,2)}}{R^1} \cdot \left[1 + \frac{\rho_{(1,3)}}{R^2} \cdot \left(1 + \frac{\rho_{(2,3)}}{R^3}\right)\right] + \frac{\sigma_{(1,2)}}{R^2} + \frac{\sigma_{(1,3)}}{R^1} \cdot \left(1 + \frac{\rho_{(1,2)}}{R^2}\right) + \frac{\sigma_{(1,3)}}{R^2} \cdot \left(1 + \frac{\rho_{(2,3)}}{R^3}\right) + \frac{\sigma_{(2,3)}}{R^2} \cdot \left(1 + \frac{\rho_{(2,3)}}{R^3}\right) \quad (16)$$

For any selection of the node and flow parameters, $V = V^a \wedge V^b$ is an end-to-end delay bound for the tagged flow.

■

Having in mind the above example, we first describe a general algorithm for computing the end-to-end delay bound once a set of cuts is identified, and then discuss how to compute a set of cuts.

5.1 Computing the end-to-end delay bound

Given a non-nested tandem T of N nodes and set of m cuts $SC = \{c_i, 1 \le i \le m\}$, we describe the algorithm for computing the end-to-end delay bound. Call F the set of all flows in T. Similarly, call T_i the sub-tandem $\{c_{i-1}, c_i - 1\}$, and $F_i = \{(j,k) \in F : k \ge c_{i-1} \text{ or } j < c_i\}$ the subset of flows traversing T_i. Let $FC_i = \{(j,k) \in F : j \le c_i \le k\}$ denote the set of flows crossing cut c_i, and let $(j,k)|_{T_i} = (j \vee c_{i-i}, k \wedge c_i - 1)$ be the *sub-tandem reduction* of flow $(j,k) \in F_i$. The algorithm can be described as follows: consider each sub-tandem T_i iteratively in increasing order. Assume that, for each flow $(j,k) \in FC_{i-1}$, its arrival curve at cut c_{i-1} is available. Then, compute the LUDB d^{T_i} for the tagged flow *and* the arrival curve at cut c_i for all flows $(j,k) \in FC_i$. Finally, compute the end-to-end delay bound as $\sum_{1 \le i \le m} d^{T_i}$. However, in doing this, a key optimization is required, i.e. flows have to be kept aggregated as much as possible. Note that, even if we assume that at most one flow exists for a given pair of nodes (j,k) in the tandem, we cannot avoid that for two distinct flows (a,b), (a',b'), it is $(a,b)|_{T_i} = (a',b')|_{T_i}$. When this happens, aggregating the two flows within T_i leads to tighter bounds. This can be justified with a simple example: consider two flows whose leaky-bucket arrival curves are α_1 and α_2, which are FIFO multiplexed into the same rate-latency service curve β. Then, both $\alpha_{1,2}^{"} = \alpha_1 \odot \overline{E}(\beta, \alpha_2, s_2) + \alpha_2 \odot \overline{E}(\beta, \alpha_1, s_1)$ and $\alpha_{1,2}^{'} = (\alpha_1 + \alpha_2) \odot \beta$ are output constraints for the aggregate of the two flows. However, one can easily recognize that $\alpha_{1,2}^{'} < \alpha_{1,2}^{"}$, i.e. it is a tighter constraint.

Flow aggregation can be exploited in the following four ways:

a. in order to compute the LUDB in T_i, any two flows (a,b), (a',b'), such that $(a,b)|_{T_i} = (a',b')|_{T_i}$, can be aggregated at their entry node in T_i. This has been done, for instance, in Example 5.1, for $SC_b = \{2,4\}$ and $T_2 = \{2,3\}$.
b. In order to compute the arrival curve at cut c_i for a flow $(j,k) \in F_i \cap FC_i$, any two flows (a,b), (a',b'), such that $(a,b)|_{T_i} = (a',b')|_{T_i}$, can be aggregated at their entry node in T_i. However, flow (j,k) itself must not be aggregated with others;
c. any two flows in F_i (a,b), (a',b), leaving at the *same* node b, can be *permanently aggregated* at cut c_i. This means that, starting from the sub-tandem T_{i+1}, the two flows can in fact be regarded as a single flow for all computations;
d. consider two flows (a,b) and (a',b'), (with $b \ne b'$), such that $\{(a,b),(a',b')\} \in F_i \cap FC_k$ for some i and $k > i$. At least when traversing T_h, $i+1 \le h \le k$, the two flows can be temporarily aggregated and considered as a single flow. Thus, their joint output arrival curve at cuts c_{h-1} can be computed and used in sub-tandem T_h. Note that, since the two flows will actually leave at different nodes (since $b \ne b'$), we cannot avoid to compute also their single output arrival curves, which we will be forced to use later on (possibly in sub-tandem T_{i+k+1}). However, at least we can avoid using them when not strictly necessary.

