A. Bouillard, G. Stea, "Exact Worst-case Delay in FIFO-multiplexing Feed-forward Networks", *IEEE/ACM Transactions on Networking*, vol. 23, no. 5, October 2015, pp. 1387-1400, DOI 10.1109/TNET.2014.2332071

Exact Worst-case Delay in FIFO-multiplexing Feed-forward Networks**

Anne Bouillard and Giovanni Stea‡

June 6, 2014

Abstract

In this paper we compute the actual worst-case end-to-end delay for a flow in a feed-forward network of FIFO-multiplexing service curve nodes, where flows are shaped by piecewise-affine concave arrival curves, and service curves are piecewise affine and convex. We show that the worst-case delay problem can be formulated as a mixed integer-linear programming problem, whose size grows exponentially with the number of nodes involved. Furthermore, we present approximate solution schemes to find upper and lower delay bounds on the worst-case delay. Both only require to solve just *one* linear programming problem, and yield bounds which are generally more accurate than those found in the previous work, which are computed under more restrictive assumptions.

1 Introduction

Several of today's networked applications require deterministic guarantees on the worst-case end-to-end delay that a packet may be experience. This is true of multimedia applications, where packets exceeding the maximum tolerable delay are discarded – with ensuing performance degradation, and all the more for real-time applications involving Machine-to-Machine communications through a multi-hop network. Examples of networks used to carry either or both types of traffic range from multi-service networks, such as the Internet, to dedicated networks, such as Network-on-Chips [15] and AFDX avionic networks [1], wireless sensor networks [16, 24], or industrial Ethernet installations [27]. In several of these, due to scalability reasons, nodes (e.g., switches or routers) maintain a limited number of queues, and traffic belonging to several flows is buffered First-Come-First-Served, or *FIFO*. We call this paradigm *flow aggregation* or

**A preliminary version of this paper appeared at ValueTools 2012 [9]

†A. Bouillard is with the department of Informatics of ENS/INRIA, 45 Rue d'Ulm, 75230 Paris CEDEX 05, France (`Anne.Bouillard@ens.fr`) and has partially carried out the work presented in this paper at LINCS (`www.lincs.fr`).

‡G. Stea is with the Department of Information Engineering of the University of Pisa, Largo L. Lazzarino 1, 56122, Pisa, Italy (`g.stea@iet.unipi.it`).

multiplexing. For instance, in the Internet domain, Behavior Aggregates of Differentiated Services networks (DiffServ [6]), or traffic trunks in Multi-Protocol Label Switching (MPLS, [23]), are in fact composed of many flows sharing the same queue at each node. In the above context, assessing the worst-case delay for single traffic flows is a challenging task, since the latter depends on the behavior of all the other flows that the *tagged* flow, i.e., the one being investigated, may find along its path.

In the recent past, Network Calculus (NC, [12, 13, 17]) a theory for deterministic network performance analysis, has been employed to compute upper and lower bounds on the worst-case delay (WCD) of a flow traversing a tandem of nodes employing flow aggregation. Both FIFO and *blind* multiplexing, i.e., one where no assumption is made regarding the order of queueing of packets from different flows, have been analyzed using NC. For feed-forward networks of blind multiplexing nodes, the exact WCD has been computed in [7], where authors show that the problem is NP-hard in general, but becomes polynomial for tandem networks. This result is, in theory, applicable to FIFO-multiplexing networks as well, FIFO multiplexing being a sub-case of blind multiplexing. However, in this case, it yields an overly pessimistic *upper bound* on the WCD, rather than the WCD itself. As far as FIFO multiplexing is concerned, tandem networks have been analyzed using NC, and both upper and lower bounds on the WCD have been derived using a method called *Least Upper Delay Bound* (LUDB) [18, 20, 3, 4, 5]. The LUDB method relies on using *equivalent service curves*, obtained by progressively removing the flows that intersect the path of the tagged flow. However, equivalent service curves often introduce *pessimism*, hence the LUDB is not guaranteed to be equal to the WCD. For some specific networks e.g., sink-tree networks ([18]), it actually is, but [3] shows that this is not always the case, even in simple tandems. Work [5] compares the LUDB and lower bounds on the WCD in several scenarios, showing that there are cases when the two diverge, and provides further arguments against the tightness of the LUDB in the general case. Furthermore, the only topologies that have been analyzed so far using this method are tree networks ([18, 20]) and tandems ([20, 4, 5]). No generalization to feed-forward networks has ever been presented. Therefore, computing *the* WCD in a general feed-forward FIFO-multiplexing network remains an open problem.

In this paper, we solve the above problem using a Mathematical Programming (*MP*) approach. Unlike the LUDB method, we avoid using equivalent service curves. Instead, we characterize the nodes through accurate input-output relationships, derived by combining aggregate service-curve constraints and the FIFO property. This way, we are able to formulate the WCD problem as a *Mixed Integer-Linear Programming* (MILP) problem, which can be solved by standard solvers (e.g., CPLEX). We show that, its theoretical significance notwithstanding, MILP-based computation of the WCD is feasible only at small network sizes (i.e., up to six or seven nodes). However, the MILP formulation can be modified to obtain very good upper and lower bounds, which require solving only *linear* programs. MP-based upper and lower bounds can be obtained much faster, and they are nearly always surprisingly close (i.e., less than 1% apart in practical

cases). More importantly, the LP-based upper bound does not diverge from the WCD, hence it can be used in lieu of the LUDB when the latter is known to be unreliable. The MP approach presents additional points of strength: unlike the LUDB, it can be applied to *any* feed-forward network. Furthermore, it can be used with *arbitrary piecewise-affine concave* arrival curves and for *arbitrary piecewise-affine convex* service curves, whereas the LUDB only works with simpler single leaky-bucket arrival curves and rate-latency service curves. The first generalization allows *double* leaky-bucket curves to be used, which are of practical significance (see, e.g., [10, 15]), since they model finite peak-rate constraints. Moreover, the MP approach can be used to obtain the worst-case backlog at each node. Finally, it yields exact results with fluid-flow traffic, but it can be used to compute *bounds* with packetized traffic as well.

We evaluate the MP approach numerically in the two dimensions of computation time and bound accuracy, comparing it against the LUDB in several scenarios. Our results show that the MP upper bound outperforms the LUDB as for accuracy (the ratio between the two being as low as 40% in some cases). Such accuracy, however, comes at the expenses of an increase in computation time. The MP upper bound does scale worse than the LUDB, although such a comparison is rendered somewhat unfair by the different structure of the software employed, i.e., a general-purpose optimizer against a highly specialized software (called DEBORAH, DElay BOund Rating AlgoritHm, [4]), whose performance enhancement was itself the subject of a paper [5].

The rest of the paper is organized as follows: Section II reports some background and notation on Network Calculus. In Section III we introduce the hypotheses and state our problem formally. We present our contribution in Section IV. Section V discusses the related work in detail. We report evaluation results in Section VI. Finally, conclusions are reported in Section VII.

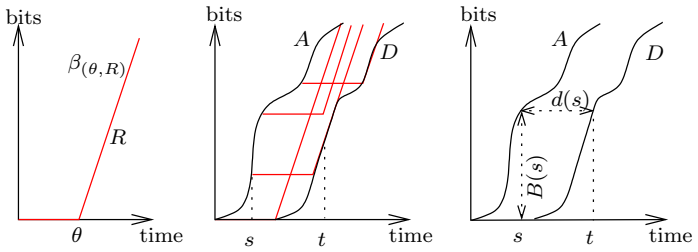

Figure 1: Network calculus concepts: a rate-latency service curve (left), the convolution property: at time t, $D(t) = A(s) + \beta_{(\theta,R)}(t - s)$ (center), backlog and delay computation for a bit arriving at time s (right).

2 Network Calculus background

This section introduces basic Network Calculus concepts, using the same notation as in [17]. In NC, a data flow is described by means of a wide-sense increasing and left-continuous cumulative function $R : \mathbb{R}_+ \to \mathbb{R}_+$, where $R(t)$ is the number of bits seen on the flow in time interval $[0, t[$. In particular, $R(0) = 0$.

A wide-sense increasing function α is said to be an *arrival curve* for a flow characterized by a cumulative function A (or, equivalently, A is α-*upper constrained*) if:

$$\forall s \leq t, \quad A(t) - A(s) \leq \alpha(t - s).$$

As an example, a common *leaky-bucket shaper*, with sustainable *rate* ρ and *burst size* σ, enforces the concave affine arrival curve $\gamma_{\sigma,\rho}(t) = \sigma + \rho t$.

Let A and D be the *Cumulative Arrival* and *Cumulative Departure* functions (CAF and CDF) characterizing the same data flow at the input and output of a network element (or *node*), respectively. If the network element is lossless and does not create any data (an assumption we stick to throughout the paper), it is $A \geq D$. This said, the network element can be modeled through the *service curve* β if:

$$\forall t \geq 0, \quad D(t) \geq \inf_{0 \leq s \leq t} A(s) + \beta(t - s). \tag{1}$$

The flow is said to be guaranteed the (minimum) service curve β. The infimum on the right side of Eq. (1), as a function of t, is called the min-plus convolution of A and β, and is denoted by $A \otimes \beta$. Min-plus convolution is commutative and associative. Figure 1, center, shows how convolution is computed by sliding β along A and taking the infimum for each time. A well-known property of convolution is given by the following lemma (see again Figure 1, center):

Lemma 1. *[17, Chapter 1.3] If f and g are left-continuous and wide-sense increasing, then for all $t \geq 0$, there exists $s \leq t$ such that $f \otimes g(t) = f(s) + g(t - s)$.*

Several network elements, such as delay elements, packet schedulers, links, and regulators, can be modeled through service curves. Many (e.g., most packet schedulers) can be modeled through *rate-latency* service curves, defined as:

$$\beta_{\theta,R}(t) = R(t - \theta)_+.$$

for some $\theta > 0$ (the latency) and $R > 0$ (the rate). Notation $(.)_+$ denotes $\max(., 0)$ and should not be confused with $f(t^+)$ which instead denotes the right limit of f at t in the following. For instance, a constant-rate server (e.g., a wired link) can be modeled as a rate-latency curve with a null latency. Note that rate-latency curves are convex. A fundamental result of Network Calculus is that the service curve of a tandem of network elements traversed by the same flow is obtained by convolving the service curves of each network element.

