Non-definitive version – Final version to appear in the 40th IEEE Real-Time Systems Symposium (RTSS)

On Cyclic Dependencies and Regulators in Time-Sensitive Networks

Ludovic Thomas
I&C
EPFL
Lausanne, Switzerland
ludovic.thomas@epfl.ch

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

Ahlem Mifdaoui
ISAE-Supaéro
Université de Toulouse
Toulouse, France
ahlem.mifdaoui@isae-supaero.fr

***Abstract*—For time-sensitive networks, as in the context of IEEE TSN and IETF Detnet, cyclic dependencies are associated with certain fundamental properties such as improving availability and decreasing reconfiguration effort. Nevertheless, the existence of cyclic dependencies can cause very large latency bounds or even global instability, thus making the proof of the timing predictability of such networks a much more challenging issue. Cyclic dependencies can be removed by reshaping flows inside the network, by means of regulators. We consider FIFO-per-class networks with two types of regulators: per-flow regulators and interleaved regulators (the latter reshape entire flow aggregates). Such regulators come with a hardware cost that is less for an interleaved regulator than for a per-flow regulator; both can affect the latency bounds in different ways. We analyze the benefits of both types of regulators in partial and full deployments in terms of latency. First, we propose Low-Cost Acyclic Network (LCAN), a new algorithm for finding the optimum number of regulators for breaking all cyclic dependencies. Then, we provide another algorithm, Fixed-Point Total Flow Analysis (FP-TFA), for computing end-to-end delay bounds for general topologies, i.e., with and without cyclic dependencies. An extensive analysis of these proposed algorithms was conducted on generic grid topologies. For these test networks, we find that FP-TFA computes small latency bounds; but, at a medium to high utilization, the benefit of regulators becomes apparent. At high utilization or for high line transmission-rates, a small number of per-flow regulators has an effect on the latency bound larger than a small number of interleaved regulators. Moreover, interleaved regulators need to be placed everywhere in the network to provide noticeable improvements. We validate the applicability of our approaches on a realistic industrial time-sensitive network.**

I. Introduction

During the last decade, the significance of time-sensitive networks has increased in many real-time application areas such as automotive industry [1], avionics [2] [3], and industrial automation [4]. Such networks strive to support, through traffic synchronization [5], scheduling [6] and control [7], deterministic worst-case delay bounds and jitter. The overall architecture of time-sensitive networks [8], [9] enables seamless network configuration and multi-path forwarding to guarantee high reliability and availability levels. But mapping mixed-criticality flows, such as control, audio and best-effort traffic, on these general topologies can induce cyclic dependencies, i.e., the graph of interferences between flow paths can have cycles. The existence of such cyclic dependencies makes the proof of determinism a much more challenging issue and can lead to system instability, i.e., unbounded delays [10], [11].

In this paper, our main concern is guaranteeing deterministic worst-case delay bounds in time-sensitive networks with cyclic dependencies, while minimizing the deployment costs and keeping high-scalability architectures.

Among analytical methods for conducting worst-case performance analysis of networks with cyclic dependencies, only a few techniques have been proposed in the literature, mainly based on network calculus [12]. The high modularity and scalability of such a framework make it particularly efficient for complex communication networks [13]. It has been actually used for the certification of safety-critical networks, such as in avionics [14] [15] and space [16]. The most relevant solutions in this area can broadly be categorized in two main classes: coping with cyclic dependencies [17] [18] or breaking cyclic dependencies [19]–[21]. The former consists mainly in solving a fixed-point problem to compute deterministic delay bounds when satisfying a given system stability condition. This class of solutions offers low deployment costs, but generally leads to overly pessimistic delay bounds, thus limiting the network scalability (size) and resource efficiency (utilization rate). The latter guarantees cycle-free networks by relying on either specific traffic-routing algorithms [19] that add further constraints on the network design, or a full deployment of traffic regulators within all network devices [20], [21] that are more expensive to deploy. Traffic regulators reshape flows by removing the increased burstiness due to interference with other flows, thus a full deployment in every node removes cyclic dependencies in a radical way. These regulators come in two types: per-flow regulator (PFR) (also called "per-flow shaper") and interleaved regulator (IR) (see Section II-B for details). The latter reshapes entire flow aggregates, whereas the former acts on individual flows hence requires per-flow queues. Both types of regulators enable higher scalability and efficiency for such networks but are more complex to deploy than solutions that do not use them or cope solely with cyclic dependencies.