5.2 Computing a set of cuts

Hereafter, we present a greedy algorithm for computing a set of cuts in a tandem. The latter selects the longest downstream nested sub-tandem at each iteration. First of all, we consider the subset of flows which might potentially intersect with each other, i.e. $F' = F \setminus \{(i,i), 1 \le i \le N\} \setminus \{(1,N)\}$, since flows spanning just one node cannot intersect with other flows, nor can the tagged flow, which encompasses all others. The pseudocode for the algorithm is shown in Figure 6. The `NextCut` variable stores the value of the next cut, which is initially $N+1$ (see lines 1 and 13), and can be decreased when two intersecting flows are detected (line 6). Going downstream along the tandem (lines 2 and 8), at each node i the set G includes all the flows (or sub-tandem reductions of) which might potentially intersect with others. Thus, whenever a new node i is considered, flows which have left the tandem at $i-1$ are removed from G (line 3), and flows which enter at i are examined (lines 4-5) before they are added to G (line 7): if any of these flows intersect with another flow (h,k), in G, $h < i$, then we must place the next cut c_i at most at node $k+1$ (line 6). Once a cut c_i is reached (line 9), the computations are reset by substituting all the flows (h,k) which are intersected by that cut with their remainder (c_i,k) (lines 11-13). Furthermore, the `NextCut` variable and the G set are reset (line 14).

```
1.  NextCut=N+1;  SC = {N+1};  G = ∅;  i=1;
2.  While i<N do
3.      G ← G \ {(l,i-1) : (l,i-1) ∈ F'}
4.      For each flow (i,j) ∈ F',
5.          If h < i ≤ k < j  for some (h,k) ∈ G  then
6.              NextCut=min(NextCut,k+1)
7.          G ← G ∪ {(i,j)}
8.      Increase i
9.      If i==NextCut then
10.         SC ← SC ∪ {i}
11.         For each flow (h,k) ∈ G,
12.             If i ≤ k  then  F' ← F' ∪ {(i,k)}
13.         F' ← F' \ G
14.         NextCut=N+1;  G = ∅;
```

Figure 6 – computation of a set of cuts

As an example, consider the tandem shown in Figure 7. It is $F' = \{(1,2),(2,5),(3,4),(4,5)\} \subset F$. At node 1, $G = \{(1,2)\}$. Then, at node 2, flow $(2,5)$ is found to intersect with flow $(1,2)$, and therefore a cut has to be placed at node 3. Starting from node 3, we have $F' = \{(3,5),(3,4),(4,5)\}$, $(3,5)$ being the sub-tandem reduc-

tion of flow $(2,5)$ after cut 3. We then have $G=\{(3,5),(3,4)\}$. At node 4, flow $(4,5)$ is found to intersect with flow $(3,4)$, and therefore a cut has to be placed at node 5.

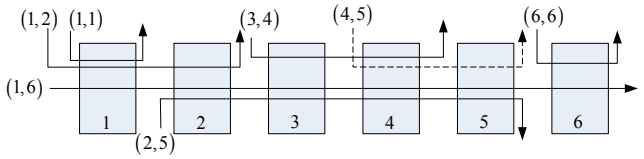

Figure 7 – a sample non-nested tandem

The algorithm yields $SC_1=\{3,5,7\}$, which is in fact obtained by considering the longest sub-path in a greedy fashion. However, we observe that there exist two sets, namely $SC_2=\{2,5,7\}$ and $SC_2=\{2,4,7\}$, both of which generate one three-node sub-tandem. As the example shows, several different sets of cuts can be superimposed to a non-nested tandem, and for each set of cuts a different delay bound can be computed. In general, given M sets of cuts $SC_1,...,SC_M$, we can compute M end-to-end delay bounds V^i, $1\le i\le M$, and then compute:

$$V=\bigwedge_{1\le i\le M} V^i$$

i.e., the tightest delay bound among those that can be found with this method. For instance, in Example 5.1, 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 parameters.