In a network element that serves bits in FIFO order, one can determine the delay of each bit and the backlog at each time instant by comparing the CAF

and the CDF. More specifically, the delay of a bit arriving at s is equal to:

$$d(s) = \inf\{u \geq 0 : A(s) \leq D(s+u)\}.$$

In other words, it is the horizontal distance between the CAF and the CDF measured at time s on the former. If A and D are continuous, then it is $D(t) = A(s)$, where $t = s+d(s)$ and $d(s)$ is the smallest value that satisfies this equation (see Figure 1, right). The same holds for a tandem of N nodes traversed by a flow: the end-to-end delay of the bit arriving at time t is the horizontal distance between point $(t, A(t))$ on the CAF at node 1 and the point at the same quota of the CDF at node N. An *upper bound* on the delay for a flow can be computed by combining its arrival curve α and the service curve β of the (tandem of) node(s) it traverses. The delay bound is:

$$h(\alpha, \beta) = \sup_{t \geq 0}[\inf\{d \geq 0 \mid \alpha(t-d) \leq \beta(t)\}]. \tag{2}$$

Intuitively, $h(\alpha, \beta)$ is the amount of time the curve α must be shifted forward in time so that it lies below β. The backlog at time s is $B(s) = A(s) - D(s)$, i.e., the vertical distance between the CAF and the CDF at time s. In a similar way as for the delay, the backlog is upper bounded by:

$$v(\alpha, \beta) = \sup_{t \geq 0}[\alpha(t) - \beta(t)]. \tag{3}$$

FIFO Multiplexing

All the above modeling holds when a *single flow* traverses a node (or tandem thereof). Under FIFO multiplexing, traffic of flows arriving at a node are buffered First-Come-First-Served in a single queue. Thus, a bit of a tagged flow arriving at time t will depart only when all the traffic arrived before time t (and belonging to any flow traversing that node) has left. This property (henceforth referred to as the *FIFO property*) is a strong hypothesis, which allows one to compute the CDFs of *single flows* from their CAFs, despite multiplexing. We exemplify this in Figure 2, where two flows, 1 and 2, traverse a FIFO-multiplexing node characterized by a rate-latency service curve $\beta_{\theta,R}$ (this can obviously be generalized to any number and shape of CAFs and any service curve). The two piecewise-linear CAFs, A_1 and A_2 are shown at the bottom-left corner. What the node really sees as an input is the *multiplexing* of the two CAFs, i.e., their aggregate $A = A_1 + A_2$, shown in the top graph. The node transforms the aggregate CAF A into an aggregate CDF D, which is wide-sense increasing and satisfies Eq. (1). The figure reports the one obtained when equality holds in Eq. (1), which is always continuous if the service curve rate is finite, and has a slope R. Now, take a point on D, e.g., $(t, D(t))$. There exists a unique $s \leq t$ such that $A(s) \leq D(t) \leq A(s^+)$ (even if A is discontinuous), and all the traffic that has departed by t has arrived by s (recall that CAFs are left-continuous). Then, $D_1(t)$ and $D_2(t)$ – which are such that $D_1(t) + D_2(t) = D(t)$ – must also satisfy $A_i(s) \leq D_i(t) \leq A_i(s^+)$, $i \in \{1, 2\}$, otherwise there would be some bit in

either CAF that arrived *later than s and departed by t*, which would make the node non-FIFO. This can be exploited to compute $D_1(t)$ and $D_2(t)$ (bottom of Figure 2). In fact, three cases are possible:

1. If A_2 is discontinuous in s (e.g., $A_2(s^+) = A_2(s) + \sigma$), but A_1 is not, then on some non trivial interval that includes t, D_1 is constant, while D_2 has the same slope as D, i.e., R. This is what happens at time t in Figure 2.

2. If, instead, D is affine with slope R in the interval $[t_1, t_2]$, and on the *corresponding* interval $[s_1, s_2]$ (i.e., the interval when the bits that depart in $[t_1, t_2]$ arrive at the input), A_i is affine with slope ρ_i, $i \in \{1, 2\}$, then D_i is affine on the interval $[t_1, t_2]$ with slope $\frac{\rho_i}{\rho_1+\rho_2} R$. This is what happens, e.g., towards the right edge of the time axis in Figure 2.

3. Finally, if *both* A_1 and A_2 are discontinuous, some additional care must be taken. In this case (not shown in the figure), $D_1(t)$ and $D_2(t)$ are not uniquely defined, hence any wide-sense increasing D_1 and D_2 satisfying the above equalities are possible CDFs.

3 System model

We analyze a network of FIFO multiplexing *nodes*, i.e., queueing points where flows contend for the output link bandwidth (e.g. router interfaces). A flow is a distinguishable stream of traffic, traversing a single *path*, i.e., a sequence of nodes. For instance, path $p = (i_1, \ldots, i_\ell)$ is the one of a flow that enters the network at node i_1, traverses nodes $i_1, \ldots, i_\ell$ in that order and then departs. We restrict ourselves to *feed-forward* networks, i.e., those whose nodes can be numbered in such a way that the path of every flow is an increasing sequence. Many interesting networks are feed-forward, e.g. those having a linear, star or tree topology and those employing spanning-tree routing, up-down routing [26], and turn-prohibition [28]. Thus, we henceforth assume that nodes are labeled as explained above, and paths consist of increasing label sequences. For ease of notation we assume that no two flows traverse the same path p, hence we can identify a flow through its path. This is not restrictive, since two flows traversing the same path of FIFO nodes are subject to the same WCD, and piecewise-affine concave arrival curves are additive. We write $h \in p$ if node h belongs to path p, and define the *front* of a path $p = (i_1, \ldots, i_\ell)$ as $fr(p) = i_1$. We denote by ϵ the empty path. Given a path $p = (i_1, \ldots, i_\ell)$, we define $succ_p(i_j) = i_{j+1}$, for $j < \ell$ and $succ_p(i_\ell) = \epsilon$. We are interested in observing one *tagged flow*.

A feed-forward network can be represented by a Directed Acyclic Graph (DAG), whose vertices are the network nodes. In the latter, a directed edge between nodes i and j exists if $j = succ_p(i)$ for some path p. We denote with $succ(i)$ the set of successors of node i. Therefore, two flows share resources if both their paths traverse the same node. Consider a tagged flow p, whose WCD we want to compute. By the very definition of feed-forward network, we need only consider the sub-graph consisting of the nodes from which the exit node of

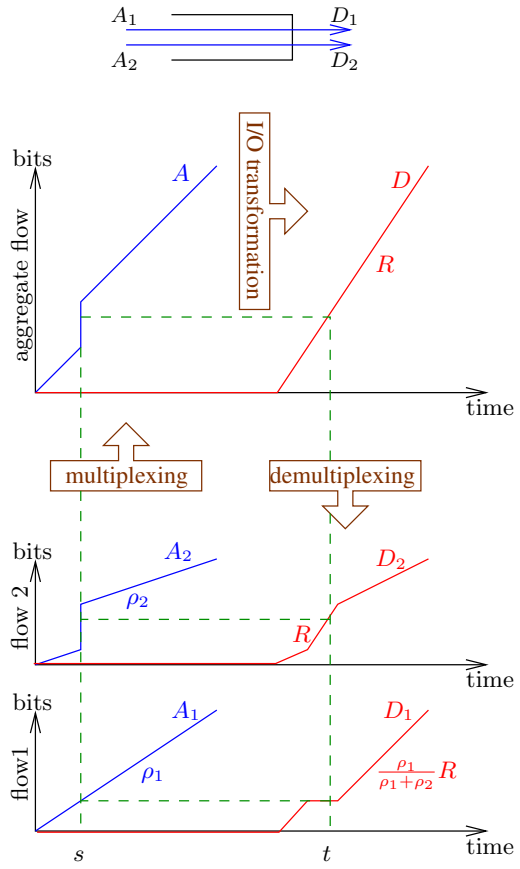

Figure 2: Input-output relationship at a FIFO node.

p can be reached. In fact, what happens at any other node cannot be relevant to the WCD of p. Thus, from now on we will only consider the sub-graph given by the fan-in of p, and number the nodes from 1 to N accordingly, N being the exit node of the tagged flow. With reference to Figure 3, assuming that the tagged flow's path is $p = (5, 6, 7, 8)$, then nodes 9, 10 and 11 need not be included in our model.

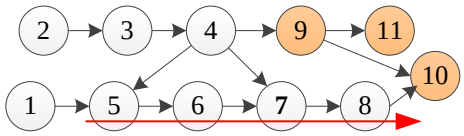

Figure 3: A feed-forward network.

For a flow $p = (i_1, \ldots, i_\ell)$, the CDF at a node i_j is the CAF at $succ_p(i_j)$, hence we need a non-ambiguous notation to identify per-flow cumulative functions: for $h \in p$, we will use $F_p^h(t)$ to denote the cumulative function of flow p observed at the *input* of node h at time t. $F_p^\epsilon(t)$ will denote the CDF of the flow at its exit node. Furthermore, we will occasionally denote the aggregate CAF at node h as $A^h = \sum_{p \ni h} F_p^h$ and the aggregate CDF as $D^h = \sum_{p \ni h} F_p^{succ_p(h)}$, where $p \ni h = \{p \mid h \in p\}$. Nodes will always be indicated in superscripts, and flows/paths in subscripts.

We assume that node h offers a service curve β^h to A^h, and β^h is wide-sense increasing, piecewise affine and convex. Note that both constant-rate curves (e.g. a wired link) and rate-latency curves (which model a multi-queue link arbitrated by a scheduler) fall into this category. Furthermore, we assume that the arrival process of flow p, $F_p^{fr(p)}$, is α_p-upper constrained, where α_p is wide-sense increasing, piecewise affine and concave (i.e., a multiple leaky-bucket curve).

A system is said to be *stable* if there exists a constant C such that the backlog of each node is upper bounded by C. Let $R^h = \lim_{t\to\infty} \beta^h(t)/t$ and $\rho_p = \lim_{t\to\infty} \alpha_p(t)/t$. We assume that the system is stable, that is, $\forall h \in [1, N]$, $R^h \geq \sum_{p \ni h} \rho_p$ (see [17] for an example). Furthermore, we assume that nodes are lossless. We will show later on that this assumption can be verified *a posteriori*, by computing the amount of required worst-case buffer space.

A *scenario* for a feed-forward network with N nodes is a family of cumulative functions F_p^h, p path, $h \in p$, such that:

1. $\forall p, h$, F_p^h are wide-sense increasing and left-continuous;
2. $\forall p$, $F_p^h \geq F_p^{succ_p(h)}$;
3. $\forall p$, $F_p^{fr(p)}$ is α_p-upper constrained;
4. $\forall h \in [1, N]$, $D^h \geq A^h \otimes \beta^h$;
5. Nodes satisfy the FIFO property.

Scenarios describe the feasible trajectories of the system under the assumptions of the system model. We are interested in finding the WCD for a tagged flow p, *i.e.* the maximum delay that one of its bit can experience under all the scenarios.

4 Mathematical programming approach

In this section, we show that the WCD is the optimum of a mixed integer-linear program (MILP), and that – by relaxing and imposing constraints, respectively – upper and lower bounds on the WCD can be obtained as the optima of linear programs. Furthermore, we show that the same model can be adapted to compute the worst-case backlog at a node. We carry out the analysis assuming that traffic is *fluid*, and we discuss how packetization affects our findings at the end of the section. Unfortunately, our approach is notationally heavy. We thus introduce notation and concepts progressively, describing two relatively simple examples first, and then move to considering arbitrary feed-forward networks. A table of symbols is reported in Appendix A for ease of reference.

4.1 Single-node scenario

Let us first focus on the simple, yet meaningful example of a single node traversed by two flows. Note that, for this example only, we have to derogate slightly from our hypotheses and notation and assume that two flows traverse the same path (there being just one). While merging them into a single flow would yield exactly the same results, keeping them separate allows for writing the model in a more general way, which will facilitate understanding later on. We thus denote the two CAFs/CDFs as A_1, A_2 and D_1, D_2. For each departure time t_1 of a bit of data, there exists another time t_2 when that bit arrived (henceforth referred to as the *FIFO time* of t_1, $FIFO(t_1)$), and a time t_3 which verifies the convolution property stated in Lemma 1 at time t_1 (henceforth referred to as the *service curve time* of t_1, $SC(t_1)$). As $\beta \geq 0$, $t_3 \leq t_2$, and obviously $t_2 \leq t_1$ since the system is causal.

