Efficient and Accurate Handling of Periodic Flows in Time-Sensitive Networks

Seyed Mohammadhossein Tabatabaee, Marc Boyer, Jean-Yves Le Boudec, Jörn Migge

Efficient and Accurate Handling of Periodic Flows in Time-Sensitive Networks

Seyed Mohammadhossein Tabatabaee
EPFL
Lausanne, Switzerland
hossein.tabatabaee@epfl.ch

Marc Boyer
ONERA
Toulouse, France
marc.boyer@onera.fr

Jean-Yves Le Boudec
EPFL
Lausanne, Switzerland
jean-yves.leboudec@epfl.ch

Jörn Migge
RealTime-at-Work (RTaW)
Nancy, France
jorn.migge@realtimeatwork.com

***Abstract*—Total Flow Analysis (TFA) is a method for the worst-case analysis of time-sensitive networks. It uses service curve characterizations of the network nodes and arrival curves of flows at their sources; for tractability, the latter are often taken to be linear functions. For periodic flows, which are common in time-sensitive networks, linear arrival curves are known to provide less good bounds than ultimately pseudo-periodic (UPP) arrival curves, which exactly capture the periodic behaviours. However, in existing tools, applying TFA with many flows and UPP curves quickly becomes intractable because when aggregating several UPP curves, the pseudo-period of the aggregate might become extremely large. We propose a solution to this problem, called Finite-Horizon TFA. The method computes finite horizons over which arrival and service curves can be restricted without affecting the end-results of TFA. It can be applied to networks with cyclic dependencies. We numerically show that, while remaining computationally feasible, the method significantly improves the bounds obtained by TFA when using linear curves.**

I. Introduction

We are interested in delay bounds for the worst case, as is typical in the context of deterministic networking and time-sensitive networks. We assume that flows are grouped into classes, packets inside one class are processed first in first out (FIFO) and classes are isolated using schedulers. The number of bits that flows can generate is limited at sources by arrival curve constraints; this is necessary for the existence of a finite delay bound. Network calculus [1], [2] is a standard approach to find delay bounds. Specifically, network calculus abstracts, by means of a service curve, the service offered by a node to a class. Then, by combining the service curve of the node with the aggregate arrival curves of flows at this node, a bound on the worst-case delay at a node is obtained. The results equally apply to real-time systems, by mapping flow to task, packet to job, packet size to job-execution time, and service curve to "delivery curve" [3], [4].

Total Flow Analysis (TFA) [5], [6] obtains end-to-end delay bounds in FIFO networks and can be applied to per-class networks that are FIFO per class, where a service curve is known at every node and an arrival curve is known for every flow at the source. When the network is feed-forward, validity and correctness of TFA are shown with arrival curves and service curves of generic shapes. TFA is extended to networks with cyclic dependencies by FixPoint-TFA (FP-TFA) [7] and its variants such as SyncTFA [8], however with the restriction that arrival and service curves should be linear (i.e., token-bucket arrival curves and rate-latency service curves). Such a restriction often results in bounds that are pessimistic as they cannot accurately abstract the arrival model and the service model. For instance, for periodic flows, common in real-time and time-sensitive networks, a pseudo-periodic arrival curve, a stair function, exactly captures the periodic behaviour of the flow traffic (see Fig. 2). As another example, non-convex service curves for schedulers such as Deficit Round-Robin [9], Weighted Round-Robin [10], and Credit Base Shaper [11] are known to improve delay bounds compared to rate-latency ones, as they accurately capture the scheduler's behavior (see Fig. 2). For time-sensitive networks, as in the context of IEEE TSN and IETF Detnet, cyclic dependencies are linked to certain primary properties such as improving availability and decreasing reconfiguration effort, hence are important and cannot be ignored. However, as of today, methods that analyze networks with cyclic dependencies are restricted to only linear curves, hence are potentially pessimistic.

Our first step is to generalize the theory of FP-TFA to arrival and service curves of generic shapes, which provides tighter bounds for networks with cyclic dependencies. Specifically, we present a new version of FP-TFA, called Generic FP-TFA (*GFP-TFA*), and prove its validity and correctness for arrival and service curves with generic shapes (Theorem 3). GFP-TFA is not a practical algorithm when there are many periodic sources, as we explain next, however, it serves as a theoretical reference; furthermore, it is used as a building block in the main algorithm later presented in this paper.

Tools such as RTaW [12], Nancy [13], DiscoDNC [14], etc. use infinite precision arithmetic (with rational numbers) and implement Ultimately Pseudo-Periodic (UPP) curves; UPP curves have a transient part at the beginning, followed by a periodic pattern. UPP curves are of interest in practice as they have a finite representation, and moreover, they capture periodic behaviors (see Fig. 1a). For instance, in the case of periodic flows, UPP curves can accurately describe their periodic arrival curve, and in the case of service curves, UPP curves can describe non-convexity. However, applying GFP-TFA with many flows and UPP curves quickly becomes intractable. This is because when aggregating several UPP curves, the pseudo-period of the aggregate function might become extremely large; moreover, the required memory to

store the aggregate function might explode as the number of segments required to describe the aggregate function quickly grows (see Fig. 3). An example where this issue occurs is the avionic onboard communication system analyzed in [15]. More industrial examples can be found in Section VII.

An attempt to overcome this issue consists in replacing periods by smaller values such that the hyper-periods remain small, e.g., when aggregating three UPP curves with pseudo-periods equal to 3, 4, and 8, the hyper-period becomes 24; but, the pseudo-period 3 can be safely replaced by 2 thus the hyper-period becomes 8. However, this increases the load, hence the bounds, and moreover, it is not robust: It should be reapplied whenever a change happens, e.g., a period changes or a new periodic flow is added. Also, if a network is highly loaded, tweaking periods might violate local stability and the network analysis fails. This is why, for tractability, TFA is generally applied with linear arrival curves. In summary, on one hand, we have UPP curves that provide good bounds but applying GFP-TFA with many periodic flows and UPP curves might be practically intractable; on the other hand, we have linear curves, which are very tractable but provide less good bounds. Authors in [16] show that the network-calculus delay-computation (i.e., horizontal deviation) only depends on a finite part at the beginning of the arrival curve and the service curve and not the complete curves. This motivates us to find a middle point, namely, curves that follow original, UPP curves up to a finite horizon, and beyond that, follow simpler, linear curves. An Ultimately Affine (UA) curve can exactly capture this: a UA curve has a transient part at the beginning, followed by a linear curve (see Fig. 1b); working with UA curves mitigates the description complexity of UPP curves for an aggregate curve (see Fig. 3). Moreover, as UA curves are a subset of UPP curves, one can use a UPP implementation to handle UA curves. The main problem is now how to carefully construct such UA curves, from original, UPP curves and their linear upper/lower bounds, such that the end-results are not affected, i.e., as good as those obtained with original, UPP curves.

In order to solve this problem, we propose a second new version of TFA, called Finite-Horizon TFA (*FH-TFA*), which can be viewed as an efficient, practical replacement for GFP-TFA, and hence can be applied to networks with cyclic dependencies. The method computes sufficient finite horizons for every UPP arrival and service curve by adopting the compact domains of [16]. Note that the results about compact domains in [16] that are presented for a single node do not directly apply to TFA or FP-TFA; indeed, in a network analysis, arrival curves of flows increase as flows go deeper into the network, hence the compact domains required for delay computations increase as well. To address this, FH-TFA first applies FP-TFA using linear curves; this is very fast and provides enough information to compute sufficient finite horizons. Next, it constructs UA curves where the duration of the transient part is set to the computed sufficient finite horizons (see Fig. 4). Last, it applies GFP-TFA with these UA curves; the complexity is small due to the replacement of UPP curves by UA curves. In the common case where service curves are super-additive, we prove that FH-TFA produces the very same bounds as the intractable GFP-TFA (Theorem 4). Since FH-TFA is considerably less complex, this provides a tractable, efficient solution to the analysis of networks with UPP curves.

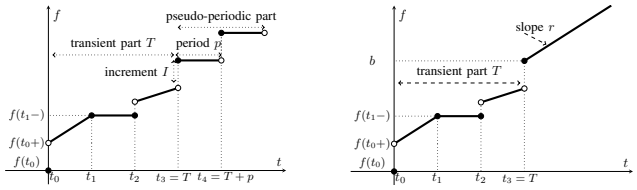
The contributions of this paper are as follows:

- We develop and validate FH-TFA, an algorithm that provides delay bounds for deterministic networks with generic topology, and generic arrival and service curves that are implemented as UPP or UA curves (Theorem 4). In contrast, for networks with cyclic dependencies, existing versions of TFA are limited to linear arrival and service curves, which may affect the quality of the delay bounds.
- We develop and validate GFP-TFA, an algorithm that generalizes the theory of existing versions of TFA (FP-TFA) to arrival curves and service curves of generic shapes, and provides tighter bounds for networks with generic shapes (Theorem 3). GFP-TFA is used to derive a building block of FH-TFA and to establish the validity of FH-TFA. GFP-TFA is of independent interest, but in practice applying it with many periodic sources with different periods might be intractable. In the common case where the service curves are super-additive, FH-TFA produces the very same bounds as the GFP-TFA (item (2) of Theorem 4) but at considerably less complexity.
- FH-TFA always provides valid delay bounds that are guaranteed to be less than or equal to those obtained by FP-TFA (item (1) of Theorem 4).
- FH-TFA is thus the only known method that obtains formally proven delay bounds with TFA, and can handle many periodic sources with different periods.

We give a numerical application to real, industrial cases provided by industrial partners of RTaW, a leading company in Ethernet TSN design, performance evaluation and automated configuration tools; these examples are anonymized and slightly changed. We observe the following:

- GFP-TFA with UPP curves cannot be applied to any of the examples we tested, as it produces a memory size error.
- In contrast, FH-TFA with UPP curves applies and remains tractable in all examples we tested, even in a very large, industrial network with cyclic dependencies and many periodic flows.
- Bounds obtained with FH-TFA and UPP curves (recall that they are the same as would be obtained with GFP-TFA) are considerably less than those obtained with linear approximations of the UPP curves.

The rest of the paper is organized as follows. Section II gives the necessary background and state-of-the-art. Section III describes the problem definition, the system model, the network under study, and the resulting graph. Section IV describes GFP-TFA, a theoretical solution to the problem at hand, and proves its validity. Section V describes FH-TFA, a practical



(a) UPP curve (b) UA curve

Fig. 1: Left: Function f, a UPP curve with a representation $([s_1, s_2, s_3, s_4], T, p, I)$ where $[s_1, s_2, s_3, s_4]$ are the 4 affine segments in $[0,\ T+p)$, T is the rank (size of the transient part), p is the pseudo-period, and I is the increment. Values outside this interval can be computed on demand by $\forall t > T,\ f(t + kp) = f(t) + kI$. Right: Function f, a UA curve with a representation $([s_1, s_2, s_3], T, p, I)$ where $[s_1, s_2, s_3]$ are the 3 affine segments in $[0,\ T)$, T is the rank, b and r are the burst and the slope of the linear part with $\forall t \geq T,\ f(t) = b + r(t - T)$.

solution to the problem at hand, and proves its validity. Section VI contains proofs of theorems. Section VII applies FH-TFA to some industrial networks, and gives the obtained delay bounds and run-times. Section VIII concludes the paper.