Since the computation of the LUDB – especially for large tandems – may require complex computations, one might wonder whether an *optimal* set of cuts, i.e., one leading to the smallest delay bound, can be identified without computing all the bounds, or at least whether some sets of cuts can be excluded *a priori* since they will certainly lead to larger bounds. Research on this topic is still ongoing at the time of writing. However, we can give some guidelines. Broadly speaking, the smaller the number of cuts is, the smaller the end-to-end delay bound is likely to be. In fact, cutting the tandem entails assuming partial worst-case scenarios which are not simultaneously possible. Consider for instance the computation related to the set SC_a in Example 5.1: the worst-case output arrival curves $\alpha^3_{(1,3)}$, $\alpha^3_{(2,3)}$ are obtained separately, under *different* scenarios, which are not simultaneously possible. Furthermore, both scenarios are not simultaneously possible with the one that leads to the partial delay bound $d^{\{1,2\}}$. Therefore, using $\alpha^3_{(1,3)}$, $\alpha^3_{(2,3)}$ and $d^{\{1,2\}}$ in the same computation overestimates the delay bound. Similar considerations hold for set SC_b. To reinforce the above statement, we observe that, in all the above expressions, there is at least one burst $\sigma_{(i,j)}$ which appears more than once. This further motivates us to think that neither bound is tight (although to the best of our knowledge no formal proof exists that the "pay burst only once" principle holds in FIFO multiplexing tandems).

This said, one might wonder whether the smallest delay bound is achieved with the set of (or, more precisely, with one of the sets of) cuts of minimum cardinality. Unfortunately, the answer is not so straightforward. The reason is that, for each cut, the set of output arrival curves for possibly all the n flows intersected by the cut need be computed (although in the previous sub-section we have described some optimization that may actually reduce that number). The larger n is, the more different output arrival curves need be computed independently. However the cumulative departure functions of *all* the flows at the cut are correlated, since they have been scheduled together through one or more upstream FIFO nodes. Therefore, assuming that they are independent worsens the scenario on which the subsequent partial delay bounds are computed. The larger n is, the more such effect worsens the delay bounds. Thus, it is not just the *number* of cuts that matters.

6. A CASE STUDY

In order to show the effectiveness of the proposed approach, we compare it to per-node analysis in a case study tandem (shown in Figure 8). An even number of nodes N are traversed by a tagged flow $(1,N)$, by all flows $(i,i+1)$, $1\le i<N$, and by two flows $(1,1)$ and (N,N) for symmetry. We assume that all flows have the same leaky-bucket arrival curve $\gamma_{\sigma,\rho}$, and that all nodes have the same rate-latency service curve $\beta_{R,\theta}$, with $R\ge 3\cdot\rho$. The greedy cut computation algorithm produces the set of cuts $SC=\{k: k=2i+1, 1\le i\le N/2\}$, hence the end-to-end delay is computed as the sum of the maximum number of partial delay bounds, which is the most unfavorable condition for our methodology.

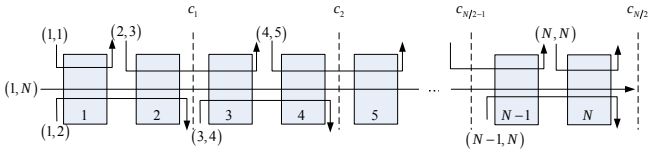

Figure 8 – the case-study tandem

Each sub-tandem $T_i=\{2i-1,2i\}$, $1\le i\le N/2$, is a two-node, one-level nested tandem, for which we need to compute:

a) the end-to-end delay bound d^{T_i};
b) the output arrival curve for the tagged flow at the cut, $\alpha^{2i+1}_{(1,N)}$;
c) the output arrival curve for flow $(2i,2i+1)$ at the cut, $\alpha^{2i+1}_{(2i,2i+1)}$.

Assuming that we have in input the same quantities computed for sub-tandem T_{i-1}.