Now, the WCD is the maximum horizontal distance between a point that lies on the CDF of the tagged flow – representing one bit of the latter – and the corresponding point "on" its CAF (see, e.g., Figure 2 at times s and t). This is complicated by the fact that CAFs may not be continuous, i.e., the bit we are looking at may have arrived within a burst. However, given a point on the CDF, the point in the Cartesian plane "on" the CAF at the same quota is always defined, even when the CAF is discontinuous (see, e.g., flow 2 in Figure 2). We then set up a linear programming problem with the following variables: *time variables*: t_1, t_2, t_3, which are the abscissas of the points we need to examine; *function variables*: $D_i t_1$, $A_i t_k$, with $i \in \{1, 2\}$, $k \in \{2, 3\}$, where $D_i t_1 = D_i(t_1)$ and:

$$A_i t_k = \begin{cases} D_i t_1 & \text{if } t_k = FIFO(t_1) \\ A_i(t_k) & \text{otherwise} \end{cases}$$

which represent their ordinates. Note that, as shown in Figure 4, variable $A_i t_k$ is always equal to $A_i(t_k)$ *unless* function A_i is discontinuous in t_k. In this last case, it is equal to the number of the bit *inside the burst at* t_k having the same quota as the bit of the CDF that we are looking at, i.e. $D_i(t_1)$.

Then, the WCD is computed by solving the following problem ($i \in \{1, 2\}$):

$$\begin{array}{lll} \max \quad t_1 - t_2 & \text{s.}t. & \\ D_1 t_1 + D_2 t_1 \geq A_1 t_3 + A_2 t_3 + \beta(t_1 - t_3) & & (i) \\ D_i t_1 = A_i t_2 & \forall i & (ii) \\ t_3 \leq t_2 \leq t_1 & & (iii) \\ A_i t_3 \leq A_i t_2 & \forall i & (iv) \\ A_i t_2 - A_i t_3 \leq \alpha_i(t_2 - t_3) & \forall i & (v) \\ t_k \in \mathbb{R}_+, A_i t_k \in \mathbb{R}_+, D_i t_k \in \mathbb{R}_+ & \forall i, k & (vi) \end{array}$$

where (i) is the service curve constraint, (ii) is the FIFO property, (iii) is the time ordering, (iv) is the monotonicity, and (v) is the arrival curve constraint. We use a notation for function variables which mirrors the functional one, so as to allow readers to easily switch between MP constraints (i-v) and the NC properties expressed in the scenario constraints (1-5) reported at the end of Section III.

All the above constraints are linear, or can be easily linearized: if α_1 and α_2 are piecewise affine and concave, then they can be written as

$$\alpha_i(t) = [\min_m Y_{i,m}](t),$$

where $Y_{i,m}$ are affine curves. Hence, arrival curve constraints can be linearized as follows:

$$A_i t_2 - A_i t_3 \leq \alpha_i(t_2 - t_3) \qquad \Leftrightarrow \forall m,\ A_i t_2 - A_i t_3 \leq Y_{i,m}(t_2 - t_3).$$

Thus, each arrival curve constraint yields as many linear constraints as its linear pieces. The same holds, *mutatis mutandis*, for service curve constraints, given that β is piecewise affine and convex. The objective function is linear as well, hence computing the WCD entails solving an LP problem.

4.2 Two-node scenario

Consider now a tandem of two nodes traversed by three flows as in Figure 5. Let the tagged flow be the one traversing the whole tandem, i.e. $p = (1, 2)$. At its exit node 2, both the service curve constraints and the FIFO constraints for flows (2) and $(1, 2)$ can be written down. Therefore, given a departure time at node 2, call it t_1, we can add its FIFO time $t_2 = FIFO^2(t_1)$ and its SC time $t_3 = SC^2(t_1)$ at node 2. We use a node superscript, i.e., $FIFO^h$ and SC^h, to emphasize the node h where the FIFO and SC properties are applied. Times t_2 and t_3 are those at which we observe the *input* of node 2 and the *output* of node 1. We have an arrival curve constraint for flow (2). Instead, we have no arrival

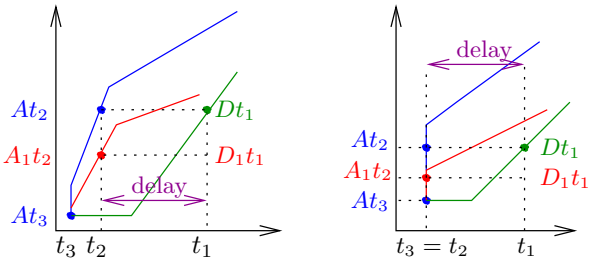

Figure 4: Some variables and constraints for two examples. The blue curve is the CAF $A(t) = A_1(t) + A_2(t)$, the green the CDF $D(t)$. The red curve represents $A_1(t)$. Relevant points of the plane either *on* the curves or included in a discontinuity are also highlighted.

curve constraints for cross-flow (1) and for the tagged flow $(1, 2)$, since these constraints have effect at the ingress of node 1. The inequalities concerning node 2 are hence similar to those of the single-node case, i.e.:

$$\begin{cases} F^{\epsilon}_{(2)}t_1 + F^{\epsilon}_{(1,2)}t_1 \geq F^{2}_{(2)}t_3 + F^{2}_{(1,2)}t_3 + \beta_2(t_1 - t_3) \\ F^{\epsilon}_{(p)}t_1 = F^{2}_{(p)}t_2, \quad p \in \{(2), (1,2)\} \\ t_3 \leq t_2 \leq t_1 \\ F^{2}_{(p)}t_3 \leq F^{2}_{(p)}t_2, \quad p \in \{(2), (1,2)\} \\ F^{2}_{(2)}t_2 - F^{2}_{(2)}t_3 \leq \alpha_{(2)}(t_2 - t_3) \end{cases}$$

Now, in order to compute the end-to-end delay of the bit that departs node 2 at

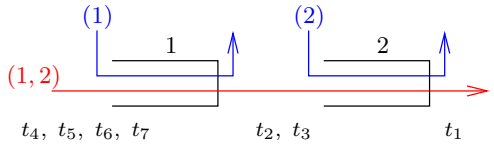

Figure 5: Tandem of two nodes and three flows.

t_1, we need the FIFO time of t_2, i.e. the time at which the same bit enters node 1. In order to compute it, we iterate the same procedure at node 1, now for each of the two times t_2 and t_3 at which we observe its output: let $t_4 = FIFO^1(t_2)$ and $t_5 = SC^1(t_2)$, and similarly let $t_6 = FIFO^1(t_3)$ and $t_7 = SC^1(t_3)$. The bit that exits node 1 at time t_2 entered that node at time t_4. Hence, the WCD is the maximum difference $t_1 - t_4$, which is in fact the objective function to be maximized. To complete the example, we must add the set of constraints related to node 1, i.e. service curve, FIFO, time ordering, monotonicity and arrival constraints.

The service constraints are:

$$F^{2}_{(1,2)}t_i + F^{2}_{(1)}t_i \geq F^{1}_{(1,2)}t_j + F^{1}_{(1)}t_j + \beta^1(t_i - t_j),$$

with $(t_i, t_j) \in \{(t_2, t_5), (t_3, t_7)\}$.

The FIFO constraints are: $F_p^2 t_i = F_p^1 t_j$, with $p \in \{(1), (1,2)\}$ and $(t_i, t_j) \in \{(t_2, t_4), (t_3, t_6)\}$.

The time ordering constraints are:

$$\begin{cases} t_2 \geq t_4 \geq t_5 \\ t_3 \geq t_6 \geq t_7 \\ t_4 \geq t_6 \\ t_5 \geq t_7 \end{cases}$$

Again, the first two follow from causality and from $\beta \geq 0$, same as for node 2. The third one instead arises from the fact that cumulative functions are non decreasing, hence $t_2 \geq t_3$ implies that $t_4 = FIFO^1(t_2) \geq t_6 = FIFO^1(t_3)$. The fourth one derives from the above and from the fact that β is convex, hence $t_2 \geq t_3$ implies that $t_5 = SC^1(t_2) \geq t_7 = SC^1(t_3)$. We leave to the alert reader the straightforward task of adding the monotonicity constraints that are implied by the above time ordering. Unfortunately, although the set of output times at node 1, i.e., t_2, t_3, is totally ordered, the set of *input* times $t_4, \ldots, t_7$ is only *partially* ordered. In fact, the above inequalities are not enough to discriminate whether $t_5 \geq t_6$ or vice-versa. This creates a problem, because we need to ensure that, whichever the order, monotonicity is preserved: we cannot have both $t_5 > t_6$ *and* $F_p^1(t_5) < F_p^1(t_6)$. In principle, monotonicity may be guaranteed by including non-linear constraints, e.g.:

$$(t_5 - t_6) \cdot (F_p^1 t_5 - F_p^1 t_6) \geq 0, p \in \{(1), (1,2)\}.$$

However, these constraints are non-linear and (worse yet) non-convex, hence including them would make the problem very hard to solve in practice. We can instead preserve monotonicity by using *indicator constraints*, i.e. constraints activated by binary variables. We define a binary variable b, a large positive constant M, and we replace the above constraints with the following set:

$$\begin{cases} t_5 + (1-b) \cdot M \geq t_6 \\ t_5 \leq b \cdot M + t_6 \\ F_p^1 t_5 + (1-b) \cdot M \geq F_p^1 t_6 \\ F_p^1 t_5 \leq b \cdot M + F_p^1 t_6 \end{cases}$$

This way, for a large enough M, odd constraints are active if $b = 1$, otherwise they are inactive. The reverse holds for even constraints. This allows us to preserve linearity, hence to take advantage of off-the-shelf MILP solvers (e.g., CPLEX), which are generally faster. Note that CPLEX computes a suitable value for M by itself, which is clearly a plus. From now on, we will use the shorthand $x \geq_b y$ to denote the following pair of constraints:

$$\begin{cases} x + (1-b) \cdot M \geq y \\ x \leq b \cdot M + y \end{cases}$$

where b is a binary variable, M is a large constant and x, y, are arbitrary variables. For ease of notation, we will also use $x \geq_\emptyset y$ to mean $x \geq y$.

Coming to arrival constraints at node 1, those related to pairs of ordered times are the following:

$$F_p^1 t_k - F_p^1 t_\ell \leq \alpha_p(t_k - t_\ell),$$

$(k, \ell) \in \{(4,5),(4,6),(4,7),(5,7),(6,7)\}$, $p \in \{(1),(1,2)\}$.

Furthermore, we must add arrival constraints for unordered times t_5, t_6. In fact, we cannot simply write $F_p^1 t_5 - F_p^1 t_6 \leq \alpha_p(t_5 - t_6)$, since the arrival curve is only defined for non negative arguments (see the definition in Section II). If $t_5 \leq t_6$, then $F_p^1 t_5 \leq F_p^1 t_6$ by monotonicity, hence the correct arrival curve constraint should be $F_p^1 t_6 - F_p^1 t_5 \leq \alpha_p(t_6 - t_5)$. Thus, we proceed as follows: assuming $\alpha_p(t) = \sigma_p + \rho_p \cdot t$ (if α_p is piecewise affine and concave, we can simply iterate the reasoning over all its linear pieces, themselves affine functions as above), it is trivial to show that the two following constraints:

$$\begin{cases} F_p^1 t_5 - F_p^1 t_6 \leq \alpha_p(t_5 - t_6) & \text{if } t_5 \geq t_6 \\ F_p^1 t_6 - F_p^1 t_5 \leq \alpha_p(t_6 - t_5) & \text{if } t_5 \leq t_6 \end{cases}$$

are equivalent to:

$$\begin{cases} F_p^1 t_5 - F_p^1 t_6 \leq \alpha_p(t_5 - t_6) + (1 - b) \cdot M \\ F_p^1 t_6 - F_p^1 t_5 \leq \alpha_p(t_6 - t_5) + b \cdot M \end{cases}$$

The last two constraints are linear. When $b = 1$ the first one is active and the second is inactive, and vice-versa for $b = 0$. Using a similar notation as for the previous case, we will write $F_p^1 t_5 - F_p^1 t_6 \leq_b \alpha_p(t_5 - t_6)$ as a shorthand for the above two linear inequalities (or $2n$ linear inequalities, if α_p is the min of n affine functions). The complete formulation of the above two-node problem is reported in Appendix B for ease of reference.