II. Background and State-of-the-Art

A. Family of Functions and Operators

In this section, we define UPP and UA functions, and explain how to represent such functions with a finite amount of information. We also provide some background on network calculus and operators used in the paper.

1) UPP and UA Curves: We follow the terminology in [17, Definition 1]. Let $\mathscr{F}$ denote the set of piece-wise linear and wide-sense increasing functions $f : \mathbb{Q}^+ \mapsto \mathbb{Q}^+ \cup \{+\infty\}$ where $\mathbb{Q}^+$ is the set of non-negative rational numbers. For $f \in \mathscr{F}$:

- f is *Ultimately Affine (UA)* if there exist $T, r, b \in \mathbb{Q}^+$ such that for all $t \geq T,\ f(t) = r(t - T) + b$; T is called a rank of function f, and the smallest possible value for T is the rank of the function; r and b are called the rate and the burst of the linear part (see Fig. 1b).
- f is *Ultimately Pseudo-Periodic (UPP)* if there exist $T, I \in \mathbb{Q}^+$ and $p \in \mathbb{Q}^+ \setminus \{0\}$ such that $\forall t > T$ and every non-negative integer k, $f(t + kp) = f(t) + kI$; p is called a pseudo-period and I is called an increment (see Fig. 1a).

The tuple (S, T, p, I) is a finite representation for a UPP function f: S represents values of f in the interval $[0, T + p)$, then values of f beyond this interval can be computed using S, pseudo-period p, and increment I. S is defined as follows: $S = [s_1, \ldots, s_k]$ is a list of affine segments where for $i \in [0, k],\ s_i = (t_i, t_{i+1}, f(t_i), f(t_i+), f(t_{i+1}-))$ such that $\forall t \in]t_i, t_{i+1}[,\ f(t)$ is the affine function that connects points $(t_i, f(t_i+))$ and $(t_{i+1}, f(t_{i+1}-))$, with $f(t+) = \lim_{\epsilon \to 0} f(t + \epsilon)$ and $f(t-) = \lim_{\epsilon \to 0} f(t - \epsilon)$. We require that (1) $t_1 = 0$, (2) there exists i_0 where $t_{i_0} = T$, and (3) $t_k < T + p$ and $t_{k+1} = T + p$. We assume that S is the minimal set, i.e., at each t_i, there is either a discontinuity or a change of slope (see Fig. 1a).

The tuple (S, T, r, b) is a finite representation for a UA function f; S is defined in a similar manner as UPP functions with $t_{k+1} = T$, and $\forall t \geq T,\ f(t) = b + r(t - T)$ (see Fig. 1b).

It is shown that UPP (resp. UA) functions are closed under addition, subtraction, min-plus convolution, min-plus deconvolution (see Section II-A2 for definitions), minimum and maximum of two functions [17]. Moreover, such operations are automated in tools such as RealTime-at-Work (RTaW) [12], Nancy [13], DiscoDNC [14], and etc. [17]–[19]; these interpreters provides efficient implementations of min-plus convolution, min-plus deconvolution, horizontal deviation, and a maximum and minimum of functions. All computations use infinite precision arithmetic (with rational numbers), and functions are represented as UPP or UA functions.

2) Network Calculus Background: We say that a flow has $\alpha \in \mathscr{F}$ as *arrival curve* if the number of bits generated by this flow for any $t' \leq t$ is upper bounded by $\alpha(t - t')$. An arrival curve α can always be assumed to be sub-additive, i.e., to satisfy $\alpha(t' + t) \leq \alpha(t') + \alpha(t)$ for all t, t', as otherwise it can be replaced by its sub-additive closure [20]. A periodic flow that sends up to b bits every p time units has, as arrival curve, the stair function, defined by $\nu_{p,b}(t) = b \left\lceil \frac{t}{p} \right\rceil$; it is UPP. Another frequently used arrival curve is the token-bucket function $\alpha = \gamma_{r,b}$, with rate r and burst b, defined by $\gamma_{r,b}(t) = rt + b$ for $t > 0$ and $\gamma_{r,b}(t) = 0$ for $t = 0$; it is UA (see Fig. 2). Both of these arrival curves are sub-additive.

Consider a system and a flow through the system. We say that a system offers a *strict service curve* $\beta \in \mathscr{F}$ to the flow if the number of bits of the flow output by the system in any *backlogged* interval $(t', t]$ is at least $\beta(t - t')$. A strict service curve is a special case of service curves; the exact definition of a non-strict service curve (known as minimum or min-plus minimal service curve) can be found in [2, Section 5.2]. A strict service curve β can be always assumed to be super-additive (i.e., to satisfy $\beta(t' + t) \geq \beta(t') + \beta(t)$ for all t, t'), otherwise, it can be replaced by its super-additive closure [2]. Service curves of schedulers such as Deficit Round-Robin [9], Weighted Round-Robin, and Interleaved Weighted Round-Robin [10] are strict hence super-additive. Non-strict service curves are often super-additive but not always. Service curves of Credit-Based Shapers [11] and Non-Preemptive Static Priority [2] are not strict, however, it can be shown that in the common cases they are super-additive. A frequently used service curve is the rate-latency function $\beta_{c,L} \in \mathscr{F}$, with rate c and latency L, defined by $\beta_{c,L}(t) = c[t - L]^+$, where we use the notation $[x]^+ = \max\{x, 0\}$ (see Fig. 2); it is UA and super-additive.

Assume that a flow, constrained by an arrival curve α, traverses a FIFO system that offers a service curve β. The delay of the flow is upper bounded by the horizontal deviation defined by $\text{hDev}(\alpha, \beta) = \sup_{t \geq 0}\{\inf\{d \geq 0 | \alpha(t) \leq \beta(t + d)\}\}$. The backlog for the flow is upper-bounded by the vertical deviation defined by $\text{vDev}(\alpha, \beta) = \sup_{t \geq 0}\{\alpha(t) - \beta(t)\}$.

For $f, g \in \mathscr{F}$, the min-plus convolution is defined by $(f \otimes g)(t) = \inf_{0 \leq t' \leq t}\{f(t - t') + g(t')\}$ and the min-plus deconvolution by $(f \oslash g)(t) = \sup_{t' \geq 0}\{f(t + t') - g(t')\}$ [1], [2], [21]. The pure delay function with parameter d is defined by $\delta_d(t) = 0$ for $t \leq d$ and $\delta_d(t) = \infty$ for $t > d$. We use

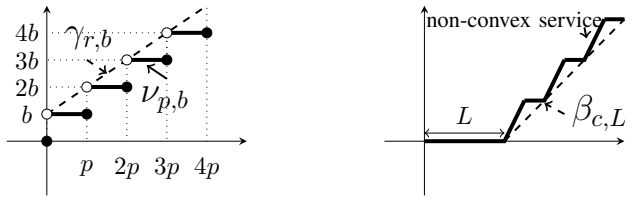


Fig. 2: Left: the stair function $\nu_{b,p} \in \mathscr{F}$ defined for $t \geq 0$ by $\nu_{b,p}(t) = b\left\lceil \frac{t}{p} \right\rceil$ and token-bucket function $\gamma_{r,b} \in \mathscr{F}$ defined for $t > 0$ by $\gamma_{r,b}(t) = b + rt$ and for $t = 0$ by $\gamma_{r,b}(0) = 0$ (in the figure, we have $r = \frac{b}{p}$). Right: a non-convex service curve and a rate-latency $\beta_{c,L} \in \mathscr{F}$ that lower bounds it with $\beta_{c,L}(t) = \max(0, c(t-L))$.

$(f \oslash \delta_d)(t) = f(t+d)$ (left-shift).

B. Total Flow Analysis (TFA)

Total Flow Analysis (TFA) [5]–[8] is a method for obtaining worst-case delay and backlog bounds in a FIFO network. In a network where a service curve is known at every node and an arrival curve for every flow is known at the source, one run of TFA returns a valid delay bound at every node and propagated burstiness for flows. Although TFA is simple and modular, it takes into account the effect of packetizer and line-shaping. When the graph induced by flows is feed-forward (i.e, cycle-free), each node is visited in the topological order, whereby a delay bound and output burstiness bounds of flows are computed; output burstiness of flows are used as an input by the following nodes. If the graph induced by flows has cyclic dependencies, a topological order cannot be defined; instead an iterative method is used and a fixpoint is computed [7], [8]. The method in [7], called FP-TFA, requires to first make some artificial cuts in the induced graph in order to create a feed-forward network. It then computes estimated burstiness of flows at cuts and iterates. It is shown that, if the iteration converges, the obtained fixpoint is a valid bound on the burstiness of flows at cuts, and the network is stable. Other versions of TFA that do not make cuts are presented in [8], where it is shown that they are equivalent to FP-TFA, i.e., they provide the same bounds and stability regions [8]. For networks with cyclic dependencies, all versions of TFA assume only token-bucket arrival curves and rate-latency service curves, and the validity of the results are shown with these assumptions. In Section IV, we provide a new version of FP-TFA, called Generic FP-TFA (*GFP-TFA*), that can be applied with any arrival curves and service curves of generic shapes, including UPP and UA ones. When arrival curve are token-bucket and service curves are rate-latency, GFP-TFA is essentially the same as FP-TFA.

C. Compact Domains for Delay Computation

It has been observed in [15] that, for some systems, the computation of the delay bounds does not require to handle the full function (i.e., all values of the function for all time t in $[0, +\infty)$), but only its values on a finite prefix domain. However, the theory in [15] lacks a proof that computation in such compact domains does not affect the accuracy of the end-results. The challenge consists in computing in each node a value h such that the computation on the compact domain $[0, h]$ is sufficient to get accurate result on this node but also on the next ones along the flow path. The intuition is the following: Consider a flow that traverses two nodes in sequence. Assume that the first node (resp. the second node) requires that the arrival curve of the flow at the input of the node is accurate in $[0, h]$ (resp. $[0, h']$). As the arrival curve of the flow increases along the path and some information is lost at propagation, arrival curve of the flow at the input of the first node should be accurate for some $[0, h'']$ where $h'' > \max(h, h')$ is large enough.

Authors in [22] derive compact domains where they prove that the accuracy of the end-results are not affected; their results and proofs only hold for input/output relations in acyclic network of the Greedy-Processing Component (GPC), used in Real-Time Calculus (RTC).

Authors in [16] derive such compact domains, in a more general context. They show that, at a single node where an arrival curve and a service curve are known, network calculus operations, including delay computations, can be restricted to finite domains without affecting the end-results. Here, we rewrite one of their findings that we use in the paper, using our notation:

Theorem 1 (Theorem 4 of [16])**.** *Consider a flow constrained by an arrival curve α that traverse a node that offers a super-additive service curve β. Let α' and α'' be a lower bound and upper bound, respectively, for α, i.e., $\alpha' \leq \alpha \leq \alpha''$. Also, let β' and β'' be an upper bound and lower bound, respectively, for β, i.e., $\beta'' \leq \beta \leq \beta'$. Define*

$$h^{\alpha} \stackrel{def}{=} \max(u, v) \text{ and } h^{\beta} \stackrel{def}{=} \max(u + D'', v) \quad (1)$$

with

$$D' = hDev(\alpha', \beta') \quad (2)$$

$$B' = vDev(\alpha', \beta') \quad (3)$$

$$u = \sup_{t \geq 0}\{\alpha''(t) \geq \beta''(t + D')\} \quad (4)$$

$$v = \sup_{t \geq 0}\{\alpha''(t) \geq \beta''(t) + B'\} \quad (5)$$

$$D'' = hDev(\alpha'', \beta'') \quad (6)$$

Then, the horizontal deviation $hDev(\alpha, \beta)$ and the vertical deviation $vDev(\alpha, \beta)$ depend only on the values of $\alpha(t)$ for $t \in [0, h^{\alpha}]$ and $\beta(t)$ for $t \in [0, h^{\beta}]$ (see Section II-A2 for definitions of super-additive service curve, hDev and vDev.)