A variant would be to deploy regulators only within a subset of nodes (partial deployment). This solution positively affects the scalability and efficiency of the network, in comparison to existing solutions coping with cyclic dependencies. Moreover, it incurs lower deployment costs and complexity than full deployment. In this paper, we analyze the benefits of both types of regulators in partial or full deployments, in terms of latency. For partial deployment, we also research how to select a subset of nodes where regulators are to be placed.

Contributions: Our main contributions are as follows:

- We propose low-cost acyclic network (LCAN), an algorithm that finds the optimum number of regulators for breaking all cyclic dependencies;
- We provide another algorithm, fixed-point total-flow analysis (FP-TFA), for computing small end-to-end delay bounds for general topologies, i.e., with and without cyclic dependencies. This algorithm is based on the most recent work in network calculus [22] and takes advantage of our improved bound for the burstiness increase within packetizers.
- We conduct an extensive analysis on a generic grid topology and conclude that partial-deployment schemes achieve a good compromise between cost and performance. If the utilization or the transmission rate is high, we find that a small number of per-flow regulators has an effect on latency larger than a small number of interleaved regulators. Moreover, interleaved regulators need to be placed everywhere in the network to provide noticeable improvements;
- We validate the applicability of partial-deployment approaches on a realistic industrial time-sensitive network.

The rest of the paper is organized as follows. In Sections II and III, we present the problem statement and existing solutions in the literature respectively. We detail the main system assumptions and the model in Section IV. We introduce the new algorithm LCAN in Section V and the new analysis approach FP-TFA in Section VI. We evaluate our proposal in comparison to existing solutions in Section VII. Finally, we validate our solution on a realistic industrial time-sensitive network in Section VIII.

II. Problem Statement

We begin this paper by defining the issue of cyclic dependencies and how they represent a challenge.

A. Cyclic Dependencies

Consider a network where flows have fixed paths and are bounded at sources by leaky buckets; such a constraint, with parameters r (the rate) and b (the burstiness, also called "bucket size" or "burst size" [23]), means that the number of bits that the flow can send over a window of duration t is upper bounded by $rt + b$. Whenever such a flow is subject to a bounded but variable delay, the jitter it suffers (difference between the best- and the worst-case delay) increases its burstiness, that is then propagated to the next element in the flow path. This burstiness-propagation effect can lead to

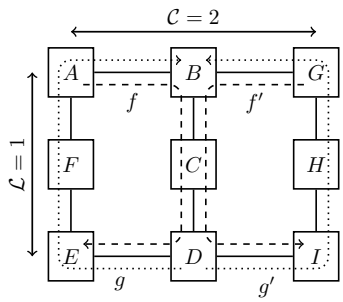

Fig. 1. Toy example of several cyclic dependencies

situations in which the burstiness of the flows have a cyclic dependency on each other. Consider the theoretical session-oriented network presented in Figure 1 where boxes A to I are switches and flows f,g,f' and g' are bounded by leaky-buckets.

When exiting switch A, f suffers a variable delay due to the contention at the output port: f and g interfere, as they both compete to exit A in the direction of B. The worst-case delay depends both on the burstiness of f and g, before A. This worst-case delay implies an increase of the worst-case burstiness of flow f at A's output. The new burstiness is propagated to switch B, where f suffers a delay that increases again its burstiness. At D, the propagated burstiness of f competes with the fresh flow g, thus creating a delay and a burstiness increase for both g and f. From D to A, g continues to have an increased burstiness at each hop. We reach a dependency because the burstiness of g before A depends on the delay within A, and this delay depends on the burstiness of g before A.