Putting numbers into the above problem, if we consider $\alpha_{(1,2)}(t) = \alpha_{(1)}(t) = 1 + t/3$, $\alpha_{(2)}(t) = \min(t, 11 + t/3)$, $\beta^1(t) = \beta^2(t) = (t-1)_+$, we obtain that the maximum of $t_1 - t_4$ is equal to 10.167, obtained for $b = 1$, i.e. $t_5 \geq t_6$. The maximum is indeed the WCD, since the constraints describe all the possible scenarios (the formal proof will be given later on). Besides, we observe that, if we do not take into account flow (2)'s peak-rate constraint, assuming instead $\alpha_{(2)}(t) = 1 + t/3$, we obtain a WCD of 15.33, which confirms that allowing for more general, piecewise-affine concave arrival curves is indeed useful.

4.3 General feed-forward network

Summarizing from the previous examples, computing the WCD entails writing a linear programming problem, whose variables are the *times* at which the cumulative functions must be observed, and the cumulative *function values* at these times. For networks consisting of two or more nodes, times are not totally ordered, hence binary variables must be added to preserve monotonicity and correctness of the arrival constraints. It is evident that, if we observe the output of node j at k times, then we need $2k$ times at the input of the same

node, since each output time t spawns two input times $t' = FIFO^j(t)$ and $t'' = SC^j(t)$. Therefore, the number of variables and constraints of the problem is exponential in the number of nodes. For instance, the number of times for a tandem network of N nodes is equal to $2^{N+1} - 1$ (see [9]).

At a high level, the algorithm for writing down the MILP whose optimum solution is the WCD of the tagged flow can be explained as follows. Starting from node (the exit node of the tagged flow N, we visit the nodes in *inverse label order*, until we get to node 1. At each node j, we do the following:

- first, we collect all the output times. These are the input times at all the successors of j (if $j < N$), or set $\{t_1\}$ at node N. Call $\mathcal{T}_{out}^j$ the set of output times at node j;
- then, we construct the set of *input* times for node j, call it $\mathcal{T}_{in}^j$, from the FIFO and SC times of each time in $\mathcal{T}_{out}^j$;
- we then order the times in $\mathcal{T}_{out}^j$. The latter, in fact, may not be totally ordered if j has more than one successor. If this is the case, binary variables are required;
- we then derive a total order for times in $\mathcal{T}_{in}^j$, starting by inferring inequalities from the ones in $\mathcal{T}_{out}^j$, and adding binary variables when needed;
- we finally write down all the constraints related to node j (i.e., FIFO, service curve, time ordering, monotonicity, arrival constraints).

The fact that we proceed in inverse label order ensures us that set $\mathcal{T}_{out}^j$ is always well-defined, since we already possess the sets $\mathcal{T}_{in}^k$of all nodes k that are reachable from j. Finally, in order to write down the objective function, we start from time t_1 at the output of node N and trace back the FIFO times by following backwards the path of the tagged flow until its ingress node. As an example, consider the three-node network of Figure 6. The figure reports the sets $\mathcal{T}_{out}^j$ and $\mathcal{T}_{in}^j$ for all the nodes, which are computed from right to left. Assume that, at node 1, input times are added by following the order of the output times, so that $t_8 = FIFO^1(t_2)$, $t_9 = SC^1(t_2)$, $t_{10} = FIFO^1(t_3)$, and so on until $t_{19} = SC^1(t_7)$. Then, if we assume that the tagged flow is $(1, 3)$, the objective function will be $t_1 - t_8 = t_1 - FIFO^1(FIFO^3(t_1))$. If, instead, we want to compute the WCD for flow $(1, 2, 3)$, we will need to maximize $t_1 - t_{12} = t_1 - FIFO^1(FIFO^2(FIFO^3(t_1)))$.

We now describe the constraints formally.

Times and their ordering constraints We define the time variables recursively (from N to 1) as:

- $\mathcal{T}_{out}^N = \{t_1\}$;
- $\forall j \in \{1, \ldots, N\}$, $\mathcal{T}_{in}^j = FIFO^j(\mathcal{T}_{out}^j) \cup SC^j(\mathcal{T}_{out}^j)$;
- $\forall j \in \{1, \ldots, N-1\}$, $\mathcal{T}_{out}^j = \cup_{k \in succ(j)} \mathcal{T}_{in}^k$,

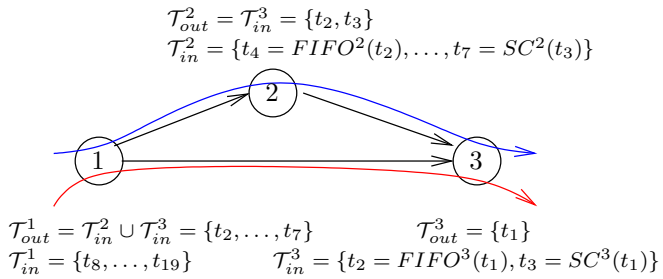

Figure 6: Three-node network.

where for a set $\mathcal{S}$, $FIFO^j(\mathcal{S}) = \{FIFO^j(t) \mid t \in \mathcal{S}\}$, (the same holds for $SC^j(\mathcal{S})$), and $FIFO^j(t)$ (resp. $SC^j(t)$) entails creating a new variable. Note that sets $\mathcal{T}^j_{in}$ are disjoint.

We now need to derive a set of time inequalities to ensure the monotonicity of functions and the arrival constraints. We do not need to order *every* pair of times, rather, only those that appear in the same constraint. For example, for the tandem of subsection IV-B, the order of t_2 and t_7 is irrelevant. Thus, we need a total order on $\mathcal{T}^j_{in}$ and $\mathcal{T}^j_{out}$ for all j.

Ordering constraints for $\mathcal{T}^j_{in}$ are of three types:

- *known*: those included in $\mathcal{K}$, defined as the transitive closure of the following set:

$$\begin{aligned}&\{t \geq FIFO^j(t) \geq SC^j(t) \mid 1 \leq j \leq N,\ t \in \mathcal{T}^j_{out}\}\\ \cup\ &\{FIFO^j(t) \geq FIFO^j(t') \text{ and } SC^j(t) \geq SC^j(t') \mid\\ &\qquad 1 \leq j \leq N,\ t, t' \in \mathcal{T}^j_{out},\ t \geq t'\}\end{aligned}$$

- *inherited*: those concerning pairs of times $t_x,\ t_y \in \mathcal{T}^j_{in}$, whose order can be inferred from the order of two times (of its successors) t, t', such that $t \geq_b t'$. The "old" binary variable b will thus be inherited to order these times as well, i.e. $t_x \geq_b t_y$;

- *new*: those concerning the pair of times in $\mathcal{T}^j_{in}$ whose order is neither known nor inherited from successors. For these, a "new" binary variable must be defined.

For instance, for the network of Figure 6, we have that $t_5 \geq_b t_6$, which is a new constraint at the input of node 2. Furthermore, at the input of node 1, we have $t_{14} = FIFO^1(t_5)$, $t_{15} = SC^1(t_5)$, $t_{16} = FIFO^1(t_6)$, hence $t_{14} \geq_b t_{16}$ is an inherited constraint, whereas t_{15} and t_{16} must instead be ordered through a new binary variable.

Now, the ordering of $\mathcal{T}_{in}^j$ depends on the ordering of $\mathcal{T}_{out}^j$, and that of $\mathcal{T}_{out}^j$ will depend on $\mathcal{T}_{in}^k$, where k is a successor of j. Hence, in order to sort the times, we need to proceed recursively backwards. Let us denote by $\mathcal{I}_{in}^j$ (resp. $\mathcal{I}_{out}^j$) the set of inequalities required to totally order $\mathcal{T}_{in}^j$ (resp. $\mathcal{T}_{out}^j$). $\mathcal{I}_{in}^j$ is built by adding known constraints first, then those inherited from $\mathcal{I}_{out}^j$, and finally adding new binary variables when required. $\mathcal{I}_{out}^j$, instead, is built by assembling sets $\mathcal{I}_{in}^k$ for all nodes $k = succ(j)$, and then ordering unordered times (i.e., belonging to different successors) by adding new variables. The starting point is $\mathcal{I}_{out}^N = \emptyset$, from which every other set can be computed. The pseudocode for computing sets $\mathcal{I}_{in}^j$ and $\mathcal{I}_{out}^j$ is shown in Appendix C. Let $\mathcal{C}_T$ denote the set of time constraints, i.e.:

$$\mathcal{C}_T = \bigcup_{1 \leq j \leq N} (\mathcal{I}_{in}^j \cup \mathcal{I}_{out}^j).$$

Function variables and their constraints Now that all the time constraints are set, we can write down all the function constraints, which depend on the latter. For each node j, we have the following constraints:

FIFO constraints: $\forall p \ni j$, $\forall t \in \mathcal{T}_{out}^j$, and $t' = FIFO^j(t)$,

$$F_p^j t' = F_p^{succ_p(j)} t.$$

SC constraints: $\forall t \in \mathcal{T}_{out}^j$, and $t' = SC^j(t)$,

$$\sum_{p \ni j} F_p^{succ_p(j)} t = \sum_{p \ni j} F_p^j t' + \beta^j (t - t').$$

Monotonicity constraints: $\forall p \ni j$, $\forall t \geq_b t' \in \mathcal{I}_{in}^j$,

$$F_p^j t \geq_b F_p^j t',$$

and $\forall t \geq_b t' \in \mathcal{I}_{out}^j$,

$$F_p^{succ_p(j)} t \geq_b F_p^{succ_p(j)} t'.$$

Arrival constraints: $\forall p \ni j$, such that $j = fr(p)$, $\forall t \geq_b t' \in \mathcal{I}_{in}^j$,

$$F_p^j t - F_p^j t' \leq_b \alpha_p (t - t').$$

Call $\mathcal{C}_F$ the set of all the function variables constraints.

Objective If the objective is to compute the maximum delay of flow $(j_1, \ldots, j_k)$ with $j_k = N$, then set $t_0 = FIFO^{j_1}(FIFO^{j_2}(\cdots(FIFO^{j_k}(t_1))))$. Our MILP is then:

$$(*) \qquad \text{Maximize } (t_1 - t_0) \text{ subject to constraints } \mathcal{C}_T \cup \mathcal{C}_F.$$

We now prove that the optimum of $(*)$ is the WCD.

Theorem 1. *The optimum of* $(*)$ *is the WCD for the flow of interest.*

We give the proof in two separate lemmas. Lemma 2 states that, given a scenario with a delay d, $(*)$ has a solution with the same delay. Lemma 3 shows that, given a solution of $(*)$, there is a scenario having at least the same delay.