Note that in the above, α'' and β'' are valid, safe arrival curve and service curve, respectively; however, α' and β' are unsafe arrival curve and service curve, respectively. Note that the method requires that service curves are super-additive (see Section II-A2). Authors in [16] also find such compact domains for min-plus deconvolution, and explain how to integrate such compact domains with Pay-Burst-Only-Once (PBOO) [14] and Pay-Multiplexing-Only-Once (PMOO) [14] principles in sink-tree networks with arbitrary multiplexing.

TABLE I: Notation List

f	A flow
E^{cut}	Cutset: removing E^{cut} creates a feed-forward graph
$\mathcal{G} = (\mathcal{N}, \mathcal{E})$	The graph induced by flows of class
$\text{path}(f)$	The sequence of output ports in the path of flow f
$\mathcal{N}, n$	The set of all output ports, an output port
$\mathcal{E}, e$	The set of edges, an edge
$\text{In}(n)$	The set of edges of that are incidents at node n
$\text{flows}(n)$	The set of flows at output port n
α_n	Aggregate arrival curve of all flows at the input of node n
α_n^{fresh}	Aggregate arrival curve of all fresh flows at the input of node n
$\alpha_n^{\text{transit}}$	Aggregate arrival curve of all transit flows at the input of node n
$\alpha_{f,n}$	An arrival curve for flow f at the input of node n
α_f^0	An arrival curve for flow f at the source
β_n	A service curve offered to node n
α^0	Collection of arrival curves of all flows at the source
d	Collection of delay bound d_n for every node n
τ^{cut}	Collection of delay-jitter bounds $\tau_{f,n}$ for every flow f at cuts
τ	Collection of delay-jitter bounds $\tau_{f,n}$ for every flow f at every node n in its path
β	Collection of service curves of all nodes
d_n	Delay bound at node n
$\tau_{f,n}$	Delay-jitter bound for flow f from the source to the input of node n
h	Sufficient horizon
$l_f^{\max}$	Maximum packet size for flow f
$l_f^{\min}$	Minimum packet size for flow f
Δ	Minimum resolution of times, e.g., 1 nanosecond
c_n	Transmission rate of the line fed by output port n
δ_d	Pure delay function with $\delta_d(t) = 0$ for $t \leq d$ and $\delta_d(t) = \infty$ for $t > d$
$\beta_{c,L}$	Rate-latency function with $\beta_{c,L}(t) = \max(0, c(t-L))$
$\mathbb{Q}^+$	Set of non-negative rational numbers
$\nu_{p,b}$	Stair function with $\nu_{p,b}(t) = b\left\lceil \frac{t}{p} \right\rceil$
$\gamma_{r,b}$	Token-bucket function with $\gamma_{r,b}(0) = 0$ and $\gamma_{r,b}(t) = rt + b$ for $t > 0$
hDev	Horizontal deviation $\text{hDev}(\alpha, \beta) = \sup_{t \geq 0}\{\inf\{d \geq 0 \mid \alpha(t) \leq \beta(t+d)\}\}$
$\otimes$	Min-plus convolution $(f \otimes g)(t) = \inf_{0 \leq s \leq t}\{f(t-s) + g(s)\}$
$\oslash$	Min-plus deconvolution $(f \oslash g)(t) = \sup_{s \geq 0}\{f(t+s) - g(s)\}$
vDev	Vertical deviation $\text{vDev}(\alpha, \beta) = \sup_{t>0}\{\alpha(t) - \beta(t)\}$

Later, authors in [23], generalize the work of [22], using the findings of [16], to be independent of GPC's operational semantics. Lastly, authors in [2, Prop. 5.13] provide looser bounds than those of Theorem 1.

III. System Model

We consider a packet-switched network. We assume that flows are grouped into static classes, and packets inside a class are processed First-In-First-Out (FIFO). Every device represents switches or routers and consists of input ports, output ports, and a switching fabric. Each packet enters a device via an input port and is stored in a packetizer. A packetizer releases a packet only when the entire packet is received. Then, the packet goes through a switching fabric. Then, the packet, based on its class, is either queued in a FIFO-per-class queue or exits the network via a terminal port.

In the rest of the paper we focus on one class of interest. We assume that the service offered to the aggregate of all flows that use some output port, say n, from exit of the packetizer (on an input port) to the transmission line fed by output port n can be modelled with a service curve β_n^{UPP}, where β_n^{UPP} is a UPP function (see Section II-A). Let c_n denote the transmission rate of the line fed by output port n. Each flow f of the class of interest is constrained at source by an arrival curve $\alpha_f^{0,\text{UPP}}$, where $\alpha_f^{0,\text{UPP}}$ is a UPP function (see Section II-A). In the case of periodic flows, arrival curves are stair function (see Section II-A2). We assume that $\alpha_f^{0,\text{UPP}}(0+) \geq l^{\max}$, where $f(t+) = \lim_{\epsilon \leftarrow 0} f(t+\epsilon)$. Also, flows are statically assigned to a path, and let $\text{path}(f)$ denote a sequence of nodes in the path of flow f. Let $\text{flows}(f)$ denote set of flows at node n.

We assume that the network is locally stable, i.e., the aggregate long-term arrival rate to each output port is strictly less than the long-term service rate.

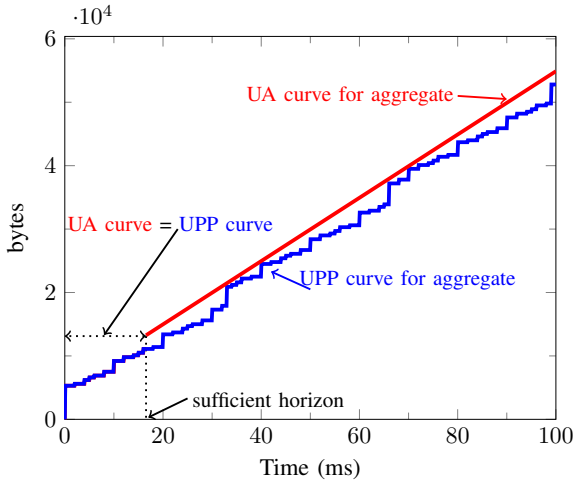


Fig. 3: We consider a single node that offers a rate-latency service curve $\beta_{c,L}$ with $c = 1$Gb/s and $L = 16\mu$s. We assume 6 fresh, periodic flows that are constrained by stair function $\nu_{p,b}$ with $p \in \{2, 4, 5, 10, 33, 100\}$ms and $b \in \{3, 3, 3, 10, 30, 3\} * 100$bytes. The aggregated arrival curve has the pseudo-period equal to 3300 and moreover, 2020 segments are required to represent it; here we only plot values in $[0, 100]$. Whereas, only 13 segments are required to represent the UA curve computed using our method. The sufficient horizon in this example is 16.37.

The graph $\mathcal{G} = (\mathcal{N}, \mathcal{E})$ induced by flows is the directed graph defined as follows: 1) Let $\mathcal{N}$ denote the subset of all non-terminal output ports used by at least one flow. 2) The directed edge $e = (n, n') \in \mathcal{E}$ exists if there is at least one flow that traverses n and n' in this order. We say that $\mathcal{G}$ has a cyclic dependency if it contains at least one cycle. Let $E^{\text{cut}} \subseteq \mathcal{E}$ be a cut such that artificially removing the edges in E^{cut} creates a feed-forward graph. Such cuts can be obtained by any traversal graph algorithm [24]. Without loss of generality, output ports are labeled in a topological order of the cut graph, starting from output ports at edges. Such topological orders exists as the cut network is feed-forward.

Problem Statement: The first problem is to provide, and prove the validity of, a new version of TFA that handles arrival and service curves of generic shapes in networks with generic topologies. The second problem is to apply the concept of sufficient horizon and obtain a new version of TFA for generic topologies that remains tractable with many UPP curves.

IV. GFP-TFA: A New Version of FP-TFA That Handles Arrival Curves and Service Curves of Generic Shapes

In this section, we present GFP-TFA, an adaptation of FP-TFA, that can be applied with arrival curves and service curves of generic shapes. GFP-TFA is an algorithm that exactly solves the problem (see Section III), but, GFP-TFA has high computational complexity and in practice, applying GFP-TFA with many UPP curves might be intractable. However, it serves as a theoretical reference, and it is used as a building block in the main algorithm, FH-TFA, presented in Section V.

FP-TFA assumes that arrival curves are token-bucket, then only the burst of arrival curves increases along the path, called propagated burstiness, and the rate remains unchanged; thus one of the variables that FP-TFA iterates on is the propagated burstiness. However, for arrival curves of generic shapes, propagated burstiness is not defined. To address this, in GFP-TFA, we replace propagated burstiness by delay-jitter bound from the source to the point of interest (variable τ): This is because an arrival curve for a flow at the point of interest is the one at the source, shifted to the left by a delay-jitter bound from the source to the point of interest; unlike propagated burstiness, this result holds for arrival curves of the generic shapes, thus GFP-TFA can be applied with any types of arrival curves. Also, FP-TFA uses a result on the effect of packetizer that is only expressed for token-bucket arrival curves, and needs to be adapted for arrival curves of generic shapes. We do this adaption in Theorem 2. The transformation of FP-TFA into GFP-TFA is otherwise straightforward, but for the sake of completeness, we describe GFP-TFA in details. The validity of GFP-TFA, which is less straightforward, is given in Theorem 3.

Recall that, as explained in Section II-B, FP-TFA first cuts the network and analyzes the resulting cut network, which is feed-forward, and iterates on propagated burstiness at cuts until a fix-point is reached. We follow the same structure, with some adaptation.

A. FF-TFA: TFA for Feed-Forward Networks

FF-TFA is a building block of GFP-TFA. It applies to the feed-forward network obtained after removing a cutset and is described in Algorithm 1. It takes as input the collection α^0 of arrival curves of all flows at the sources, the collection β of service curves of all nodes, the cutset E^{cut}, and a collection τ^{cut} of delay-jitter bounds from source to cut for every flow that is cut. It outputs a collection d of per-node delay bounds and a collection τ of delay-jitter bounds for every flow from its source to each node in its path.

The delay-jitter bound for every flow f at every node in its path is initialized to zero (line 1); for a flow at a cut, it is initialized to the corresponding value in τ^{cut} (lines 2-4). Then, nodes are visited in the topological order of the cut network; an aggregate arrival curve at the input of node n is computed (lines 5-6) by using the function aggregateArrivalCurve$_n$ defined at line 14. This function implements the effect of line shaping (line 19) and packetizer (line 21). Line shaping [25], [26] addresses the fact that when some flows are known to arrive from the same link (i.e., carried by the same edge), a better arrival curve can be computed for the aggregate of flows; specifically, for an edge e, α_e, an aggregate arrival curve for flows carried by edge e, can be replaced by $\alpha_e \otimes \gamma_{c_e,0}$ where c_e is the maximum link speed of the edge e. For the packetizer (see Section III), [7] studies the effect of packetizer when the aggregate flow is constrained by a token-bucket arrival; here we present a minor adaptation of [7, Theorem 1] that applies to arrival curves of generic shapes.