A formal definition of cyclic dependencies can be found in [11, Chapter 12], quoted here:

Definition. For a given network, consider its underlying directed graph $G = (V, E)$ defined by: V, the set of vertexes, is the set of output ports in the network. For a, b two output ports in V, (a, b) is a directed edge in E if at least one flow crosses output port a and just after output port b. A cyclic dependency in the network is defined as a cycle in its underlying graph.

When there is no cyclic dependency, we say that the network is "feed-forward". The definition highlights that we do not consider cyclic data-dependencies commonly viewed in the real-time community but rather cyclic dependencies that are induced by the paths of the flows.

The underlying graph of the network in Figure 1 is given in Figure 2 (For any switch, we use the cardinal points for distinguishing its output ports). For example, $(E_{\text{north}}, F_{\text{north}})$ is an edge in the underlying graph because flow g satisfies the above condition. The graph is not acyclic and the network is not feed-forward.

Cyclic dependencies can lead to the global instability of a network. In [10], Andrews proved that for any utilization ratio (even as close to 0 as desired), there exists a first in, first out (FIFO) network with cyclic dependencies in which the delays of the flows are not bounded.

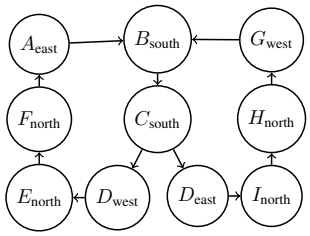

Fig. 2. Underlying graph (defined in Section II-A) of the toy example

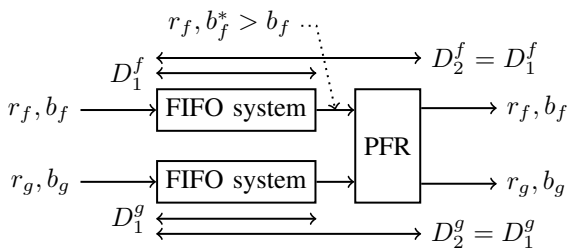

Fig. 3. Shaping-for-free principle (defined in Section II-B) with a PFR

B. Regulators

To prevent cyclic dependencies and the instability they can cause, burst propagation can be blocked by using hardware components called regulators. Regulators are placed just before an output port and reduce the burstiness of the flows by delaying some packets.

Regulators come in two flavours: per-flow regulators (PFRs) and interleaved regulators (IRs). A PFR, with parameter (r, b) for the flow of interest, buffers the data of the flow in a FIFO queue (one per flow) and releases packets as early as possible while ensuring that the output of this flow never has more than $rt + b$ bits over any period of duration t. A well-known implementation is Linux's Token Bucket Filter [23]. The PFR can delay some packets, however, it does not increase the *worst-case* delay [21], [24] ("shaping for free" property). This is illustrated in Figure 3. Any regulated flow is treated differently from the PFR perspective and can come from a different FIFO system. If the PFR is configured so that it resets the burstiness of each flow back to its value at the FIFO-system's input, then the shaping-for-free property holds: the overall worst-case delay D_2^f for flow f equals its worst-case delay D_1^f in the FIFO system (and $D_2^g = D_1^g$ as well).

Interleaved Regulators were introduced by [25] (under the name of "Urgency Based Scheduler", also called "Asynchronous Traffic Shaping" by IEEE TSN) in an effort to reduce the required hardware. An IR has a single FIFO queue for all the flows it regulates (but every flow f has its own regulation parameter (r_f, b_f)). The IR examines only the packet at the head of its FIFO queue and releases it as soon as so doing does not violate the constraint of this flow. Packets of other flows can thus be delayed by the packet at the head of the queue.