Lemma 2. *Let F be a scenario of the network with delay d. There exists a feasible solution in $(*)$ such that $t_1 - t_0 = d$.*

Proof. Let $F = (F_p^j)$ be a scenario and t_1 be the time of departure of the bit of interest (the one that experiences delay d). Recall that $A^j = \sum_{p \ni j} F_p^j$, $D^j = \sum_{p \ni j} F_p^{succ_p(j)}$, and that cumulative functions are left-continuous. Then, by Lemma 1 we can find a time t_3 such that $D^N(t_1) \geq A^N(t_3) + \beta^N(t_1 - t_3)$ and a time t_2 such that $\forall p \ni N$, $F_p^N(t_2) \leq F_p^\epsilon(t_1) \leq F_p^N(t_2^+)$. In $(*)$, we set time variables $t_2 = FIFO^N(t_1)$, $t_3 = SC^N(t_1)$, and the related function variables according to the trajectories, i.e., $F_p^N t_3 = F_p^N(t_3)$ and $F_p^\epsilon t_1 = F_p^\epsilon(t_1)$, but with $F_p^N t_2 = F_p^\epsilon t_1$, in order to account for discontinuities as explained in Section IV-A. By backward induction, we can go on defining times and function variables: if t and $F_p^{succ_p(j)} t$ are defined, one can find t'' such that $D^j(t) \geq A^j(t'') + \beta^j(t - t'')$ and t' such that $F_p^j(t') \leq F_p^{succ_p(j)} t \leq F_p^j(t'^+)$. Time variables are set accordingly: $t' = FIFO^j(t)$, $t'' = SC^j(t)$; for t' we set $F_p^j t' = F_p^{succ_p(j)} t$, again to account for discontinuities.

After the above variables have been assigned, we enforce a total order on the times related to node j: if $t < t'$, then we add constraint $t \leq t'$ to $(*)$. If $t = t'$ and there exists p such that $F_p^j(t) < F_p^j(t')$, then we also set $t \leq t'$. Since we started from a feasible scenario, one cannot have $F_p^j(t) < F_p^j(t')$ *and* $F_{p'}^j(t) > F_{p'}^j(t')$ for two different flows p and p'. Binary variables in $(*)$ are assigned using this total order. Once this is done, since we started from a feasible scenario, cumulative functions are non-decreasing and the arrival processes $F_p^{fr(p)}$ are α_p-upper constrained. The corresponding constraints in (*) are thus satisfied with the above-mentioned assignment of function variables, times and binary variables.

To conclude the proof, we just need to observe that t_1 belongs to the CDF of the last node by definition, and that $t_0 = FIFO^{j_1}(FIFO^{j_2}(\cdots(FIFO^{j_k}(t_1))))$. Then, $d = t_1 - t_0$ is a solution of $(*)$. □

Lemma 3. *For any solution of $(*)$ such that $t_1 - t_0 = d$, there exists a scenario of the system where the bit that leaves at t_1 has a delay at least equal to d.*

Proof. Consider a solution of $(*)$. We will construct a scenario $F = (F_p^j)$ that verifies constraints 1-5 listed at the end of Section III. Fix a path p, a node j and consider pairs of variables t_k, $F_p^j t_k$. Let $E_{t_k} = \{t_h \mid t_h = t_k\}$. We set $F_{(p)}^j(t_k) = \min_{t_h \in E_{t_k}} F_p^j t_h$ for each $j \in p$ hence these functions are defined at times t_k in $(*)$, and constraints therein ensure that they are non-decreasing, if observed at these times only. For the starting node of a path $j = fr(p)$ we extrapolate from the above values F_p^j as the largest α_p-upper constrained

function. Such function is (see [8], Lemma 2):

$$F_p^j(t) = \min(\min\{F_p^j(t_k) + \alpha_p(t - t_k) \mid t \le t_k\}, \\ \min\{F_p^j(t_k) \mid t \ge t_k\}).$$

This function is continuous except at some times where the variable $F^{fr(p)}t_k$ exists. In that case, due to arrival and monotonicity constraints, $F_p^{fr(p)}(t_k^+) \ge \max_{t_h \in E_{t_k}} F_p^{fr(p)} t_h$. If j is a generic node in p (but different from N), we extrapolate:

$$F_p^{succ_p(j)}(t) = \min[F_p^j(t), \min_{t_k \ge t} F_p^{succ_p(j)}(t_k)],$$

so that outside the times t_k defined in $(*)$, $F_p^{succ_p(j)}$ is made of bursts only or it is equal to its predecessor's CDF. Finally, if $j = N$, F_p^ϵ are defined so that the FIFO order is preserved and $F_p^\epsilon(t) = F_p^N(t)$ for $t \le t_3$ and $D^N(t) = \max(A^N(t_3), A^N \otimes \beta^N(t))$ for $t \ge t_3$. Due to the latter equality (service curve constraint), we may have to replace t_1 by $t_1' > t_1$, to maintain $F_p^\epsilon(t_1') = F_p^N t_2$. It is always possible for a single server to set the CDFs so that the FIFO order is preserved. The above extrapolation yields wide-sense increasing, left-continuous functions.

For each pair of times defined in $(*)$, the FIFO order is preserved and $D^h \ge A^h \otimes \beta^h$, since the constraints ensure it. Now, we move to considering times which are not part of problem $(*)$: for $t_h < t \le t_k$, where t_h and t_k are consecutive times defined at server j in $(*)$, first:

$$F_p^{succ_p(j)}(t) = \min(F_p^j(t), F_p^{succ_p(j)}(t_k)) = \\ \min(F_p^j(t), F_p^j FIFO^j(t_k)).$$

Now, either of two cases are given:

1. $t < FIFO^j(t_k)$ and $F_p^{succ_p(j)}(t) = F_p^j(t)$
2. $t \ge FIFO^j(t_k)$ and $F_p^{succ_p(j)}(t) = F_p^j FIFO^j(t_k)$.

As $FIFO^j(t_k)$ does not depend on p, given a time t, the same case will hold for *each* path p traversing j. In both cases, the FIFO order is preserved. This property also implies that $D^h(t) = \sum_{p \ni h} F_p^{succ_p(h)}(t) = A^h(t)$ if $t < FIFO^j(t_k)$ or $D^h(t) = \sum_{p \ni h} F_p^j FIFO^j(t_k) = \sum_{p \ni h} F_p^{succ_p(j)}(t_k) = D^h(t_k)$ if $t \ge FIFO^j(t_k)$. Then:

$$D^h(t) = \min(A^h(t), D^h(t_k)) \ge \\ \min(A^h(t), A^h \otimes \beta^h(t_k)) \ge A^h \otimes \beta^h(t),$$

hence the service constraints are satisfied. This also holds for node N by construction. We have then built a scenario that verifies properties 1-5 in Section III. The arrival time of the bit of data leaving at time t_1' is t_0, so the delay of this bit is $t_1' - t_0 \ge t_1 - t_0 = d$, hence it is at least equal to the one of the solution of $(*)$. □

4.4 Upper and lower bounds on the WCD

Computing the WCD requires solving a MILP, whose number of variables and constraints grows exponentially with the number of nodes. However, it is easy to find good bounds on the WCD at a reasonable computation cost, i.e., solving just one *linear program*. An *upper bound* can be found by giving up monotonicity (which, we recall, is what puts binary variables into the picture). We only keep the partial ordering of time and function variables, the service constraints, and those arrival constraints that follow from pairs of ordered times (e.g., the first set in the two-node example in the Appendix), and solve one LP with these constraints. Since this problem is a relaxation of the one for the WCD, its optimum is clearly an upper bound on the WCD, call it V_{LP}. This may indeed verify all monotonicity constraints (which can be easily checked *a posteriori*), in which case it is *the* WCD.

Similarly, if we enforce manually a total order on the times (and on the corresponding function values), we obtain a feasible scenario, i.e. one where all cumulative functions satisfy all the constraints, and we can dispense with the binary variables. This way, we instead obtain a *lower bound* on the WCD. An efficient way to obtain a lower bound is to *reduce the number of times*, by imposing additional constraints on the latter. We do this by forcing the SC times related to *all* the times of a node to be equal. In other words, if $\mathcal{T}_{out}^j$ is the set of times at the output of node j, then we add the constraints:

$$\forall t_x, t_y \in \mathcal{T}_{out}^j, SC^j(t_x) = SC^j(t_y). \tag{4}$$

This way, instead of doubling, the number of times only increases by one at each node. The alert reader will observe that, starting from a totally ordered $\mathcal{T}_{out}^j$, Eq. (4) yields a totally ordered $\mathcal{T}_{in}^j$: this is because FIFO times inherit a total order from the output times, and the earliest FIFO time is larger than the corresponding SC time by definition. Therefore, Eq. (4) is sufficient to produce the required time order in every network whose DAG is a tree (therein including tandem networks). For general DAGs, where a node j may have two or more successors, we still need to enforce a total order on $\mathcal{T}_{out}^j$. The rule that we adopt is to sort the times based on the node label: if $\mathcal{T}_{out}^j = \mathcal{T}_{in}^k \cup \mathcal{T}_{in}^l$, with $k > l$, then we set $\min\{t|t \in \mathcal{T}_{in}^k\} \geq \max\{t|t \in \mathcal{T}_{in}^l\}$. We call v_{LP} the lower bound thus obtained.

Note that the complexity of computing the two bounds is different: the upper bound requires solving one LP, whose size (in terms of variables and constraints) is always *exponential* with the number of nodes, hence it is still an exponential problem. The number of times for the lower bound, instead, varies with the topology. For instance, when the DAG is a tree, it is quadratic, hence *polynomial* with the number of nodes. Since LPs can be solved in polynomial time, computing the lower bound is itself a polynomial problem in that case. In any case, the number of times of the lower-bound problem is always considerably smaller than the one of the upper-bound problem for the same topology. In the next section we will show how accurate the two bounds are, and how costly it is to compute them.

4.5 Worst-case backlog at nodes

The same model used so far lends itself to computing the *worst-case backlog* (WCB) at a node (or bounds on the latter). This is especially useful to dimension the buffers at each nodes so that no overflow occurs. Without loss of generality, we can assume that we compute the WCB at node N (otherwise we just need to remove some nodes from the DAG).

Let t_1 be the time at which the WCB occurs. Then, we have to modify our linear program as follows:

- $\forall j \in \{1, \ldots, N\}$, add t_1 to every set $\mathcal{T}_{out}^j$ and $\mathcal{T}_{in}^j$;
- $\forall j$, $\forall t \in \mathcal{T}_{in}^j$ add $t_1 \geq t$ to $\mathcal{T}_{in}^j$;
- $\forall j$, $\forall p \ni j$, $\forall t \in \mathcal{T}_{in}^j$, write the monotonicity and the arrival constraints related to t_1: $F_p^j t_1 \geq F_p^j t$ and if $j = fr(p)$, then $F_p^j t_1 - F_p^j t \leq \alpha_p(t_1 - t)$.

The WCB at node N is then obtained by maximizing

$$\sum_{p \ni N} F_p^{fr(p)} t_1 - \sum_{p \ni N} F_p^{\epsilon} t_1.$$

Indeed, as we have no constraints on the maximum service rate, the latter can become infinite at time t_1, and all the data that arrived in the network and exit the network at node N can be served instantaneously at every node except N.

To illustrate why we need to consider infinite service, consider the following simple case: a two-node tandem traversed by one flow. Figure 7 shows the three cumulative functions $F^j, j \in \{1, 2, \epsilon\}$ and the backlog at node 2. On the left, the service is *exact*, meaning that $F^2 = F^1 \otimes \beta^1$ and $F^\epsilon = F^2 \otimes \beta^2$. On the right, the service rate at node 1 becomes infinite at time t_1. It is obvious that the maximum backlog at node 2 is larger in this second case.