Theorem 2 (Output Arrival Curve at the Output of Packetizer)**.** *Consider a packetizer that is placed on a transmission line with a fixed rate c, and assume that it serves an aggregate flow, constrained by an arrival curve α; also, let l^{max} be the maximum packet size of the aggregate flow. Then, $\alpha \oslash \delta_{\frac{l^{max}}{c}}$, is an arrival curve for the aggregate flow at the output of the packetizer.*

The proof is in Section VI. Note that $\alpha \oslash \delta_{\frac{l^{\max}}{c}}$ is equal to α shifted to left by $\frac{l^{\max}}{c}$, i.e., $(\alpha \oslash \delta_{\frac{l^{\max}}{c}})(t) = \alpha(t + \frac{l^{\max}}{c})$.

Combining α_n, an arrival curve for the aggregate of flows at the input of node n with β_n, the service curve offered by node n, the improved network calculus delay bound is computed as in [27] (line 8): When an output port is followed by a transmission line, authors in [27, Theorem 5] find delay bounds that improve on the classical network calculus result (which is equal to horizontal deviation between the arrival and the service curve, i.e., hDev (α_n, β_n)); this is implemented in line 8. Then, the delay-jitter of node n is added for flows at successors of node n (lines 9-11); note that a delay-jitter at node n is obtained by the subtraction of the worst-case and best-case delay bound at a node, i.e., subtraction of d_n and the transmission time of a packet of minimum size.

B. GFP-TFA

GFP-TFA is described in Algorithm 2: It takes as input α^0, a collection of arrival curves for each flow at the source, β, a collection of service curves offered by each node, and a cutset E^{cut} such that the cut network is feed-forward. In lines 1-7, it first computes $\bar{\tau}^{\text{cut}}$, a collection of valid delay-jitter bounds from the sources to the cuts for every flow at cuts. It initializes delay-jitter bounds from the source to the cut to zero for every flow at the cut (line 2). It then iteratively calls FF-TFA, described in Algorithm 1, to compute new values for delay-jitter bounds of flows from source to cut. Note that since the initial values of $\bar{\tau}^{\text{cut}}$ are 0, the scheme is monotonically non-decreasing, and thus either converges or goes to ∞. To force termination, the values are rounded up to an integer number of a chosen time resolution Δ (this is similar to rounding of burstiness to an integer number of bits in the original version of FP-TFA); then the iteration stops either when $\bar{\tau}^{\text{cut}}$ becomes stationary, or reaches a very large value called "infinite".

Then, if $\bar{\tau}^{\text{cut}}$ is finite, FF-TFA is called one last time, and hence a collection of valid, finite delay bounds at each node and a collection of valid, finite delay-jitters bounds for every flow from the source to each node in its path are obtained (because the cut network is feed-forward and locally stable). Otherwise, if the obtained $\bar{\tau}^{\text{cut}}$ bounds are infinite, GFP-TFA returns infinite bounds; in this case the network might or might not be stable.

Theorem 3 (Validity of GFP-TFA)**.** *Consider a FIFO network, as described in Section III, and apply Algorithm 2. Then, $\left(\bar{d}, \bar{\tau}\right)$ are upper bounds on per node delay and per-flow jitter.*

The proof is in Section VI. Note that if the original network is feed-forward, the cutset E^{cut} is empty and GFP-TFA applies

Algorithm 1: $(d, \tau) = \text{FF-TFA}\left(\alpha^0, \beta, E^{\text{cut}}, \tau^{\text{cut}}\right)$

Input : $\left(\alpha^0, \beta, E^{\text{cut}}, \tau^{\text{cut}}\right)$, collection of arrival curves of all flows at sources, collection of service curves of all nodes, a cutset that creates a feed-forward network, and a collection of delay-jitter bounds from source to cut, for every flow that is present at a cut edge.

Output: (d, τ), a collection of per-node delay bounds and a collection of delay-jitter bounds for every flow from its source to input of every node on its path.

1 $\tau_{f,n} \leftarrow 0$, $\forall$flow f and $\forall$node $n \in \text{path}(n)$;
2 **for** *each edge* $e = (n', n) \in E^{cut}$ **do**
3 **for** *each flow* f *carried by edge* e **do**
4 $\tau_{f,n} \leftarrow \tau^{\text{cut}}_{f,n}$;
5 **for** *each node* n *in the topological order of the cut network* **do**
6 $\alpha_n \leftarrow \text{aggregateArrivalCurve}_n\left(\alpha^0, \tau\right)$;
7 $l^{\text{min}}_n \leftarrow \min_{f \in \text{In}(n)} l^{\text{min}}_f$;
8 $d_n \leftarrow \text{hDev}\left(\alpha_n - l^{\text{min}}_n, \beta_n\right) + \frac{l^{\text{min}}_n}{c_n}$;
9 **for** *each edge* $e = (n, n')$ *in the original, uncut network* **do**
10 **for** *each flow* f *carried by edge* e **do**
11 $\tau_{f,n'} \leftarrow \tau_{f,n} + (d_n - \frac{l^{\text{min}}_n}{c_n})$;
12 **return** (d, τ);
13 **Function** $\alpha_n = \textit{aggregateArrivalCurve}_n\left(\alpha^0, \tau\right)$
14 $\alpha^{\text{fresh}}_n \leftarrow \sum_{\text{fresh} f \in \text{In}(n)} \alpha^0_f$;
15 **for** *each edge* $e \in \textit{In}(n)$ **do**
16 **for** *each flow* f *carried by edge* e **do**
17 $\alpha_{f,n} \leftarrow \alpha^0_f \oslash \delta_{\tau_{f,n}}$;
18 $\alpha_e \leftarrow \sum_{f \in e} \alpha_{f,n}$;
 // Effect of line-shaping Section IV-A
19 $\alpha_e \leftarrow \alpha_e \otimes \gamma_{c_e,0}$;
 // Effect of packetizer Section IV-A
20 $l^{\text{max}}_e \leftarrow \max_{f \in e} l^{\text{max}}_f$;
21 $\alpha_e \leftarrow \alpha_e \oslash \delta_{\frac{l^{\text{max}}_e}{c_e}}$;
22 $\alpha^{\text{transit}}_n \leftarrow \sum_{e \in \text{In}(n)} \alpha_e$;
23 $\alpha_n \leftarrow \alpha^{\text{transit}}_n + \alpha^{\text{fresh}}_n$;
24 **return** α_n;

Algorithm 2: $\left(\bar{d}, \bar{\tau}\right) = \text{GFP-TFA}\left(\alpha^0, \beta, E^{\text{cut}}\right)$

Input : $\left(\alpha^0, \beta, E^{\text{cut}}\right)$, collection of arrival curves of all flows at the source, collection of service curves of all nodes, and a cutset such that removing them creates a feed-forward network, respectively.

Output : $\left(\bar{d}, \bar{\tau}\right)$, a collection of valid per-node delay bound at every node and a collection of delay-jitter bound for every flow from the source to each node in its path, respectively.

1 $k \leftarrow 0$;
2 $\tau^{\text{cut},k}_{f,n} \leftarrow 0$, $\forall$flow f and $\forall e = (n', n) \in E^{\text{cut}}$;
3 **while** *(*$\tau^{cut,k} > \tau^{cut,k-1}$*) and (*$\tau^{cut,k} <$ *infinite)* **do**
4 $k \leftarrow k + 1$;
 // FF-TFA is described in Algorithm 1
5 $(d, \tau) \leftarrow \text{FF-TFA}\left(\alpha^0, \beta, E^{\text{cut}}, \tau^{\text{cut},k-1}\right)$;
6 Extract new values for $\tau^{\text{cut},k}$ from τ and round them up to an integer number of the minimum resolution Δ (e.g., 1 nanosecond);
7 $\bar{\tau}^{\text{cut}} \leftarrow \tau^{\text{cut},k}$;
8 **if** *all element of* $\bar{\tau}^{cut}$ *are finite* **then**
9 $\left(\bar{d}, \bar{\tau}\right) \leftarrow \text{FF-TFA}\left(\alpha^0, \beta, E^{\text{cut}}, \bar{\tau}^{\text{cut}}\right)$
10 **else**
11 $\left(\bar{d}, \bar{\tau}\right) \leftarrow \infty$;
12 **return** $\left(\bar{d}, \bar{\tau}\right)$;

FF-TFA only once. When arrival curves α^0_f of every flow f at the source are token-bucket and service curves β_n offered by every node n are rate-latency, GFP-TFA is essentially the same as FP-TFA of [7].

V. FH-TFA: A Practical Version of GFP-TFA

In this section, we present our second new version of TFA, FH-TFA, which provides the exact same bounds as theoretical GFP-TFA (when service curves are super-additive), while reducing the complexity.

The main difference between GFP-TFA and FH-TFA is as follows: Instead of working with original, UPP curves, FH-TFA only keeps a part of each UPP curve that affects the end-result, and beyond that, uses linear upper-bounds (resp. lower-bounds) of arrival (resp. service) curves. Specifically, FH-TFA first computes sufficient finite horizons for every curve by applying the compact domains of [16], presented in Theorem 1. Note that, with TFA, arrival curves of flows increase as we go deeper into the network, hence the compact domains required for delay computations increase as well. To address this, FH-TFA first applies GFP-TFA using linear curves, i.e., token-bucket arrival curves and rate-latency service curves; this is very fast and provides enough information to compute sufficient finite horizons. It then replaces every UPP curve by a UA curve that follows the UPP curve up to the computed sufficient horizon, and beyond that follows a linear upper-bound (resp. lower-bound) of the UPP arrival (resp. service) curve (see Fig. 4).

We first describe FH-TFA, and we then prove its validity and accuracy in Theorem 4.

A. Description of FH-TFA

FH-TFA is described in Algorithm 3:

- Arrival curves of all flows at the source (resp. service curves at all nodes) are lower-bounded and upper-bounded by token-bucket (resp. rate-latency) curves (lines 1-6); see Section V-A1.
- We obtain compact domains, required for delay computation, as in equation (1) of Theorem 1 at every node. As (1) requires to know a lower-bound and an upper-bound

for the arrival curve at a node, we run two instances of GFP-TFA (using linear curves) once with safe curves, (i.e., upper-bounds of arrival curves and lower bounds of service curves), and once with unsafe curves. When this is done, we obtain compact domains at every node n, i.e., $\left(h_n^{\alpha}, h_n^{\beta}\right)$ (lines 7-14). Note that if the application of GFP-TFA with safe curves returns infinite bounds (i.e., $(\bar{d}^{L''}, \bar{\tau}^{L''})$ at line 7 are infinite), we then cannot find finite compact domains and thus the rest of the algorithm is skipped, and GFP-TFA is applied with UPP, original curves (line 23). Such cases might happen in networks with cyclic dependencies that are highly loaded.

- We then compute a sufficient horizon for the arrival curve of each flow at the source; as the arrival curve of the flow increases along its path, this horizon should be large enough such that the arrival curve of the flow respects the compact domains at every node in its path (lines 15-18); see Section V-A2.
- A UA curve is constructed, for each UPP curve, that follows the original, UPP curve up to the computed sufficient horizon, and beyond that, follows the linear upper (resp. lower for service curves) of the original, UPP curve (lines 18-20); see Section V-A3 and Fig. 4.
- Lastly, GFP-TFA is run using UA curves (line 21).