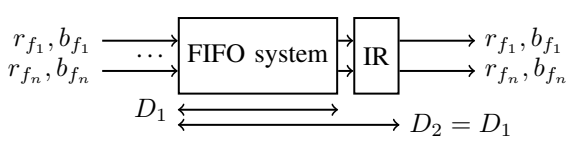

Fig. 4. Shaping-for-free principle with an IR

Nonetheless, an IR that is placed after a FIFO system does not increase the worst-case delay of the FIFO system, as illustrated in Figure 4. Note that this "shaping-for-free" property of the IR holds only if all the flows in the FIFO queue of the IR come from the same previous FIFO system. Also observe that the IR can increase the worst-case delay of a flow when this worst-case delay is less than the worst-case delay across all flows at the node of interest.

Interleaved regulators and per-flow regulators are two behavioral models. Their specific implementation within a network element is still an open discussion among the TSN task group. The ongoing draft IEEE802.1Qcr Asynchronous Traffic Shaping (ATS) specifies the functional requirements for a bridge to behave as per the IR model and, with slight modifications, as per the PFR model. At the time of this writing, this functional description is not yet published and does not imply any corresponding hardware requirement, as several solutions can be envisioned to achieve a same network function. Even if these hardware requirements will probably have a given cost, we keep an abstract definition of costs and focus on the behavioral models.

C. Network Calculus

Network calculus is a mathematical framework initiated by Cruz [26], [27] and then extended in [12], [28]. It uses cumulative arrival functions $R(t)$ [resp $R^*(t)$] which count the amount of bits that have entered [resp. exited] a node between 0 and t. Then the delay and backlog at a node at any time are obtained from the horizontal and vertical distances between R and R^*.

In a real system, R and R^* are unknown, so the framework uses an upper bound on the traffic: α is an arrival curve if $\forall t, s \geq 0, R(t+s) - R(t) \leq \alpha(s)$. Similarly, it uses a lower bound on the service provided by a node to the traffic: β is a service curve is $\forall t, R^*(t) \geq \inf_{s \leq t}(R(s) + \beta(t-s))$.

With these abstractions, network calculus defines an algebra based on the (min,plus) convolution and nodes can then be represented as in traditional system theory, with an input, an output and a transfer function. For instance, if a flow of arrival curve α is serialized on a medium of transmission rate c, the resulting bit stream has an arrival curve $\alpha \otimes \lambda_c$ where $\lambda_c : t \mapsto ct$ and $\otimes$ is the (min,plus) convolution: $\alpha \otimes \lambda_c(t) = \inf_{s \leq t} \alpha(s) + \lambda_c(t-s)$.

One fundamental result of network calculus is the "three bounds theorem" [12], that provides deterministic bounds on the delay and the backlog at a node, together with an upper bound on the burstiness-propagation effect. It can thus be used for providing end-to-end (ETE)-delay bounds if the network is globally stable. Finally, network calculus can be used for studying the effect of regulation on delays.

D. Full and Partial Deployments of Regulators

Regulators (both PFRs and IRs) can be deployed at every node just before an output port [29]. In such a *full deployment*, the burstiness of every flow remains the same along its path; there is no burstiness propagation and the problems caused

by cyclic dependencies are eliminated. Worst-case latencies can be computed using the "shaping-for-free" property and other network calculus results. Note that regulators have a beneficial effect as they avoid burstiness increases, however they have other properties that can negatively affect the latency. Specifically, they affect the serialization and the peak-rate limitation benefits for the aggregate of all flows they serve. For example, if a collection of flows is known to arrive on the same link, then the peak rate limitation imposed by the link can be exploited to compute smaller latency bounds, an effect known as the "line shaping". Such a limitation is no longer true at the output of a PFR or an IR. We expect the effect on latency to be very similar for full deployment of PFR versus IR (but PFRs are more complex than IRs). Our first objective is to evaluate the benefit of full deployments of PFRs or IRs on latency. To this end, we developed FP-TFA, a novel algorithm for computing delay bounds in networks with cyclic dependencies.