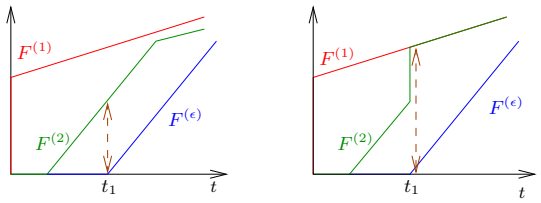

Figure 7: Maximum backlog and infinite service.

4.6 Packetization

All the results shown so far hold for fluid-flow traffic. When traffic is packetized, CAFs and CDFs are staircase functions, as in Figure 8, whose steps occur

whenever the *last bit* of a packet arrives or departs. Network calculus accounts for packetization as follows [17]: if a node offering β as a service curve is traversed by packetized traffic whose maximum packet length is ℓ, then a lower bound on the packetized CDF will be obtained by using $\tilde{\beta} = (\beta - \ell)_+$ in (1). In other words, $\tilde{\beta}$ is a service curve for the packetized traffic, and it is $\tilde{\beta} < \beta$. As the figure shows, the only points where the packetized CDF attains its lower bound are the departure times of maximum-sized packets. Let ℓ_p denote the maximum packet size for flow p. At node j, we then have $\tilde{\beta}^j = (\beta^j - \max_{p \ni j}\{\ell_p\})_+$. Call $(\tilde{*})$ the MILP obtained from $(*)$ when $\tilde{\beta}^j$ is substituted to β^j at each node $j < N$ ([17] shows that the fluid SC β^N can still be used at the *last* node). Direct consequences of Theorem 1 are:

- the optimum of $(*)$ is a *lower bound* on the packet WCD. This is because a scenario of $(*)$ is feasible in a packetized environment as well, if we consider that every flow can always send arbitrarily small packets.
- the optimum of $(\tilde{*})$ is an *upper bound* on the packet WCD, since it uses lower bounds on the service.

The above two properties imply that v_{LP} and $\tilde{V}_{LP}$ are also lower and upper bounds for the packet WCD, and they are faster to compute. The same reasoning holds, *mutatis mutandis*, for the packet WCB.

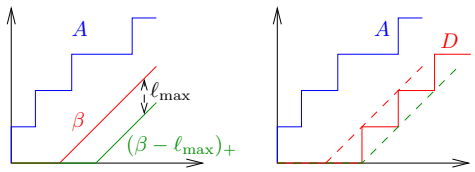

Figure 8: Packetized flow and service curves.

5 Related work

The starting point for a review on delay bounds in FIFO networks is [11], which shows that, for a *general* FIFO network (i.e., not necessarily feed-forward), upper bounds on the WCD can only be computed for small utilization factors. A *critical utilization factor* ν is defined, which is inversely proportional to the maximum path length. For a utilization $u \leq \nu$ the bound is proportional to $1/(\nu - u)$, hence approaches infinity as the utilization approaches ν. Moreover, for any utilization $u > \nu$ and finite delay d, it is possible to construct a (non feed-forward) network where some traffic exhibits a delay larger than d. However, FIFO *feed-forward* networks are known to be stable for any utilization up to 100%, hence better bounds can be found in the latter.

Some papers related to delay bounds in FIFO *tandem* (or *linear*) networks have appeared recently [19, 18, 20, 3, 4, 5, 10, 15]. They all rely on computing

and manipulating *equivalent service curves* for a tagged flow. At a single node, this is done through the following theorem:

Theorem 2. *[17, Chapter 6] Consider a node serving two flows, 1 and 2, in FIFO order. Assume that the node guarantees a minimum service curve β to the aggregate of the two flows and that flow 2 has α_2 as an arrival curve. Define the family of functions:*

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

where $1_{t>\tau}$ is equal to 1 if $t > \tau$ and zero otherwise. For any $\tau \geq 0$ such that $\beta_1(t,\tau)$ is wide-sense increasing, then flow 1 is guaranteed the (equivalent) service curve $\beta_1(t,\tau)$.

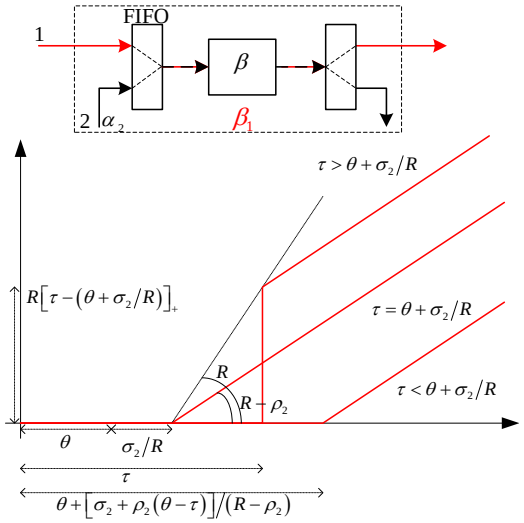

Figure 9: Equivalent service curve for flow 1 at a FIFO node.

The method known as Least Upper Delay Bound (LUDB) attempts to remove the cross flows by iteratively applying Theorem 2. This may or may not be possible, depending on the paths of the cross flows. Two cases are given, shown in Figure 10: in a so-called *"nested"* tandem, i.e., one where either any two flow paths are disjoint or one includes the other, it is possible to compute an equivalent end-to-end service curve for the tagged flow [20], by removing cross flows starting from the inmost ones. Otherwise, no end-to-end service curve can be computed: first, the tandem has to be *cut* into (possibly many) nested sub-tandems; then, bounds on the WCD of each sub-tandem must be computed and summed up to obtain a bound on the end-to-end WCD. Moreover, note that – by Theorem 2 – a flow is guaranteed an *infinity* of equivalent service curves, each one of which is obtained for a non-negative value of τ. For instance, Figure 9 shows some of them when $\alpha_2(t) = \gamma_{\sigma_2,\rho_2}$ and $\beta(t) = \beta_{\theta,R}$.

This means that, if Theorem 2 is applied iteratively n times to find an equivalent end-to-end service curve for the tagged flow, this will be a function of n non-negative parameters. Therefore, if one wants to compute a delay bound for the tagged flow through Eq. (2), the latter will itself be a (piecewise-linear) function of n parameters. The *tightest* delay bound is therefore computed by taking the *minimum* of that function, computed on all the values of the parameters, which entails solving a piecewise-linear programming (P-LP) problem. A tool called DEBORAH has been devised to solve the problem. It transforms the P-LP problem into a number of LPs, each one of which produces an upper bound on the WCD, solves all the LPs and takes the minimum solution (i.e., the least upper bound). The LUDB method, in general, does not compute the WCD. It does in sink-tree networks ([18]) and in some more special cases ([5]), but this is not always the case, even in simple nested tandems, as proved in [3]. This is due to two reasons: first, equivalent service curves are *pessimistic* constraints, i.e., given a CAF, they also allow CDFs which cannot be obtained in practice. (an example is reported in [3]). Second, [5] argues that, in non-nested tandems, the LUDB may be grossly overrated, due to the fact that cutting the tandem entails assuming separate, non compatible worst-case scenarios at each sub-tandem, hence introducing more pessimism. On the other hand, computing the LUDB is relatively easy. While the problem is still exponential with the tandem size, some intuitions described in [5] allow great speedups, so that the analysis of a 10-node tandem takes less than a second. Works [10, 15] discuss the advantages of including peak constraints (i.e., double-leaky-bucket arrival curves) in the model. While such a constraint can be freely assumed for the tagged flow, as shown in [10], it cannot be assumed for cross flows, which limits its usefulness in practice. In fact, if we assume cross-flows to be shaped by double leaky-buckets, if the overall peak rate of cross flows is larger than the node's rate, then Theorem 2 yields curves which are not wide-sense increasing, hence cannot be assumed to be equivalent service curves.

Other works in the recent past (e.g. [7, 25]) have focused on computing delay bounds for flows in feed-forward networks of blind multiplexing nodes, still using NC. *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 is always multiplexed at the lowest priority are compatible with this hypothesis. For blind-multiplexing networks the exact WCD has been computed using mathematical programming techniques in [7]. While the problem is NP-hard for general feed-forward networks, it can be solved with a single linear program of polynomial size for tandems. Even though the bounds computed therein are tight for a blind multiplexing scenario, they are not so under the FIFO property, and the gap between the WCD and these bounds can be shown to approach infinity as the node utilization approaches 100%. Assume in fact a simple scenario, where two flows share a single node: the tagged flow sends only one bit, and the cross flow has a non-null burst and a rate equal to the node rate. In a worst case, the node is always backlogged, hence – assuming that the tagged flow has the lowest priority – the bit of the tagged flow may remain in the buffer forever. This is not the

case, obviously, if the node is FIFO. All the above literature applies Network Calculus. A comprehensive survey on the recent advancements in NC can be found in [14].

A related stream of literature is instead based on a different method, called the *Trajectorial Approach* (TA, see, e.g., [22, 21, 1, 2]. In the latter, all the trajectories of a systems, i.e., the feasible interleavings of packets at the nodes, are considered, and a bound on the WCD is computed as a maximum among all the scenarios. A comparison between NC and TA was attempted in [1], where an AFDX network is employed as a case study. The results are that the TA gives better bounds most of the times (although, significantly, not always). On the other hand, [5] argues that such a comparison is not conclusive, due to the different underlying hypotheses. In fact, the TA relies on *sporadic task* arrival constraints, which are less general than concave arrival curves (i.e., some trajectories allowed under the latter are not allowed under the former). Furthermore, no study that we are aware of describes the computational cost and scalability of TA, and no publicly available tools exist that allow one to test these aspects.

6 Numerical Results

In this section we provide numerical results about the speed and accuracy of the WCD and the upper and lower bounds. An initial validation phase has been done using scenarios where the WCD can actually be computed using the LUDB method (see [18] and [5]). In all these cases, the MP optimum matches the analytical result, barring negligible numerical errors.

In order to compare our results against those of the only other comparable method we know of, i.e., DEBORAH [4], we analyze tandem networks, where arrival curves are single leaky-bucket ones, service curves are rate-latency. Experiments are carried out on a machine equipped with an Intel Core i7-3820 3.6GHz CPU, 16 Gb of RAM, Windows 7 64bit and CPLEX 12.5. We only measure the time spent by CPLEX, which also includes the time for reading the model and writing the solution. CPLEX is used with default options, a time limit of $2000s$, and a MILP optimality gap of 1% (i.e., if a computation ends by $2000s$, the WCD is within 101% of its result). Instances exceeding the time limit are discarded.

Scenarios are generated by specifying the number of nodes and a tandem type among the following: one-hop persistent, two-hop persistent (i.e., with cross-flows entering every node and spanning two hops, hence non-nested), source-tree, and random. The first three are shown in Figure 10. In random tandems, the path of the flows is selected at random (the tandem is however checked to be non-nested), and the overall number of flows F can be set as a percentage of their maximum possible number (i.e., $N \cdot (N-1)/2$ for N nodes, excluding flows traversing only one hop). Furthermore, the following variables can be set: flows bursts and rates, nodes latency, rate and utilization, instantiated by sampling a uniform distribution within a configurable range $[min, max]$. Scenarios with randomness are run 50 times with indepedent initial conditions.