1) Linear Upper and Lower Bounds of UPP Curves: This section describes the four functions used in lines 2, 3, 5, and 6 of Algorithm 3. Note that upper-bounds and lower-bounds of original, UPP curves can be freely chosen, as they are used in the method only to compute sufficient horizons, and they do not affect the end-results obtained by FH-TFA. We propose to use token-bucket curves (for arrival curves) and rate-latency curves (for service curves) as they can be easily computed, and are very tractable.

Consider flow f. We find two token-bucket curves $\alpha_f^{0,L'}$ and $\alpha_f^{0,L''}$ such that $\alpha_f^{0,L'} \leq \alpha_f^{0,\text{UPP}} \leq \alpha_f^{0,L''}$. Specifically, we compute the token-bucket function that upper (resp. lower) bounds $\alpha_f^{0,\text{UPP}}$, and achieves the minimum (resp. maximum) possible rate; let p and I be the period and increment of $\alpha_f^{0,\text{UPP}}$, then $\alpha_f^{0,L''} = \gamma_{r_f,b_f''}$ and $\alpha_f^{0,L'} = \gamma_{r_f,b_f'}$ with $r_f = \frac{I}{p}$ and

$$b_f'' = \min_{t\geq 0}\{\alpha_f^{0,\text{UPP}} - r_f t\} \text{ and } b_f' = \max_{t\geq 0}\{\alpha_f^{0,\text{UPP}} - r_f t\} \quad (7)$$

Observe that the long-term rate of $\alpha_f^{0,\text{UPP}}$, which is equal to $r_f = \frac{I}{p}$, is the minimum (resp. maximum) possible rate that $\alpha_f^{0,L''}$ (resp. $\alpha_f^{0,L'}$) can achieve, otherwise they are not an upper (resp. a lower) bound of $\alpha_f^{0,\text{UPP}}$. Then, burstiness b_f'' (resp. b_f') are chosen to be as small (resp. large) as possible. In a frequent case, where $\alpha_f^{0,\text{UPP}}$ is a stair function, say ν_{p_f,b_f} (i.e., a periodic flow that sends b_f bits each p_f seconds), $r_f = \frac{b_f}{p_f}$, $b_f'' = b_f$, and $b_f' = 0$ (see Fig. 2).

Consider node n. We find two rate-latency curves $\beta_n^{L''}$ and $\beta_n^{L'}$ such that $\beta_n^{L''} \leq \beta_n^{\text{UPP}} \leq \beta_n^{L'}$. Specifically, we compute the rate-latency function that lower (resp. upper) bounds β_n^{UPP}, and achieves the maximum (resp. minimum) possible rate; let

Algorithm 3: $(\bar{d}, \bar{\tau}) = \text{FH-TFA}\left(\alpha^{0,\text{UPP}}, \beta^{\text{UPP}}, E^{\text{cut}}\right)$

Input : $\left(\alpha^{0,\text{UPP}}, \beta^{\text{UPP}}, E^{\text{cut}}\right)$: collection of UPP arrival curves of all flows at the source, collection of UPP service curves of all nodes, and a cutset that creates a feed-forward network.

Output: $\left(\bar{d}, \bar{\tau}\right)$, a collection of valid per-node delay bound at every node and a collection of delay-jitter bounds for every flow from source to every node in its path.

1 **for** *each flow* f **do**
  // see Section V-A1
2 $\alpha_f^{0,L''} \leftarrow$ smallestAffineUpperBoundMinRate$(\alpha_f^{0,\text{UPP}})$;
3 $\alpha_f^{0,L'} \leftarrow$ largestAffineLowerBoundMaxRate$(\alpha_f^{0,\text{UPP}})$;
4 **for** *each node* n **do**
  // see Section V-A1
5 $\beta_n^{L''} \leftarrow$ largestRateLatencyLowerBoundMaxRate(β_n^{UPP});
6 $\beta_n^{L'} \leftarrow$ smallestRateLatencyUpperBoundMinRate(β_n^{UPP});
7 $(\bar{d}^{L''}, \bar{\tau}^{L''}) \leftarrow \text{GFP-TFA}\left(\alpha^{0,L''}, \beta^{L''}, E^{\text{cut}}\right)$;
8 $(\bar{d}^{L'}, \bar{\tau}^{L'}) \leftarrow \text{GFP-TFA}\left(\alpha^{0,L'}, \beta^{L'}, E^{\text{cut}}\right)$;
9 **if** $(\bar{d}^{L''}, \bar{\tau}^{L''})$ *is finite* **then**
10 **for** *each node* n **do**
    // see Function in Algorithm 1
11 $\alpha_n^{L''} \leftarrow \text{aggregateArrivalCurve}_n\left(\alpha^{0,L''}, \bar{\tau}^{L'''}\right)$;
12 $\alpha_n^{L'} \leftarrow \text{aggregateArrivalCurve}_n\left(\alpha^{0,L'}, \bar{\tau}^{L'}\right)$;
13 $\left(h^{\alpha}, h^{\beta}\right) \leftarrow$ apply (1) in Theorem 1 with $\alpha' = \alpha_n^{L'}$, $\alpha'' = \alpha_n^{L''}$, $\beta' = \beta_n^{L'}$, $\beta'' = \beta_n^{L''}$;
14 $h_n^{\alpha} \leftarrow h^{\alpha}$ and $h_n^{\beta} \leftarrow h^{\beta}$;
15 **for** *each flow* f **do**
    // see Section V-A2
16 $s_f \leftarrow$ sink of flow f;
17 $h_f^{\alpha} \leftarrow \bar{\tau}_{f,s_f}^{L''} + \max_{n\in\text{path}(f)} h_n^{\alpha}$;
    // see (9) in Section V-A3
18 $\alpha_f^{0,\text{UA}} \leftarrow \text{uaArrivalCurve}\left(\alpha_f^{0,\text{UPP}}, \alpha_f^{0,L''}, h_f^{\alpha}\right)$;
19 **for** *each node* n **do**
    // see (10) in Section V-A3
20 $\beta_n^{\text{UA}} \leftarrow \text{uaServiceCurve}\left(\beta_n^{\text{UPP}}, \beta_n^{L''}, h_n^{\beta}\right)$;
21 $\left(\bar{d}, \bar{\tau}\right) \leftarrow \text{GFP-TFA}\left(\alpha^{0,\text{UA}}, \beta^{\text{UA}}, E^{\text{cut}}\right)$;
22 **else**
23 $\left(\bar{d}, \bar{\tau}\right) \leftarrow \text{GFP-TFA}\left(\alpha^{0,\text{UPP}}, \beta^{\text{UPP}}, E^{\text{cut}}\right)$;
24 **return** $\left(\bar{d}, \bar{\tau}\right)$;

p and I be the pseudo-period and increment of β_n^{UPP}, then $\beta_n^{L''} = \beta_{c_n,L_n''}$ and $\beta_n^{L'} = \beta_{c_n,L_n'}$ with $c_n = \frac{I}{p}$ and

$$L_n'' = \min_{t\geq 0}\{t - \frac{\beta_n^{\text{UPP}}(t)}{c_n}\} \text{ and } L_n' = \max_{t\geq 0}\{t - \frac{\beta_n^{\text{UPP}}(t)}{c_n}\} \quad (8)$$

Observe that the long-term rate of β_n^{UPP}, which is equal to $c_n = \frac{I}{p}$, is the maximum (resp. minimum) possible rate that $\beta_n^{L''}$ (resp. $\beta_n^{L'}$) can achieve, otherwise they are not a lower (resp. an upper) bound of β_n^{UPP}. Observe that latency L_n'' (resp. L_n') are computed to be as small (resp. large) as possible.

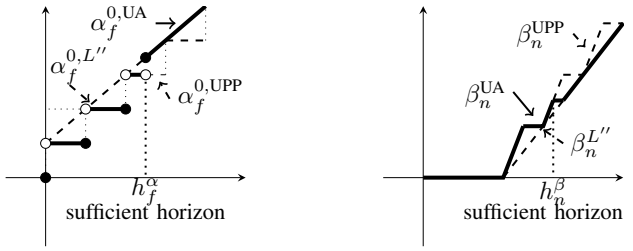


Fig. 4: $\alpha_f^{0,\mathrm{UPP}}$ (resp. β_n^{UPP}) is the original, UPP arrival curve for flow f at the source (resp. service curve of node n), $\alpha_f^{0,L''}$ (resp. $\beta_n^{L''}$) is the smallest token-bucket (resp. largest rate-latency) that upper bounds $\alpha_f^{0,\mathrm{UPP}}$ (resp. lower bounds β_n^{UPP}), and h_f^{α} (resp. h_n^{β}) is a sufficient horizon for the arrival curve of flow f (resp. service curve of node n). Then, $\alpha^{0,\mathrm{UA}}$ (resp. β_n^{UA}) follows $\alpha_f^{0,\mathrm{UPP}}$ (resp. β_n^{UPP}) in $[0, h_f^{\alpha}]$ (resp. $[0, h_n^{\beta}]$) and beyond that follows $\alpha_f^{0,L''}$ (resp. $\beta_n^{L''}$).

2) Sufficient Horizons for Arrival Curves of Flows at the Source: Consider flow f. As the arrival curve of flow f increases along its path, this horizon should be large enough such that arrival curve of the flow, respects the compact domain at every node in its path. Specifically, consider node n in the path of flow f. Then, at the input of node n, arrival of flow f is its arrival curve at the source, but shifted to left by a delay-jitter bound from the source to node n, say $\tau_{f,n}^{\mathrm{UPP}}$. Hence, sufficient horizon of flow f should be larger than or equal to $h_n^{\alpha} + \tau_{f,n}^{\mathrm{UPP}}$ at every node n in its path. Thus, we use $h_f^{\alpha} = \max_{n \in \mathrm{path}(f)} h_n^{\alpha} + \bar{\tau}_{f,s_f}^{L''}$ where s_f is the sink of flow f; note that $\bar{\tau}_{f,s_f}^{L''}$ is an upper-bound on the the end-to-end delay-jitter, for flow f, hence $\bar{\tau}_{f,s_f}^{L''} \geq \tau_{f,n}^{\mathrm{UPP}}$.

Observe that the already computed compact domain h_n^{β} is a sufficient horizon for the service curve of node n.

3) Construction of UA Curves: Consider flow f. Function $\alpha_f^{0,\mathrm{UA}} = \mathrm{uaArrivalCurve}\left(\alpha_f^{0,\mathrm{UPP}}, \alpha_f^{0,L''}, h_f^{\alpha}\right)$, at line 18 of Algorithm 3, constructs this UA curve as follows (see Fig. 4):

$$\alpha_f^{0,\mathrm{UA}}(t) = \begin{cases} \alpha_f^{0,\mathrm{UPP}}(t) & \text{if } t \leq h_f^{\alpha} \\ \alpha_f^{0,L''}(t) & \text{otherwise} \end{cases} \tag{9}$$

Consider node n. Function $\beta_n^{\mathrm{UA}} = \mathrm{uaServiceCurve}\left(\beta_n^{\mathrm{UPP}}, \beta_n^{L''}, h_n^{\beta}\right)$, at line 20 of Algorithm 3, constructs this UA curve as follows (see Fig. 4):

$$\beta_n^{\mathrm{UA}}(t) = \begin{cases} \beta_n^{\mathrm{UPP}}(t) & \text{if } t \leq h_n^{\beta} \\ \min\left(\beta_n^{\mathrm{UPP}}(h_n^{\beta}), \beta_n^{L''}(t)\right) & \text{otherwise} \end{cases} \tag{10}$$

B. Validity and Accuracy of FH-TFA

Theorem 4 (Validity and Accuracy of FH-TFA)**.** *Consider a FIFO network, as described in Section III and Algorithms 2 and 3. Then, (1) FH-TFA provides valid bounds. (2) If original, UPP service curves of all nodes are super-additive, FH-TFA and GFP-TFA provide the exact same bounds.*

The proof is in Section VI. In the common case where service curves are super-additive, FH-TFA and GFP-TFA return the same output, i.e., FH-TFA returns finite bounds if and only if GFP-TFA returns finite bounds and, if bounds are finite, both algorithms provide the same bounds. FH-TFA is thus a practical alternative to GFP-TFA, which may become too complex when there are many UPP curves.