As regulators break the propagation of burstiness, an alternative approach is to install a few regulators within the network, such that cyclic dependencies are removed (*partial deployment*). The computation of latency bounds is hence facilitated and can be performed using network calculus. Observe that, here, there is a significant difference between PFRs and IRs. If an IR is used at a point, say B, in the network to restore the burstiness that exists for a set of flows at some point, say A, the entire path from A to B must be globally FIFO for the aggregate traffic of the set of flows. In practice, this requires that A be an upstream neighbour of B. Such a limitation does not exist for a PFR. The partial deployment approach[1] has not yet been studied and raises questions. Is there any benefit on latency ? How can it be quantified in terms of costs, performance, sensitivity ? Is there a significant effect on latency if PFRs are used instead of IRs ? How to select the nodes that should implement regulation ? These questions constitute the second set of objectives of this paper.

III. Related work

Several proposals have been made to handle the problem of cyclic dependencies described in Section II-A.

Mathematical adaptations of the network-calculus framework were proposed to enable the analysis of networks that already have cyclic dependencies. This situation occurs on existing networks with an existing flow mapping. A first space of solution is the fixed-point problem formulation [11, Chapter 12]. More complex approaches rely on global interference equations. We can cite the Charny condition or the pay multiplexing only at convergence points (PMOC) framework [30], [31].

Mathematical approaches take advantage of serialization effects [31], [32] but suffer from the burst-propagation phenomenon that creates cyclic dependencies. They cannot guarantee the stability of the network *a priori* and usually provide pessimistic delay bounds [31].

[1]We consider only partial deployments of PFRs or partial deployments of IRs. We leave partial deployments of mixtures of PFRs and IRs for further study.

TABLE I
Approaches studied in this paper and methods of analysis

	Total deployment	Partial deployment	No regulator
PFR	[24] + [22]	LCAN and FP-TFA	FP-TFA
IR	[21] + [22]	LCAN and FP-TFA	

If cyclic dependencies are anticipated in a network under design, then the designer could chose to avoid them at the first place. This could be done in several ways:

a) By adding routing constraints: Commonly used in industrial networks, this approach benefits from the research on deadlock prohibition in networks on chip (NoC) [33]; turn prohibition is an example for heterogeneous networks [19]. Routing constraints give, however, little latitude to the network architect on the mapping of the flows and face configuration issues in the context of large-scale networks that are at the core of IEEE time-sensitive networking (TSN) and Internet Engineering Task Force (IETF) Detnet working groups. They can also lead to both the surcharge of some resources and to the waste of others [19]. We do not consider this class of solutions.

b) By using regulators: If adding routing constraints is not possible, the designer could choose to add the regulators described in Section II-B. Regulators break the burst propagation hence remove cyclic dependencies. Two types of regulators are considered in this paper: the per-flow regulator (PFR) [24] and the interleaved regulator (IR) [21]. Previous work consider regulators as elements deployed at each hop within the network [29].

Regulators are powerful tools, as they provide some level of control on the arrival curves at their output. However this control is interesting only if the delay they incur can be computed and bounded. In Section II-B, we present the shaping-for-free principle, a fundamental property of regulators. Compared to mathematical approaches, they do not suffer from the burst-propagation phenomenon but do not take advantage of serialization effects, as the burst of each flow is paid at each node.

The right and left columns of Table I were the only two options considered in the previous work on cyclic dependencies and regulators. In both cases, stability requirements, cost requirements and scalability requirements are only met partially. Partial-deployment schemes have not yet been studied and might represent an opportunity for a good compromise between cost, scalability and performance.

IV. System Model

We consider an asynchronous switched network with full-duplex links. We assume that there is one or several classes of traffic and that traffic flows are statically assigned to a class. At every node, all packets of all flows of a given class are processed according to some unspecified scheduling method, in order of arrival (FIFO-per-class). We focus on one of the classes. Notations are presented in Table II.