The first group of results, shown in Figure 11, refer to computation times. They are obtained varying the number of nodes, with 50 replicas per scenario, under the following common settings:

- flow bursts: $\sigma_p \in U[100, 1000]\ \forall p$;
- flow rates: $\rho_p \in U[10, 100]\ \forall p$;
- node latencies: $\theta_j \in U[0, 1]\ \forall j$;
- node utilization: $U_j \in U[0.2, 1.0]\ \forall j$, meaning that the rate of node j is set to the sum of the rates of the flows traversing it, divided by U_j. The only exceptions are the rightmost graphs of Figures 11, 12, 13, where all the nodes have the same utilization.

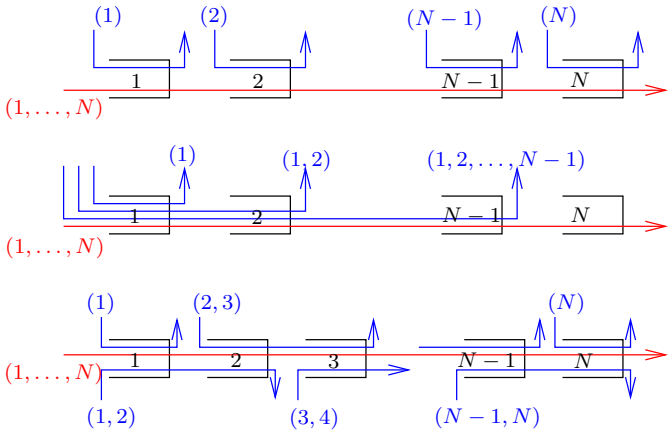

Figure 10: Two nested tandems, called *one-hop* (top) and *source-tree* (middle), and a *two-hop*, non-nested tandem (bottom).

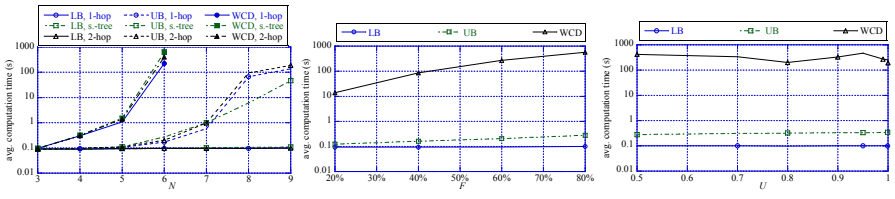

Figure 11: Computation times as a function of N for several topologies (left), and as a function of F (center) and U (right) for a random six-node tandem.

The results can be commented as follows: as already anticipated, computing the WCD is a lengthy task, whose complexity is exponential in the number of nodes. It takes 4-10 minutes to obtain the WCD for a tandem of six nodes. The

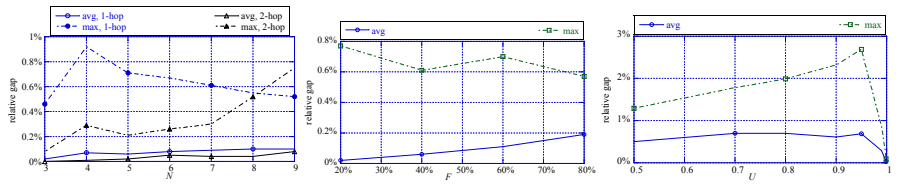

Figure 12: Gap $(V_{LP} - v_{LP})/v_{LP}$ as a function of N for two topologies (left), and as a function of F (center) and U (right) for a random six-node tandem.

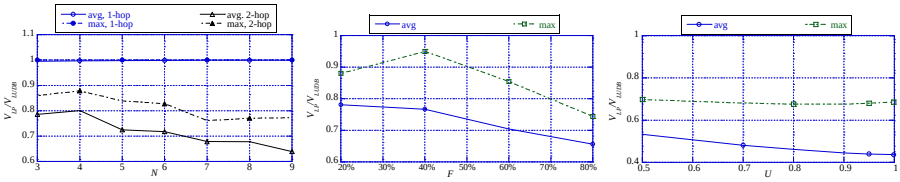

Figure 13: Gain V_{LP}/V_{LUDB} as a function of N for two topologies (left), and as a function of F (center) and U (right) for a random six-node tandem.

upper bound – being an LP – takes considerably less, and we can obtain it for tandems of up to ten nodes within similar times. The lower bound is instead very fast, being polynomial. We observe that these network sizes are normally sufficient for most purposes, the number of hops in real-time networks rarely exceeding the single figure. In all the above cases, DEBORAH computes the LUDB in less than $30ms$, i.e. much faster. Finally, the spread in the computation time (measured by the ratio of the standard deviation to the average value, not shown in the figures) increases as well with the number of nodes. 0.7% of WCDs for $N = 6$, and 10% of upper bounds for $N = 9$, could not be computed within $2000s$.

The number of flows also impacts computation times, although less than the number of nodes. The center graph of Figure 11 reports computation times for random six-node tandems, with a varying percentage of flows. The added computation time when doubling the number of flows is noticeable, but not overly so, especially in the bounds. The rightmost graph of Figure 11 shows the impact of node utilization on the computations, for a six-node tandem where the number of flows F is set to 80% and U_j is the same at each node. The figure shows that the computation times are almost unaffected by the utilization.

Two conclusive comments are in order regarding the computation time: first, these performance figures are obtained using a general-purpose solver such as CPLEX. We have in fact deliberately avoided any attempt at optimizing the solution process, this being outside the scope of the present paper. Devising optimized solution techniques to compute our results, e.g., exploiting the problem structure, is likely to buy us a considerable speedup. In fact, it already did for DEBORAH, itself heavily exploiting optimization, where a reduction of orders of magnitude in the solving time was achieved exactly by letting the structure

of the problem guide the solution process. Second, we now show that the MP upper bounds are often so good that computing the WCD is not even necessary in most cases. Consider Figure 12, which reports the gap between the upper and lower bounds, defined as $g = (V_{LP} - v_{LP})/v_{LP}$. The gap is identically null for source-tree tandems (hence it is not reported), and very small in all the other cases. Notably, its average is even smaller than the MILP optimality gap. This means that V_{LP} is a tighter estimate for the WCD than 101% of the MILP optimum in most cases, and it comes orders of magnitude faster. The average gap increases with the number of nodes and flows, and – interestingly – drops to zero when U approaches 100%: In Figure 12, right, a maximum 15% gap is achieved in one instance only with $U = 1$ (very likely due to numerical instability), whereas the gap is identically null in the other 49 ones. Although

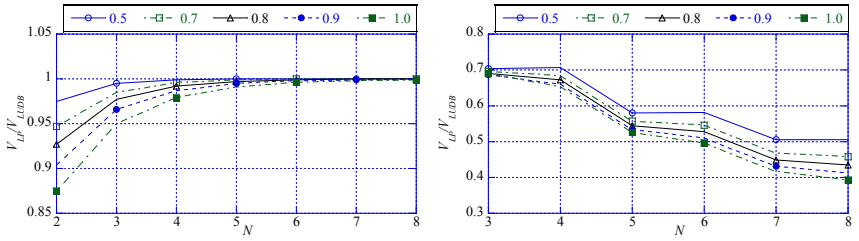

Figure 14: Gain V_{LP}/V_{LUDB} in one-hop (left) and two-hop (right) tandems with various utilizations.

V_{LP} is much cheaper to compute than the WCD, it is still in the hundreds of seconds or more for a 10-node tandem, whereas the LUDB is split-second in the same conditions. However, the accuracy of V_{LP} cannot be matched by the LUDB. Figure 13 reports the gain $o = V_{LP}/V_{LUDB}$ in the above-mentioned scenarios. The figures confirm an intuition which was already reported in [5], i.e., that the LUDB is accurate in nested tandems, and completely inaccurate in non-nested ones. In fact, for nested tandems, o is close to 1 (although always below 1, at least in our experiments), and identically equal to 1 in a source-tree tandem (hence not reported in the figure). On the other hand, with non-nested tandems, o can be as small as 40%, and generally decreases with the number of nodes, the number of flows, and the utilization. Figures 14 report o as a function of the number of nodes and utilization for two particular cases of symmetric tandems, namely a one-hop tandem (left) and two-hop tandem (right), with all flows having $\sigma = 1000$, $\rho = 100$, and all nodes having null latency. The figures confirm that the one-hop LUDB formula is asymptotically tight (see [5]), and that the overrating increases with the utilization. Interestingly, in the two-hop case the behavior of o is jaggy. This is because, in a non nested tandem, the LUDB is computed using cuts, and in this case the number of cuts increases at every odd node, hence the LUDB overrating increases accordingly.

7 Conclusions and future work

In this paper we have presented a method to compute Worst-case Delays (WCD) for flows in FIFO feed-forward networks, under the assumption that nodes exhibit piecewise-affine convex service curves, and flows are regulated by piecewise-affine concave arrival curves. This is the first work that achieves this objective, since previous works only managed to compute upper bounds on the WCD, under more restrictive hypotheses. Unlike the previous work, our method does not rely on equivalent service curves, which are definitely a source of pessimism: instead, it is based on Mathematical Programming. More specifically, it computes the WCD as the optimum of a mixed integer-linear problem (MILP), where binary (hence, integer) variables are required to enforce that cumulative functions be wide-sense increasing. The size of these MILPs grows exponentially with the number of nodes, which makes them impractical for large-scale problems: using a general-purpose solver, it takes minutes, or tens thereof, to analyze networks of six nodes. However, the MILP formulation naturally lends itself to computing upper and lower bounds, by respectively relaxing the constraints of the MILP, and reducing the admissible region by imposing further constraints. We have thus proposed a way to compute upper and lower bounds on the WCD, both requiring only one *linear* program: the one for the upper bound still has an exponential size, whereas the LP for the lower bound may have a polynomial (quadratic) size, e.g. in tandem or tree networks. The upper bound (which is, of course, the most interesting of the two) can be computed faster and/or for larger-scale networks, and it is very close to the WCD, the gap with the lower bound being below 2% in most cases. Furthermore, it is considerably more accurate than the bound computed through DEBORAH, the only comparable publicly available tool.

The work described in this paper can be extended: we believe that our problems (both the MILP for the WCD and the LP for the upper bound) can be solved optimally faster by using ad-hoc solving strategies, relying on the structure of the problems, instead of general-purpose solvers. Furthermore, it stands to reason that it should be possible to obtain the WCD either in a closed form or by defining a worst-case scenario algorithmically, possibly under more restrictive hypotheses. Investigating the above is part of the ongoing work.

8 Table of Symbols

We report the main symbols used throughout the paper in Table I.

9 Complete two-node example

We report in Table II the entire model for the two-node example of Section IV-B.