Call σ^t_i, σ^f_i the burstiness of the arrival curve at node $2i-1$ of the tagged flow and of flow $(2i-2,2i-1)$, $1<i\le N/2$, and let $\sigma^t_1=\sigma^f_1=\sigma$ be the burstiness of the tagged flow and of flow $(1,1)$ at node 1[2]. By means of straightforward algebraic manipulations, the following three recursive formulas can be computed:

$$d^{T_i}=2\theta+\frac{\sigma+\sigma^f_i}{R}+\frac{\sigma+\sigma^t_i}{R-\rho}$$

$$\sigma^f_{i+1}=\sigma+\rho\cdot\left[\theta+\frac{\sigma+\sigma^t_i+2\rho\cdot\left(\theta+\sigma^f_i/R\right)}{R}\right]$$

$$\sigma^t_{i+1}=\sigma^t_i+\rho\cdot\left[2\theta+\frac{\sigma+\sigma^f_i}{R}+\frac{\sigma}{R-\rho}\right]$$

On the other hand, we can also compute the sum of per-node delay bounds recursively as follows. Call σ^t_k, σ^f_k the burstiness of the arrival curve at node k of the tagged flow and of flow $(k-1,k)$, $1<k\le N$, and let $\sigma^t_1=\sigma^f_1=\sigma$ be the burstiness of the tagged flow and of flow $(1,1)$ at node 1. The expressions are as follows:

$$d^k=\theta+\frac{\sigma+\sigma^t_k+\sigma^f_k}{R}$$

$$\sigma^f_{k+1}=\sigma+\rho\cdot\left[\theta+\frac{\sigma^f_k+\sigma^t_k}{R}\right],\ \sigma^t_{k+1}=\sigma^t_k+\rho\cdot\left[\theta+\frac{\sigma+\sigma^f_k}{R}\right]$$

[2] The rate for all arrival curves is obviously ρ.

In Figure 9 we plot the ratio of the per node delay bound to the one found through the LUDB methodology, as a function of the number of nodes, and for several utilization values (i.e., assuming $R = 3\rho/U$, with U ranging from 20% to 100%). In the graph, we have $\theta = 1$, $\sigma = 5$ and $\rho = 4$. However, the results are almost insensitive to the value of those three parameters.

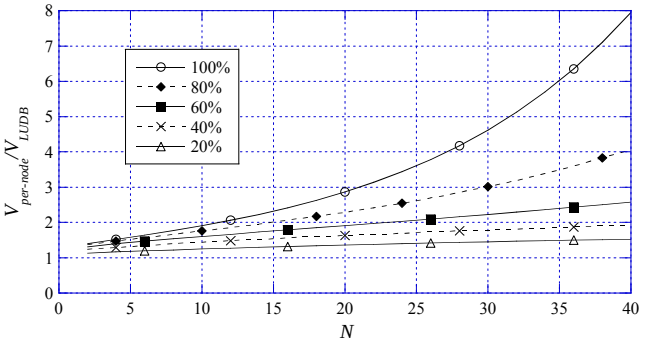

Figure 9 – Ratio of the end-to-end delay bounds computed with per-node analysis and through the LUDB methodology for several utilizations

As the figure clearly shows, the ratio is always above one, which means that the LUDB bound is always tighter. The ratio increases with the length of the tandem, and it gets larger as the utilization approaches 100%. In Figure 10, the ratio of the burst of the tagged flow's output arrival curve are plotted, showing the same behavior.

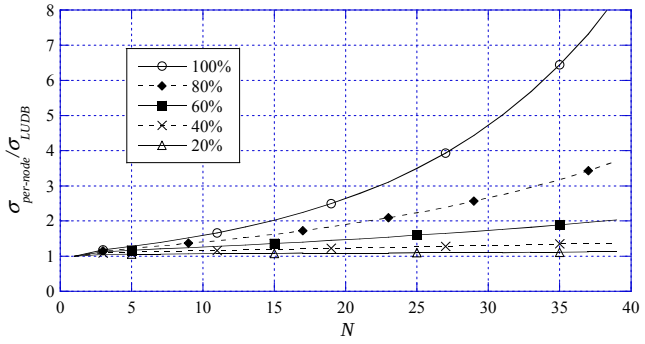

Figure 10 - Ratio of the burst of the tagged flow computed with per-node analysis and through the LUDB methodology for several utilizations