TABLE II
NOTATIONS

$\mathcal{N}$	The set of all the output ports in the network
$n \in \mathcal{N}$	An output port
$\mathcal{I}$	The set of all the input ports in the network
$\mathcal{I}(n)$	The subset of $\mathcal{I}$ containing the input ports located on the same device as the output port n
c_i	Transmission rate of the line at the input port i
c_n	Transmission rate of the line at the output port n
t_{prop}	Minimum propagation time in any link of the network
f	A flow
$f \ni n$	Flow f is crossing the port n
$f \ni (i,n)$	Flow f is entering via the input i and exiting via the output n
$\alpha_f(n)$	Arrival curve of the flow f before entering the device that contains the port n
r_f	Rate of the flow f
$b_f(n)$	Burst of the flow f before entering the device that contains the port n
$\alpha_f^*(n)$	Arrival curve of the flow f at the output of the scheduler n
$b_f^*(n)$	Burst of the flow f at the output of port n
α_f^0	Initial arrival curve of the flow f, $\alpha_f^0(t) = r_f t + b_f$ for some r_f, b_f
$\alpha_f^R(i,n)$	Regulated arrival curve for f enforced by the regulator (i,n)
β_n	Service curve of the scheduler in port n
$\gamma_{r,b}$	$\gamma_{r,b}(t) = (rt+b)1_{t>0}$, Leaky Bucket arrival curve with rate r and burst b
$\beta_{R,L}$	$\beta_{R,L}(t) = R(t-T)^+$, Rate-Latency service curve with rate R and latency L
δ_T	Variable delay service curve with a maximum delay T
λ_R	$\lambda_R(t) = Rt$, Service curve with a rate R and no latency

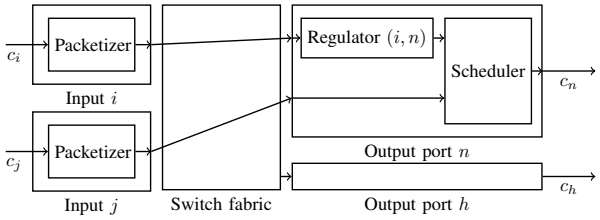

Fig. 5. Device model

A. Device Model

In Figure 5, we present the model of any device in the network. It is made of input ports, output ports, and a switch fabric. We assume that all input ports $\mathcal{I}$ and all output ports $\mathcal{N}$ in the network can be listed. Hence, port indexes i, j, n and h are not relative to the device, but absolute in the network. We assume that the propagation times from device to device are non zero, and denote by t_{prop} the minimum among all such propagation times across the entire network.

Any packet enters the device via one of its input ports, say i. Its bits are received at rate c_i where c_i is the transmission rate of the line connected to input port i. The packet is then stored entirely in a *packetizer*, i.e. a network element that releases the data only once the whole packet has been received. Table lookup and transmission through the switch fabric are then performed and we assume that these two last steps are instantaneous.

Depending on the routing table, the packet might then be transmitted to output port n. If a regulator has been installed for input i and output n (regulator (i,n) in Figure 5), the packet is then processed by the regulator before being made available for the scheduler. If no regulator is installed for that tuple (case of tuple (j,n) in Figure 5), the packet is immediately transmitted to the scheduler.