Notation	Meaning
$\alpha_p(t)$	arrival curve for the flow traversing path p
$\beta^j(t)$	service curve at node j
$\beta_1(t,\tau)$	equivalent service curve for flow 1
A, D	cumulative arrival/departure functions
F_p^j	cumulative function for flow p at the ingress of node j
$FIFO^j(t), SC^j(t)$	FIFO- and service-curve times of a time t which is observed at the output of node j
$\leq_b, \geq_b$	inequalities holding in the stated verse when binary variable b is equal to 1
$\mathcal{K}$	set of *known* inequalities
$\mathcal{T}_{in}^j, \mathcal{T}_{out}^j$	set of relevant times at the input/output of node j
$\mathcal{I}_{in}^j, \mathcal{I}_{out}^j$	set of time inequalities at the input/output of node j
$\mathcal{C}_T, \mathcal{C}_F$	set of time and function constraints

Table 1: Table of symbols

max	$t_1 - t_4$	
s.t.		
	Node 2	
$(Ord.)$	$t_3 \leq t_2 \leq t_1$	
(SC)	$F^{\epsilon}_{(2)} t_1 + F^{\epsilon}_{(1,2)} t_1 \geq F^2_{(2)} t_3 + F^2_{(1,2)} t_3 + \beta^2(t_1 - t_3)$	
$(FIFO)$	$F^{\epsilon}_{(p)} t_1 = F^2_{(p)} t_2$	$p \in \{(2),(1,2)\}$
$(Mono.)$	$F^2_{(p)} t_3 \leq F^2_{(p)} t_2$	$p \in \{(2),(1,2)\}$
$(Arr.)$	$F^2_{(2)} t_2 - F^2_{(2)} t_3 \leq \alpha_{(2)}(t_2 - t_3)$	
	Node 1	
$(Ord.)$	$t_2 \geq t_4 \geq t_5$	
	$t_3 \geq t_6 \geq t_7$	
	$t_4 \geq t_6$	
	$t_5 + (1-b) \cdot M \geq t_6$	
(SC)	$F^2_{(1,2)} t_i + F^2_{(1)} t_i \geq F^1_{(1,2)} t_j + F^1_{(1)} t_j + \beta^1(t_i - t_j)$	$(i,j) \in \{(2,5),(3,7)\}$
$(FIFO)$	$F^2_p t_i = F^1_p t_j$	$p \in \{(1),(1,2)\},\ (i,j) \in \{(2,4),(3,6)\}$
$(Mono.)$	$F^1_p t_k \geq F^1_p t_\ell$	$p \in \{(1),(1,2)\}, (k,\ell) \in \{(4,5),(4,6),(4,7),(5,7),(6,7)\}$
	$F^1_p t_5 + (1-b) \cdot M \geq F^1_p t_6$	$p \in \{(2),(1,2)\}$
	$F^1_p t_5 \leq b \cdot M + F^1_p t_6$	$p \in \{(2),(1,2)\}$
$(Arr.)$	$F^1_p t_k - F^1_p t_\ell \leq \alpha_p(t_k - t_\ell)$	$p \in \{(1),(1,2)\}, (k,\ell) \in \{(4,5),(4,6),(4,7),(5,7),(6,7)\}$
	$F^1_p t_5 - F^1_p t_6 \leq \alpha_p(t_5 - t_6) + (1-b) \cdot M$	$p \in \{(2),(1,2)\}$
	$F^1_p t_6 - F^1_p t_5 \leq \alpha_p(t_6 - t_5) + b \cdot M$	$p \in \{(2),(1,2)\}$
	Variables	
	$t_k \in \mathbb{R}_+$	$1 \leq k \leq 7$
	$F^j_p t_k \in \mathbb{R}_+$	$p \in \{(1),(2),(1,2)\},\ j \in \{1,2\},\ 1 \leq k \leq 7$
	$b \in \{0,1\}$	

Table 2: Complete MILP for the two-node example.

10 Computation of sets of inequalities $\mathcal{I}^j_{in}, \mathcal{I}^j_{out}$

The sets of inequalities $\mathcal{I}^j_{in}$ and $\mathcal{I}^j_{out}$ are built respectively according to Algorithms 1 and 2. We denote with $\mathcal{K}_{|\mathcal{S}}$ the restriction of $\mathcal{K}$ to pairs of times belonging to set $\mathcal{S}$. Algorithm 1 has two goals: first, to propagate the inherited

Algorithm 1: Computation of $\mathcal{I}^j_{in}$.

Data: $\mathcal{I}^j_{out}$, the ordering of the output times at node j; $\mathcal{K}$, the known time orders.
Result: $\mathcal{I}^j_{in}$, the ordering of the input times at node j.

begin
  $\mathcal{I}^j_{in} \leftarrow \mathcal{K}_{|\mathcal{T}^j_{in}}$;
  foreach $(t \geq_b t') \in \mathcal{I}^j_{out}$ **do**
    if $b \neq \emptyset$ **then**
      add $FIFO^j(t) \geq_b FIFO^j(t')$ and $SC^j(t) \geq_b SC^j(t')$ to $\mathcal{I}^j_{in}$;
      let b' be a new binary variable;
      add $FIFO^j(t) \geq_{b'} SC^j(t')$ to $\mathcal{I}^j_{in}$;
    let b'' be a new binary variable;
    add $SC^j(t) \geq_{b''} FIFO^j(t')$ to $\mathcal{I}^j_{in}$;

constraints from $\mathcal{I}^j_{out}$: if $t \geq t'$, then we necessarily have $FIFO^j(t) \geq FIFO^j(t')$ and $SC^j(t) \geq SC^j(t')$ and vice-versa, hence the same variable b is used. Second, to define the new constraints: as there is no knowledge concerning the value of b, we have to introduce new variables to order $FIFO^j(t/t')$ and $SC^j(t'/t)$. This way, every pair of times in $\mathcal{I}^j_{in}$ is now ordered. Note that if $t \geq t'$, then we know that $FIFO^j(t) \geq FIFO^j(t') \geq SC^j(t')$, so no new binary variable b' is needed. In Algorithm 2, we start from the order computed for the input

Algorithm 2: Computation of $\mathcal{I}^j_{out}$.

Data: $\mathcal{I}^k_{in}, \forall k \in succ(j)$, the ordering of the input times at successors of j; $\mathcal{K}$, the known time orders.
Result: $\mathcal{I}^j_{out}$, the ordering of the output times at node j.

begin
  $\mathcal{I}^j_{out} \leftarrow \mathcal{K}_{|\mathcal{T}^j_{out}} \cup \bigcup_{k \in succ(j)} \mathcal{I}^k_{in}$;
  for ℓ **from** *1* **to** $p-1$ **do**
    for m **from** $\ell+1$ **to** p **do**
      foreach $t \in \mathcal{I}^{k_\ell}_{in}$ **and** $t' \in \mathcal{I}^{k_m}_{in}$ **do**
        if $t \geq t' \notin \mathcal{K}$ **and** $t' \geq t \notin \mathcal{K}$ **then**
          let b be a new binary variable;
          add $t \geq_b t'$ to $\mathcal{I}^j_{out}$

times of the successors, as we consider the union of the times appearing in those

sets. Then, we have to set an order for the times appearing on different sets of successors. Whenever pairs of times are not in $\mathcal{K}$, we have to introduce a new binary variable.

Acknowledgements

The authors would like to thank Prof. Antonio Frangioni of the University of Pisa for providing useful suggestions on modeling the WCD problem.

References

[1] H. Bauer, J.-L. Scharbarg, and C. Fraboul. Improving the worst-case delay analysis of an AFDX network using an optimized trajectory approach. *IEEE Trans. on Industrial Informatics*, 6(4):521–523, 2010.

[2] H. Bauer, J.-L. Scharbarg, and C. Fraboul. Applying trajectory approach with static priority queuing for improving the use of available AFDX resources. *Real-Time Systems*, 48(1):101–133, 2012.

[3] L. Bisti, L. Lenzini, E. Mingozzi, and G. Stea. Estimating the worst-case delay in FIFO tandems using network calculus. In *proc. of Valuetools'08*, 2008.

[4] L. Bisti, L. Lenzini, E. Mingozzi, and G. Stea. Deborah: A tool for worst-case analysis of FIFO tandems. In *proc. of ISoLA'10, Special Track on Worst-case Traversal Time*, pages 152–168, 2010.

[5] L. Bisti, L. Lenzini, E. Mingozzi, and G. Stea. Numerical analysis of worst-case end-to-end delay bounds in FIFO tandem networks. *Real-Time Systems*, pages 527–569, 2012.

[6] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss. *An Architecture for Differentiated Services*. IETF, 1998.

[7] A. Bouillard, L. Jouhet, and E. Thierry. Tight performance bounds in the worst-case analysis of feed-forward networks. In *proc. of INFOCOM'10*, 2010.

[8] A. Bouillard and A. Junier. Worst-case delay bounds with fixed priorities using network calculus. In *proc. of Valuetools'11*, 2011.

[9] A. Bouillard and G. Stea. Exact worst-case delay for FIFO-multiplexing tandems. In *proc. of Valuetools'12*, 2012.

[10] M. Boyer. Half-modeling of shaping in FIFO net with network calculus. In *proc. of RTNS'12*, 2012.

[11] A. Charny and J.-Y. L. Boudec. Delay bounds in a network with aggregate scheduling. In *proc. of QoFIS'00*, pages 1–13, 2000.

[12] R. L. Cruz. A calculus for network delay, part I: Network elements in isolation. *IEEE Trans. on Information Theory*, 37(1):114–131, 1991.

[13] R. L. Cruz. A calculus for network delay, part II: Network analysis. *IEEE Trans. on Information Theory*, 37(1):132–141, 1991.

[14] M. Fidler. A survey of deterministic and stochastic service curve models in the network calculus. *IEEE Communications Surveys & Tutorials*, 12(1):59–86, 2010.

[15] F. Jafari, A. Jantsch, and Z. Lu. Worst-case delay analysis of variable bit-rate flows in network-on-chip with aggregate scheduling. In *proc. of DATE'12*, pages 538–541, 2012.

[16] A. Koubaa, M. Alves, and E. Tovar. Modeling and worst-case dimensioning of cluster-tree wireless sensor networks. In *proc. of IEEE RTSSS'06*, pages 412–421, 2006.

[17] J.-Y. Le Boudec and P. Thiran. *Network Calculus: A Theory of Deterministic Queuing Systems for the Internet*, volume LNCS 2050. Springer-Verlag, v. 4, may 10, 2004 edition, 2001.

[18] L. Lenzini, L. Martorini, E. Mingozzi, and G. Stea. Tight end-to-end per-flow delay bounds in FIFO multiplexing sink-tree networks. *Performance Evaluation*, 63(9-10):956–987, 2006.

[19] L. Lenzini, E. Mingozzi, and G. Stea. Delay bounds for FIFO aggregates: a case study. *Computer Communications*, 28(3):287–299, 2005.

[20] L. Lenzini, E. Mingozzi, and G. Stea. A methodology for computing end-to-end delay bounds in FIFO-multiplexing tandems. *Performance Evaluation*, 65(11-12):922–943, 2008.

[21] S. Martin and P. Minet. Schedulability analysis of flows scheduled with FIFO: application to the expedited forwarding class. In *proc. of IPDPS'06*, 2006.

[22] S. Martin, P. Minet, and L. George. Deterministic end-to-end guarantees for real-time applications in a diffserv-mpls domain. In *proc. of SERA*, pages 51–73, 2003.

[23] E. Rosen, A. Viswanathan, and R. Callon. *Multiprotocol Label Switching Architecture*. IETF, 2001.

[24] J. B. Schmitt and U. Roedig. Sensor network calculus: A framework for worst case analysis. In *proc. of DCOSS'05*, pages 141–154, 2005.

[25] J. B. Schmitt, F. A. Zdarsky, and M. Fidler. Delay bounds under arbitrary multiplexing: When network calculus leaves you in the lurch ... In *proc. of INFOCOM'08*, 2008.

[26] M. Schroeder, A. Birrell, M. Burrows, H. Murray, R. Needham, T. T. Rodeheffer, E. H. Satterthwaite, and C. P. Thacker. Autonet: a high-speed, self-configuring local area network using point-to-point links. *IEEE JSAC*, 9(8):1318–1335, 1991.

[27] T. Skeie, S. Johannessen, and O. Holmeide. Timeliness of real-time IP communication in switched industrial ethernet networks. *IEEE Trans. on Industrial Informatics*, 2:25–39, 2006.

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