7. RELATED WORK

The problem of finding end-to-end delay bounds for single flows under aggregate scheduling has been the focus of quite a few research efforts in the recent past. Some works (e.g., [24]) envisage that traffic be aggregated in the network using *fair* aggregators. The latter are non work-conserving network elements which multiplex flows fairly into aggregates, so as to prevent bursts from accumulating. While employing fair aggregators is likely to yield better bounds than those obtained with FIFO multiplexing, the added complexity of non-work conserving fair aggregators, which in fact are elements which may work at a finer grain that that of a single aggregate, should not be understated. Architectures employing those elements, besides, have not been standardized, nor (to the best of our knowledge) practically deployed. A recent work [23] presents a tool for computing end-to-end delay bounds for flows subject to *blind* multiplexing. "Blind" means that no assumption is made regarding the multiplexing criterion: for instance, both a FIFO multiplexing scheme and a strict priority multiplexing scheme in which the tagged flow is always multiplexed at the lowest priority fit this definition. In that framework, concave piecewise arrival curves are assumed as a traffic constraint, and nodes are supposed to offer *strict* service curves to the aggregates traversing them. While the first hypothesis generalizes the one used in the paper, the second one (which is mandatorily required in order to be able to deal with blind multiplexing) imposes stricter constraints on the service, since a strict service curve constraint implies a service curve one, but not vice versa.

Despite being the easiest way to aggregate flows, few results exist in the literature related to FIFO multiplexing. A survey on the subject can be found in [19]. A closed form delay bound for a generic network configuration has been derived in [20], under the fluid model assumption, and extended in [21] to take packetization effects into consideration. In both cases, when a generic network configuration is considered, a bound can be derived only for small utilization factors: let H be a bound on the number of nodes traversed by any flow, and ν a bound on the utilization at any link, then the delay bound holds only if $\nu < \nu_{\max} = 1/(H-1)$. Furthermore, the bound is inversely proportional to $1-\nu(H-1)$, that is, the bound approaches infinity when the utilization level ν gets closer to $\nu_{\max}$. It is shown in [20] that, in order to derive any tighter bound for a network in which $\nu < \nu_{\max}$, as well as bounds which can be applied to a network in which $\nu \geq \nu_{\max}$, some more assumptions about the underlying network are required. On one hand, such further assumptions may consist in the type of information included in each packet. This is the approach in [22], where timestamps are associated to packets and appropriate scheduling algorithms are introduced. On the other hand, one can assume that the network topology and configuration within an administration domain are known, and attempt deriving delay bounds which rely on such information. For instance, feed-forward networks are known to be stable, which implies that delay bounds can actually be computed for any utilization value below 100% [3].

This last approach, which is also ours, has also been the focus of previous researches. A closed form expression for the end-to-end service curve for a tagged flow in a generic tandem of FIFO nodes, obtained under the same hypotheses as in this paper (i.e. leaky-bucket constrained flows and rate-latency service curves for the aggregates) was presented in [15]. However, the result, claimed to hold for an arbitrary topology tandem, has already been disproved in [13]. A method for deriving delay bounds in feed-forward networks with leaky-bucket constrained flows and rate-latency service curves for the aggregates has also been presented in [16],[17], although no end-to-end analysis of complex tandems is taken over therein. It consists in applying Theorem 2.1 iteratively in order to obtain an end-to-end service curve for a tagged flow, as we do in this paper. However, each time a family of equivalent service curves is computed at a node, only *one* service curve is considered (specifically, the one that we would obtain by applying Corollary 2.7 with $s = 0$). For the particular case of sink-tree tandems, it has been shown in [13] that the delay thus computed is always larger than the one computed according to the LUDB methodology (which, instead, capitalizes on keeping track of *all* the service curves), and that it can be arbitrarily loose, e.g. it can go to infinite when the rate provided for the aggregate is sufficient, and all the aggregate rates are finite. The same result can now be proved to hold for non-nested tandems. Consider for instance the three-node tandem of Example 5.1. The delay bound expression computed according to [16] is:

$$D = \sum_{i=1}^{3} \theta^i + \frac{\sigma_{(1,3)}}{\left(R^1 - \rho_{(1,2)}\right) \wedge \left(R^2 - \rho_{(1,2)} - \rho_{(2,3)}\right) \wedge \left(R^3 - \rho_{(2,3)}\right)} + \frac{\sigma_{(1,2)}}{\left(R^2 - \rho_{(2,3)}\right) \wedge R^1} + \frac{\sigma_{(2,3)}}{\left(R^2 - \rho_{(1,2)}\right) \wedge R^3} + \frac{\rho_{(2,3)}}{\left(R^2 - \rho_{(1,2)}\right) \wedge R^3} \left[\theta^2 + \frac{\sigma_{(1,3)} + \sigma_{(1,2)} + \theta^2 \cdot \left(\rho_{(1,3)} + \rho_{(1,2)}\right)}{R^2} \right] \quad (17)$$

However, the rate that divides the burst of the tagged flow $\sigma_{(1,3)}$ approach $\rho_{(1,3)}$ as the rate provisioning gets tighter. This implies that, no matter how large the node rates are, the delay bound of the tagged flow goes to infinite if $\rho_{(1,3)} \to 0$. On the other hand, (13)-(16) can be easily proved to yield finite delay bounds in these conditions, which means that the bound (17) can be arbitrarily loose.