FH-TFA is always applicable and provides valid bounds, as stated in item (1) in the theorem, and super-additivity of service curves is not a requirement for FH-TFA; furthermore, the obtained bounds by FH-TFA are guaranteed to be less than or equal to those obtained by FP-TFA (which uses linear curves). We use super-additivity of service curves only to prove that bounds obtained by FH-TFA are exactly equal to those of GFP-TFA. When service curves are not super-additive, it is not clear whether they are equal to those that would be obtained by GFP-TFA (using original, UPP curves), and this is left for further study. Note that service curves of frequent schedulers, including DRR, WRR, IWRR, CBS, and Non-Preemptive Strict-Priority, are super-additive, as explained in Section II-A2.

Remarks on time and space complexity: GFP-TFA iteratively calls function FF-TFA (line 5 of Algorithm 2) and the number of iterations cannot be determined in advance, however, we analyze the complexity of one instance of FF-TFA. FF-TFA (Algorithm 1) includes operations such as addition, min-plus convolution, horizontal deviation, etc. on UPP or UA functions; [17] formally studies the computational complexity of such operations, the addition being the most costly [17, Proposition 10]. FF-TFA calls once function $\mathrm{aggragteArrivalCurve}_n$ for each node n (line 6 of Algorithm 3). With the addition being the most costly, the complexity of $\mathrm{aggragteArrivalCurve}_n$ is in the order of the complexity of summing arrival curves of all flows. Specifically, assume that there are F flows with UPP arrival curves $\alpha_f^{0,\mathrm{UPP}}$ with a pseudo-period p_f and a rank T_f for $f \in [1, F]$ (see Section II-A1). Let p be the hyper-period of the aggregate and $T = \max_{f \in [1,F]} T_f$, and let M_f be the number of segments required to define $\alpha_f^{0,\mathrm{UPP}}$ in $[0, T+p)$ for $f \in [1, F]$. Then, by [17], the required space to compute $\sum_{f=1}^{F} \alpha_f^{0,\mathrm{UPP}}$ is $\sum_{f=1}^{F} M_f$ and the addition can be computed in time $O((\sum_{f=1}^{F} M_f) \log_2 F)$. Thus, as we have N nodes, one instance of FF-TFA inside GFP-TFA (using UPP curves) is run in time $O(N(\sum_{f=1}^{F} M_f) \log_2 F)$; the required space for GFP-TFA is $O(N(\sum_{f=1}^{F} M_f) \log_2 F)$ plus the space required to store service curves β_n^{UPP} for all node n.

However, FH-TFA restricts UPP functions to a finite horizon (i.e., UA functions) and applies GFP-TFA with UA curves. Specifically, it applies GFP-TFA with $\alpha_f^{0,\mathrm{UA}}$ defined in (9). Let M_f^h be the number of segments required to define $\alpha_f^{0,\mathrm{UPP}}$ in $[0, h_f^{\alpha})$ (i.e., the number of segments required to define $\alpha_f^{0,\mathrm{UA}}$; see Figure 4) for $f \in [1, F]$. With the same reasoning as the previous paragraph, one instance of FF-TFA inside GFP-TFA, using UA curves, is run in time $O(N(\sum_{i=1}^{F} M_f^h) \log_2 F)$; the required space for FH-TFA (GFP-TFA using UA curves) is $O(N(\sum_{i=1}^{F} M_f^h) \log_2 F)$ plus the space required to store

service curves β_n^{UA} for all node n. As the number of segments in the horizon, $\sum_{f=1}^{F} M_i^h$ (i.e., transient part) might be extremely smaller than those of the hyper-period, $\sum_{f=1}^{F} M_f$ (see Figure 3 where $\sum_{f=1}^{F} M_i^h = 13$ and $\sum_{f=1}^{F} M_f = 2020$.), FH-TFA might significantly reduce the time and space complexity compared to GFP-TFA. As observed in Section VII, GFP-TFA has high computational complexity and is an impractical method, while FH-TFA is successfully applied to each of our industrial case studies.

VI. Proofs

Proof of Theorem 2. As the packetizer is fed by a transmission link with rate c, it follows that when a packet of size l arrives to the packetizer, it is released after a time equal to $\frac{l}{c}$. Hence, the release time of any packet is upper-bounded by $\frac{l^{\max}}{c}$. Then, by [2, Theorem 6.2], it follows that a pure delay $\delta_{\frac{l^{\max}}{c}}$ is a service curve for the packetizer, hence the output arrival curve is min-plus deconvolution of the input arrival curve and $\delta_{\frac{l^{\max}}{c}}$. □

Proof of Theorem 3. The proof follows similar steps as in the proof of Theorem 2 of [7]. Let F be the mapping that maps $\tau^{\text{cut},k-1}$ to $\tau^{\text{cut},k}$ in lines 1-6 of Algorithm 2.

Fix an acceptable trajectory scenario, i.e., the set of all cumulative arrival functions at all nodes in the networks such that arrival curve and service curve constraints are met. Let t^{prop} be the minimum of all link propagation delays and $\theta = \frac{t^{\text{prop}}}{2}$, so that $0 < \theta < t^{\text{prop}}$. Consider an arbitrary $\eta \geq 0$.

• Let W (resp. U) represents the set of point located just before (resp. just after) the cuts. Note that in the original, uncut network, U and W are connected via some links. Also, the network between U and W is feed-forward.

• Let V represent the points located exactly θ seconds before W. As $\theta < t^{\text{prop}}$ and t^{prop} is the minimum propagation delay, V is on the same links as W.

• For $M = \{U, V, W\}$, let C_M represent the collection of cumulative arrival functions; let C_M^η represent the collection of cumulative arrival functions stopped at time η, i.e., $C_M^\eta(t) = \min\left(C_M(t), C_M(\eta)\right)$; let $C_M^{'\eta}$ represent the collection of cumulative arrival functions when inputs at U and all sources are stopped at time η.

• For $M = \{U, V, W\}$, we denote by τ_M the collection of worst-case delay jitter from the source to M for all flows at M; we denote by τ_M^η collection of worst-case delay jitter from the source to M for all flows at M observed up to time η; we denote by $\tau_M^{'\eta}$ collection of worst-case delay jitter from the source to M for all flows at M when inputs at U and all sources are stopped at time η.

First, observe that τ_M^η and $\tau_M^{'\eta}$ are finite. This is because as η is finite and as sources are constrained at the source, a finite number of bits ever entered the network, hence delays are finite. Then:

1) observe that $\tau_V^\eta \leq \tau_V^{'\eta}$; This is implied by the fact that the network is causal, and hence, $\forall t \leq \eta, C_M^{'\eta}(t) = C_M^\eta(t)$ and $\forall t > \eta, C_M^{'\eta}(t) \geq C_M^\eta(t)$.

2) As the network between U and W is feed-forward, function F computes a bound on the delay-jitters from the source to W, given delay-jitters from the source to U, and hence, $\tau_W^{'\eta} \leq F\left(\tau_U^\eta\right)$.

3) As there is a constant delay θ between V and W, $\forall t \geq 0$, $C_W^{'\eta}(t+\theta) = C_V^{'\eta}(t)$; it follows that $\tau_W^{'\eta} \leq F\left(\tau_U^\eta\right)$ $\tau_V^{'\eta} = \tau_W^{'\eta}$.

Combine **1)**, **2)**, and **3)** and obtain

$$\tau_V^\eta = F\left(\tau_U^\eta\right) \tag{11}$$

Observe that $\tau_V^\eta = \tau_W^{\eta+\theta}$. This is because the exact same traffic that is observed by V between 0 to η is observed by W between θ to $\eta + \theta$, and the difference is only a constant delay θ; this does not change the delay-jitter. Also, as in the original, uncut network U and W are connected, $\tau_W^\eta = \tau_U^{\eta+\theta}$. Thus, $\tau_V^\eta = \tau_U^{\eta+\theta}$. Combine this with (11) and obtain $\tau_U^{\eta+\theta} = F\left(\tau_U^\eta\right)$. This is valid for every $\eta \geq 0$, apply it with $\eta = k\theta$ and obtain: $\forall k \geq 0, \tau_U^{(k+1)\theta} = F\left(\tau_U^{k\theta}\right)$. As the network is empty at time zero, $\tau_U^0 = 0$. As F is wide-sense increasing and $F\left(\bar{\tau}\right) = \bar{\tau}$ and a simple induction, it follows that $\tau_U^{k\theta} \leq \bar{\tau}$ and hence $\tau_U^\eta \leq \bar{\tau}$ for $\eta \geq 0$. Hence, $\sup_{\eta \geq 0} \tau_U^\eta \leq \bar{\tau}$, i.e., for any acceptable trajectory scenario, the worst-case delay jitter bound from a source to a cut for every flow at cuts is upper-bounded by $\bar{\tau}$. □

Proof of Theorem 4. The validity of the bounds is directly implied by the validity of the constructed UA curves. Specifically, UA curves constructed in FH-TFA, are safe upper/lower bounds of the original, UPP curves, i.e., $\alpha^{0,\text{UPP}} \leq \alpha^{0,\text{UA}}$ and $\beta^{\text{UA}} \leq \beta^{\text{UPP}}$ (see Fig. 4). Then, as GFP-TFA is isotone, it follows that bounds obtained by FH-TFA are larger than or equal to those obtained by GFP-TFA (using original, UPP curves), hence valid and (1) is shown.

We now proceed to show item (2) when service curves are super-additive. We first prove the following lemmas.

Lemma 1. *Consider Algorithm 3 and consider node n. Then, the computation of the delay bound at node n, $d_n^{UPP} = hDev\left(\alpha_n^{UPP} - l_n^{min}, \beta_n^{UPP}\right) + \frac{l_n^{min}}{c_n}$ (see line 8 of Algorithm 1), involves only values of $\alpha_n^{UPP}(t)$ for $t \in [0, h_n^\alpha]$ and $\beta_n^{UPP}(t)$ for $t \in \left[0, h_n^\beta\right]$, where α_n^{UPP} is the aggregate arrival curve at node n computed by GFP-TFA using UPP curves, l_n^{min} is the minimum packet size of flows at node n, and c_n is the transmission rate at node n.*

Proof. As $\alpha_f^{0,L''} \geq \alpha_f^{0,\text{UPP}}$ (resp. $\alpha_f^{0,L'} \leq \alpha_f^{0,\text{UPP}}$) and $\beta_n^{L''} \leq \beta_n^{\text{UPP}}$ (resp. $\beta_n^{L'} \geq \beta_n^{\text{UPP}}$) and as GFP-TFA is isotone, it follows that $(d^{L'}, \tau^{L'}) \leq (d^{\text{UPP}}, \tau^{\text{UPP}}) \leq (d^{L''}, \tau^{L''})$. Thus, $\alpha_n^{L'} \leq \alpha_n^{\text{UPP}} \leq \alpha_n^{L''}$. Also, by construction, $\beta_n^{L''} \leq \beta_n^{\text{UPP}} \leq \beta_n^{L'}$. Apply Theorem 1 with $\alpha = \alpha_n^{\text{UPP}}$, $\alpha'' = \alpha_n^{L''}$, $\alpha' = \alpha_n^{L'}$, $\beta = \beta_n^{\text{UPP}}$, $\beta'' = \beta_n^{L''}$, and $\beta' = \beta_n^{L'}$ to conclude that $\text{hDev}\left(\alpha_n^{\text{UPP}}, \beta_n^{\text{UPP}}\right)$ depends only on values of $\alpha_n^{\text{UPP}}(t)$ for $t \in [0, h_n^\alpha]$ and $\beta_n^{\text{UPP}}(t)$ for $t \in \left[0, h_n^\beta\right]$. Observe that these compact domains are also sufficient for the computation of $\text{hDev}\left(\alpha_n^{\text{UPP}} - l_n^{\min}, \beta_n^{\text{UPP}}\right)$. □

Lemma 2. *Consider Algorithm 3 and assume a τ^{cut} such that $0 \leq \tau^{cut} \leq \bar{\tau}^{cut,L''}$ where $\bar{\tau}^{cut,L''}$ are delay-jitters for flows at cuts extracted from $\bar{\tau}^{L''}$, obtained at line 7. Then, FF-TFA $\left(\alpha^{0,UPP}, \beta^{UPP}, E^{cut}, \tau^{cut}\right)$ = FF-TFA $\left(\alpha^{0,UA}, \beta^{UA}, E^{cut}, \tau^{cut}\right)$ where FF-TFA is described in Algorithm 1.*

Proof. First, we show that for every node n,

$$\forall t \in [0,\ h_n^{\beta}],\ \beta_n^{\text{UA}}(t) = \beta_n^{\text{UPP}}(t) \tag{12}$$

$$\forall f \in \text{flows}(n),\ \forall t \in [0,\ h_n^{\alpha}],\ \alpha_{f,n}^{\text{UA}}(t) = \alpha_{f,n}^{\text{UPP}}(t) \tag{13}$$

$$d_n^{\text{UA}} = d_n^{\text{UPP}} \tag{14}$$

Observe that (12) is by construction. We now prove (13) and (14). by induction on node n. Recall that, as explained in Section III, nodes are labeled in a topological order of the cut network (such orders exists as the cut network is feed-forward). We assume, to simplify the notation, that the node label n is an integer that reflects this topological order. The base case of our induction is thus for a node n that is an edge node, i.e., where all flows at node n are either fresh or cut.

Base Case: n is an edge node in the cut network

f **is fresh**: We show that for every fresh f, $\forall t \in [0,\ h_n^{\alpha}],\ \alpha_f^{0,\text{UA}}(t) = \alpha_f^{0,\text{UPP}}(t)$. By construction, $\alpha_f^{0,\text{UA}}(t) = \alpha_f^{0,\text{UPP}}(t)$ for $t \in [0,\ h_f^{\alpha}]$, hence we should show that $h_f^{\alpha} \geq h_n^{\alpha}$: Recall that $h_f^{\alpha} = \max_{n \in \text{Path}(f)} h_n^{\alpha} + \bar{\tau}_{f,s_f}^{L''}$, and as $\bar{\tau}_{f,s_f}^{L''} \geq 0$, it follows that $h_f^{\alpha} \geq h_n^{\alpha}$.

f **is cut**: For every flow f that is cut at node n (if any), we show that $\forall t \in [0,\ h_n^{\alpha}],\ \alpha_f^{0,\text{UA}}\left(t + \tau_f^{\text{cut}}\right) = \alpha_f^{0,\text{UPP}}\left(t + \tau_f^{\text{cut}}\right)$. By construction, $\alpha_f^{0,\text{UA}}(t) = \alpha_f^{0,\text{UPP}}(t)$ for $t \in [0,\ h_f^{\alpha}]$, hence we must show that $h_f^{\alpha} \geq h_n^{\alpha} + \tau_{f,n}^{\text{cut}}$: Recall that $h_f^{\alpha} = \max_{n \in \text{Path}(f)} h_n^{\alpha} + \bar{\tau}_{f,s_f}^{L''}$; observe that $h_f^{\alpha} \geq h_n^{\alpha} + \bar{\tau}_{f,s_f}^{L''}$. Then, as $s_f \geq n$ and as delay-jitter increases along the path, $\bar{\tau}_{f,s_f}^{L''} \geq \bar{\tau}_{f,n}^{L''}$, hence $h_f^{\alpha} \geq h_n^{\alpha} + \bar{\tau}_{f,n}^{L''}$. By construction, as $\bar{\tau}^{\text{cut},L''}$ is extracted form $\bar{\tau}^{L''}$, we have $\bar{\tau}_{f,n}^{L''} = \bar{\tau}_{f,n}^{\text{cut},L''}$, hence $h_f^{\alpha} \geq h_n^{\alpha} + \bar{\tau}_{f,n}^{\text{cut},L''}$. Lastly, by hypothesis of the lemma, $\bar{\tau}^{\text{cut},L''} \geq \tau^{\text{cut}}$, and therefore, $h_f^{\alpha} \geq h_n^{\alpha} + \tau_{f,n}^{\text{cut}}$.

Hence, the base case is shown for (13). Then, by Lemma 1, (12), and (13), it follows that $d_n^{\text{UA}} = d_n^{\text{UPP}}$ hence the base case is shown for (14).

Induction Case: n is not an edge node in the cut network

In this case, we assume that for every $n' < n$, (13) and (14) holds, and we prove them at node n. First, observe that, for a fresh flow or cut flow $f \in \text{flows}(n)$, the proof of (13) is similar to the base case. Second, for a transit flow $f \in \text{flows}(n)$, (13) can be rewritten as follows:

$$\forall t \in [0,\ h_n^{\alpha}],\ \alpha_f^{0,\text{UA}}(t + \tau_{f,n}^{\text{UA}}) = \alpha_f^{0,\text{UPP}}(t + \tau_{f,n}^{\text{UPP}}) \tag{15}$$

By induction hypothesis for (14), we have $d_{n'}^{\text{UA}} = d_{n'}^{\text{UPP}}$ for $n' < n$, thus $\tau_{f,n}^{\text{UA}} = \tau_{f,n}^{\text{UPP}}$ is implied by the construction. Hence, (15) can be rewritten as follows:

$$\forall t \in [0,\ h_n^{\alpha}],\ \alpha_f^{0,\text{UA}}(t + \tau_{f,n}^{\text{UPP}}) = \alpha_f^{0,\text{UPP}}(t + \tau_{f,n}^{\text{UPP}}) \tag{16}$$

Recall that $\alpha_f^{0,\text{UA}}(t) = \alpha_f^{0,\text{UPP}}(t)$ for $t \in [0,\ h_f^{\alpha}]$, we need to show that $h_f^{\alpha} \geq \tau_{f,n}^{\text{UPP}} + h_n^{\alpha}$. By construction, $h_f^{\alpha} = \max_{n \in \text{Path}(f)} h_n^{\alpha} + \bar{\tau}_{f,s_f}^{L''} \geq h_n^{\alpha} + \bar{\tau}_{f,s_f}^{L''}$. Observe that $\bar{\tau}_{f,s_f}^{L''} \geq \bar{\tau}_{f,n}^{L''}$ and $\bar{\tau}_{f,n}^{L''} \geq \tau_{f,n}^{\text{UPP}}$, and hence $h_f^{\alpha} \geq h_n^{\alpha} + \tau_{f,n}^{\text{UPP}}$. This shows (15) and hence the induction case is shown for (13). Then, by Lemma 1, (12), and (13), it follows that $d_n^{\text{UA}} = d_n^{\text{UPP}}$ hence the induction case is shown for (14). Therefore, (13) and (14) are shown. Lastly, by (14), both methods return the same collection of per-node delay bounds, hence also of delay-jitters for flows. □

We now proceed to conclude the proof of item (2) of Theorem 4. Observe that if one of the components of $(\bar{d}^{L''}, \bar{\tau}^{L''})$, obtained in line 7 of Algorithm 3, is infinite, then FH-TFA applies GFP-TFA with the original, UPP curves hence (2) is concluded. We now proceed by assuming that $(\bar{d}^{L''}, \bar{\tau}^{L''})$ is finite (thus the network is stable) and show that GFP-TFA$\left(\alpha^{0,\text{UA}}, \beta^{\text{UA}}, E^{\text{cut}}\right)$ returns the same bounds as GFP-TFA$\left(\alpha^{0,\text{UPP}}, \beta^{\text{UPP}}, E^{\text{cut}}\right)$. First, observe that GFP-TFA starts with $\tau^{\text{cut}} = 0$. Next, as FF-TFA is isotone, bounds obtained at each iteration inside GFP-TFA, using UA or UPP curves, are upper-bounded by those obtained using linear curves; hence, for flows at cuts, delay-jitter bounds obtained at each iteration using UA or UPP curves are always upper-bounded by the fix-point $\bar{\tau}^{\text{cut},L''}$. Then, iteratively apply Lemma 2 to conclude that at each iteration where GFP-TFA calls FF-TFA the same bounds are obtained using UA and UPP curves. Therefore, FH-TFA and GFP-TFA provide the exact same finite bounds. □

VII. Numerical Evaluation

We use three real, industrial networks provided by industrial partners of RTaW, a leading company in Ethernet TSN design, performance evaluation and automated configuration tools; these examples are anonymized and slightly changed. We first explain each network, and we then present the results.

A. A Feed-Forward Network

It is illustrated in Fig. 5 (a): This network is feed-forward. It has 13 end-nodes and 5 switches: Link speeds are $c = 2$Gb/s and switches (blue squares) have switching latency equal to $L = 15\mu$s, i.e., nodes offer a rate-latency service curve $\beta_{c,L}$ with $L = 0$ for end-nodes and $L = 15$ for switches. There are 50 periodic flows: periods are 0.03, 0.06, 0.12, 0.24, 1, 5, 10, 20, 25, 60, 125, 200, and 1000 ms; packet sizes are $[130, 1360]$ bytes.

B. A Small-sized Network with Cyclic Dependencies

It is illustrated in Fig. 5 (b): This network has cyclic dependencies. It has 6 end-nodes and 3 switches: Link speeds are $c = 2$Gb/s and switches (blue squares) have switching latency equal to $L = 1.5\mu$s, i.e., nodes offer a rate-latency service curve $\beta_{c,L}$ with $L = 0$ for end-nodes and $L = 1.5$ for switches. There are 56 periodic flows: periods are 0.24, 0.25, 0.28, 0.5, 0.8, 1, 1.28, 2, 2.4, 20, 24, 100, 800, 1600, and 2400 ms; packet sizes are $[65, 530]$ bytes.