8. CONCLUSIONS AND FUTURE WORK

In this paper we have presented the *Least Upper Delay Bound* (LUDB) methodology, which allows one to derive good end-to-end delay bounds in FIFO multiplexing networks of rate-latency nodes. It consists in computing the *minimum* among all the delay bounds that can be found by iteratively applying a known Network Calculus Theorem. Although the LUDB method can only be applied "as is" to nested tandems, we have devised algorithms to partition a tandem into nested sub-tandems. Thus, the end-to-end delay bound can be computed as the sum of the partial delay bounds computed in all sub-tandems. The above methodology has been shown to be more effective than per-node analysis even in a very unfavorable scenario.

This work can be extended in several directions: first of all, devising an ad-hoc algorithm for computing the LUDB given a nesting tree, with improved efficiency over standard P-LP solving methods. Then, devising algorithms for the *optimal* partitioning of non-nested tandems, so as to reduce the computations required for finding the best bound. Furthermore, implementing all the above algorithms into a standalone tool for analysis of FIFO networks (similar, in that respect, to [23]). Last, since there is reasonable evidence that the LUDB method may not yield the actual worst-case delay, especially if applied to non-nested tandems, assessing how reliable the delay bounds are. All the above mentioned extensions are being actively pursued at the time of writing.

9. REFERENCES

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

[2] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss, "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] H. Saito, Y. Miyao, M. Yoshisda, "Traffic Engineering using Multiple Multipoint-to-point LSPs.", Proc. of IEEE Infocom 2000, Tel Aviv (Israel), 26-30 March 2000, pp. 894-902.

[6] S. Bhatnagar, S. Ganguly, B. Nath, "Creating Multipoint-to-Point LSPs for Traffic Engineering", Proceedings of HPSR'03, Torino (Italy), 24-27 June 2003, pp. 201-207.

[7] T. Li, Y. Rekhter, "A provider Architecure for Differentiated Services and Traffic Engineering (PASTE)", IETF RFC 2430, October 1998

[8] 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.

[9] 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.

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

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

[12] 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.

[13] 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.

[14] 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.

[15] M. Fidler, "Extending the Network Calculus Pay Bursts Only Once Principle to Aggregate Scheduling", Proc. of QoS-IP'03, Milan (Italy), 24-26 Feb. 2003, pp. 19-34.

[16] 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.

[17] M. Fidler, "Providing Internet quality of service based on differentiated services traffic engineering", Ph.D. Thesis, Aachen University, 2003

[18] 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.

[19] J. C. R. Bennett, K. Benson, A. Charny, W. F. Courtney, and 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.

[20] A. Charny, and J.-Y. Le Boudec, "Delay Bounds in a Network with Aggregate Scheduling," Proc. of QoFIS'00, Berlin (Germany), 25-26 September 2000, pp. 1-13.

[21] Y. Jiang, "Delay Bounds for a Network of Guaranteed Rate Servers with FIFO Aggregation," Computer Networks, Vol. 40, No. 6, December 2002, pp. 683-694.

[22] Z.-L. Zhang, Z. Duan, and T.Y. Hou, "Fundamental trade-offs in aggregate packet scheduling" Proceedings of ICNP'01, Riverside (USA), 11-14 November 2001, pp. 129-137.

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

[24] J. A. Cobb, "Preserving quality of service guarantees in spite of flow aggregation", IEEE/ACM Trans. on Networking Vol. 10, No. 1, 2002, pp. 43-53.

[25] R. Fourer and R.E. Marsten, "Solving Piecewise Linear Programs: Experiments with a Simplex Approach", ORSA Journal on Computing, Vol. 4, 1992, pp. 16-31

[26] L. Lenzini, E. Mingozzi, G. Stea, "End-to-end Delay Bounds in FIFO-multiplexing Tandems", Technical Report, University of Pisa, April 2007