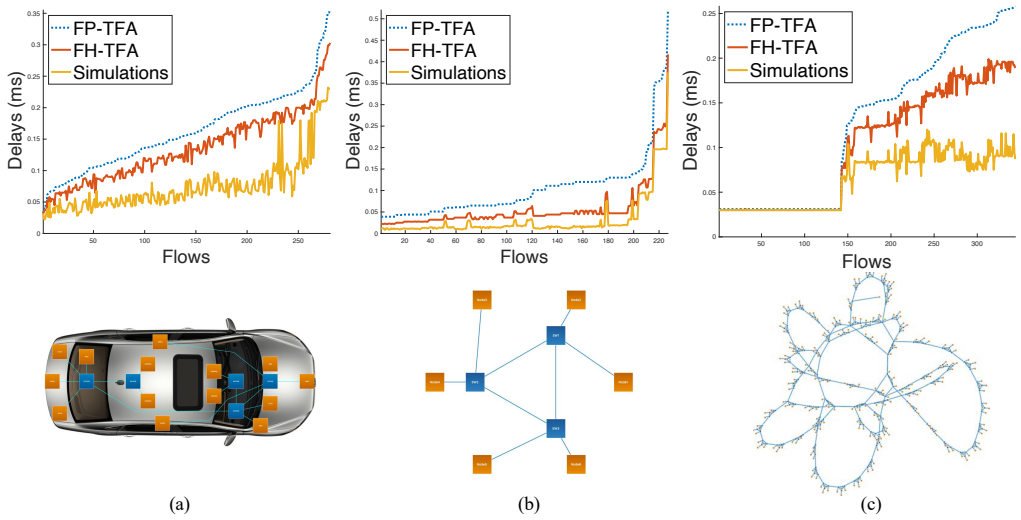


Fig. 5: Delay bounds obtained with FP-TFA, which uses token-bucket arrival curves, and our FH-TFA. GFP-TFA with original UPP curves, fails as we face a memory size error. FH-TFA significantly improves the bounds compared to those obtained by FP-TFA. Moreover, the improvement is more considerable when we compare the tightness gaps using the simulation values. See Table II for run-times. Flows are ordered by values of FP-TFA.

C. An Extremely Large Network with Cyclic Dependencies

It is illustrated in Fig. 5 (c): This network has cyclic dependencies. It has 291 end-nodes and 220 switches: Link speeds are $c = \{0.1, 1, 10\}$Gb/s and switches (blue squares) have switching latency equal to $L = 2\mu$s. There are 486 periodic flows: periods are 0.125, 0.24, 0.28, 0.608, and 10 ms; packet sizes are $[96, 1518]$ bytes.

D. Results

We apply three methods to compute end-to-end delay bounds: 1) GFP-TFA with original UPP curves, i.e., stair arrival curve for flows (see Section II-A2); 2) FP-TFA; 3) FH-TFA. Also, we use RTAW-Pegase tool to do simulation where we compute some true delays for each flow that serve as lower-bounds on the worst-case. Note that the true worst-case is between the simulation bound and the smallest delay bound, hence this provides a bound on the tightness.

First, we observe that in all three networks, GFP-TFA with original UPP curves, fails as we face a memory size error. Indeed as explained before, GFP-TFA has high computational complexity and is an impractical method that is used to validate our second version of TFA, FH-TFA. Second, FP-TFA, which uses token-bucket arrival curves, and FH-TFA are successfully applied to each network, and obtained delay bounds are illustrated in Fig. 5: FH-TFA significantly improves the bounds compared to those obtained by FP-TFA with token-bucket arrival curves; namely, FH-TFA improves bounds by around 20% (median) and 65% (maximum) compared to FP-TFA in our examples. This increases efficiency as more traffic can be accepted while meeting required deadlines, and making the network more robust to changes in network conditions and physical infrastructure. Moreover, the improvement is more considerable when we compare the tightness gap using the simulations value. Note that as service curve are rate-latency, hence super-additive, by Theorem 4, FH-TFA and GFP-TFA provide the exact same bounds.

TABLE II: Run-times for networks of Fig. 5

Method	Network (a)	Network (b)	Network (c)
GFP-TFA with UPP curves	-	-	-
FP-TFA (s)	[0.053, 0.059]	[2.66, 2.68]	[10.8, 10.88]
FH-TFA (s)	[0.8, 0.82]	[7.07, 7.1]	[35.17, 35.37]

We also provide run-times in Table II: We use the min-plus interpreter of RTaW [12] that has infinite precision using rational numbers. We use Java on a 2.6 GHz 6-Core Intel Core i7 computer. As GFP-TFA with UPP curves fails hence no run-time is provided; for the other methods, we run the program ten times, and 95% confidence interval is reported. FH-TFA is fast and practical, even for the extremely large network. Note that, in our experiment, we use the minimum resolution Δ equal to 1 nanosecond (line 6 of Algorithm 2).

VIII. Conclusion

TFA quickly becomes intractable when there are large number of periodic sources and UPP curves. This problem is frequently observed in time-sensitive and real-time networks. We provide a practical solution, FH-TFA, that obtains delay bounds that are formally proven, and can handle large number of periodic sources with different periods. In future research, we plan to explore other versions of TFA that do not require cuts, such as AsyncTFA and AltTFA [8], and to also make them practical and applicable in the presence of a large number of periodic sources and UPP curves. This is of interest as FH-TFA is currently based on GFP-TFA, which is a sequential algorithm, but other versions of TFA can benefit from parallel computations, thus might further reduce the run times.

REFERENCES

[1] J.-Y. Le Boudec and P. Thiran, *Network Calculus: A Theory of Deterministic Queuing Systems for the Internet*. Springer Science & Business Media, 2001, vol. 2050.

[2] A. Bouillard, M. Boyer, and E. Le Corronc, *Deterministic Network Calculus: From Theory to Practical Implementation*. Wiley-ISTE.

[3] D. B. Chokshi and P. Bhaduri, "Modeling fixed priority non-preemptive scheduling with real-time calculus," in *2008 14th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications*, Aug 2008, pp. 387–392.

[4] L. Thiele, S. Chakraborty, and M. Naedele, "Real-time calculus for scheduling hard real-time systems," in *2000 IEEE International Symposium on Circuits and Systems (ISCAS)*, vol. 4, May 2000, pp. 101–104 vol.4.

[5] J. B. Schmitt and F. A. Zdarsky, "The disco network calculator: A toolbox for worst case analysis," in *Proceedings of the 1st International Conference on Performance Evaluation Methodolgies and Tools*, ser. valuetools '06. New York, NY, USA: Association for Computing Machinery, 2006, p. 8–es. [Online]. Available: https://doi.org/10.1145/1190095.1190105

[6] A. Mifadoui and T. Leydier, "Beyond the Accuracy-Complexity Tradeoffs of CompositionalAnalyses using Network Calculus for Complex Networks," in *10th International Workshop on Compositional Theory and Technology for Real-Time Embedded Systems (co-located with RTSS 2017)*, Paris, France, December 2017, pp. pp. 1–8. [Online]. Available: https://hal.archives-ouvertes.fr/hal-01690096

[7] L. Thomas, J.-Y. Le Boudec, and A. Mifdaoui, "On cyclic dependencies and regulators in time-sensitive networks," in *2019 IEEE Real-Time Systems Symposium (RTSS)*, 2019, pp. 299–311.

[8] S. Plassart and J.-Y. Le Boudec, "Equivalent versions of total flow analysis," *CoRR*, vol. abs/2111.01827, 2021. [Online]. Available: https://arxiv.org/abs/2111.01827

[9] S. M. Tabatabaee and J.-Y. Le Boudec, "Deficit round-robin: A second network calculus analysis," *IEEE/ACM Transactions on Networking*, pp. 1–15, 2022.

[10] S. M. TABATABAEE, J.-Y. L. BOUDEC, and M. BOYER, "Interleaved weighted round-robin: A network calculus analysis," *IEICE Transactions on Communications*, vol. E104.B, no. 12, pp. 1479–1493, 2021.

[11] H. Daigmorte, M. Boyer, and L. Zhao, "Modelling in network calculus a TSN architecture mixing Time-Triggered, Credit Based Shaper and Best-Effort queues," Jun. 2018, working paper or preprint. [Online]. Available: https://hal.archives-ouvertes.fr/hal-01814211

[12] "RealTime-at-Work online Min-Plus interpreter for Network Calculus," https://www.realtimeatwork.com/minplus-playground, accessed: year-month-day.

[13] R. Zippo and G. Stea, "Nancy: an efficient parallel network calculus library," 2022. [Online]. Available: https://arxiv.org/abs/2205.11449

[14] S. Bondorf and J. B. Schmitt, "The DiscoDNC v2 – a comprehensive tool for deterministic network calculus," in *Proc. of the International Conference on Performance Evaluation Methodologies and Tools*, ser. ValueTools '14, December 2014, pp. 44–49. [Online]. Available: https://dl.acm.org/citation.cfm?id=2747659

[15] U. Suppiger, S. Perathoner, K. Lampka, and L. Thiele, "Modular performance analysis of large-scale distributed embedded systems: An industrial case study," Zurich, Report, 2010-11.

[16] K. Lampka, S. Bondorf, and J. Schmitt, "Achieving efficiency without sacrificing model accuracy: Network calculus on compact domains," in *2016 IEEE 24th International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS)*, 2016, pp. 313–318.

[17] A. Bouillard and É. Thierry, "An algorithmic toolbox for network calculus," *Discrete Event Dynamic Systems*, vol. 18, no. 1, pp. 3–49, 2008. [Online]. Available: https://doi.org/10.1007/s10626-007-0028-x

[18] R. Henia, A. Hamann, M. Jersak, R. Racu, K. Richter, and R. Ernst, "System level performance analysis - the symta/s approach," *Computers and Digital Techniques, IEE Proceedings -*, vol. 152, pp. 148 – 166, 04 2005.

[19] E. Wandeler and L. Thiele, "Real-Time Calculus (RTC) Toolbox," `http://www.mpa.ethz.ch/Rtctoolbox`, 2006. [Online]. Available: http://www.mpa.ethz.ch/Rtctoolbox

[20] R. Zippo and G. Stea, "Computationally efficient worst-case analysis of flow-controlled networks with network calculus," 2022. [Online]. Available: https://arxiv.org/abs/2203.02497

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

[22] N. Guan and W. Yi, "Finitary real-time calculus: Efficient performance analysis of distributed embedded systems," in *2013 IEEE 34th Real-Time Systems Symposium*, 2013, pp. 330–339.

[23] K. Lampka, S. Bondorf, J. B. Schmitt, N. Guan, and W. Yi, "Generalized finitary real-time calculus," in *IEEE INFOCOM 2017 - IEEE Conference on Computer Communications*, 2017, pp. 1–9.

[24] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms, Third Edition*, 3rd ed. The MIT Press, 2009.

[25] A. Mifdaoui and T. Leydier, "Beyond the Accuracy-Complexity Tradeoffs of CompositionalAnalyses using Network Calculus for Complex Networks," in *10th International Workshop on Compositional Theory and Technology for Real-Time Embedded Systems (co-located with RTSS 2017)*, Paris, France, December 2017, pp. pp. 1–8. [Online]. Available: https://hal.archives-ouvertes.fr/hal-01690096

[26] J. Grieu, "Analyse et évaluation de techniques de commutation ethernet pour l'interconnexion des systèmes avioniques," September 2004. [Online]. Available: https://oatao.univ-toulouse.fr/7385/

[27] E. Mohammadpour, E. Stai, and J.-Y. L. Boudec, "Improved network calculus delay bounds in time-sensitive networks," 2022. [Online]. Available: https://arxiv.org/abs/2204.10906