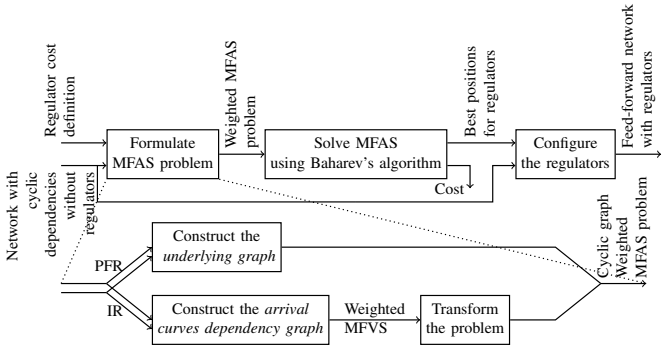

Fig. 6. Overview of the LCAN algorithm

The scheduler implements a FIFO-per-class policy, and we assume its minimum service for the considered class can be lower-bounded by a rate-latency service curve β_{R_n,L_n}, also denoted with β_n. For all $n \in \mathcal{N}$, we assume $R_n \leq c_n$ and $\forall i \in \mathcal{I}(n), R_n \leq c_i$: the service rate is always lower than all the line transmission rates on the device. When a packet is selected by the scheduler for transmission, it is serialized on the transmission line at rate c_n.

B. Flows and Regulators

For any flow f and any output port n, $\alpha_f(n)$ represents the arrival curve of f before entering the device that contains output port n and $\alpha_f^*(n)$ represents its arrival curve at the output of port n. α_f^0 is the initial arrival curve of flow f (at the source). Whenever the flow is processed by a regulator installed at tuple (i,n), we note $\alpha_f^R(i,n)$ the regulated output arrival curve of this regulator for flow f. For a PFR, this arrival curve is the initial arrival curve; for an IR it is the arrival curve computed at the previous hop – see Section V-A for details. The initial arrival curve for any flow f is of the leaky bucket type, i.e of the form $\alpha_f^0(t) = r_f t + b_f$ for some r_f, b_f. We assume that every output port n meets its local-stability requirement as described in [11], i.e. $\sum_{f \ni n} r_f < R_n$.

V. LCAN: CONFIGURING AND POSITIONING REGULATORS WITHIN A NETWORK

In this section, we describe LCAN, an algorithm that computes a partial deployment of either PFRs or IRs with minimal cost, subject to the constraint of breaking all cyclic dependencies. The cost is the sum of the costs of every regulator. The cost of one regulator is configurable and is given by an external function specified by the user. For instance, the cost can be defined according to:

- the number of flows a regulator has to process,
- the properties of the flows a regulator has to process (e.g., rate, burst),
- the size of the device on which the regulator is implemented.

Additionally, constraints can be added to the algorithm to account for industrial requirements. For instance, a device could be out of the operator's control and thus could not implement any regulator; or a device could host a maximum

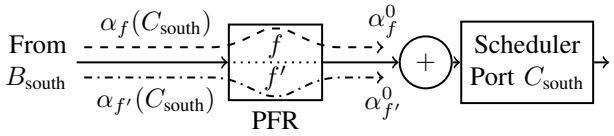

Fig. 7. Output port C_{south} of the network example with a PFR

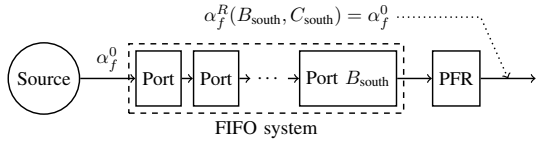

Fig. 8. Meeting of the requirements for the shaping-for-free property with a PFR

number of regulators, etc. As described in Section II-B, these aspects are still under discussion in the real-time community. To anticipate any potential outcome of this discussion, LCAN takes the cost definition as an input. However, in this paper, we are using unitary costs, and if a PFR is placed for a given (input, output) pair, we count it as one regulator, regardless of the number of flows.

An overview of LCAN is presented in Figure 6. It operates in three steps (Figure 6). First, it creates a minimum feedback arc set (MFAS) problem, a well-known graph-theory problem that represents the objective. The output of this step is a directed weighted graph with cycles, the weights representing the configurable costs of the regulators. Second, it uses a state-of-the-art algorithm to solve the MFAS problem. In the last step (not presented hereafter), the network is configured and made ready to be analyzed by any feed-forward method.

A. Graph Construction

To explain how graphs are built by LCAN, we consider again the toy example in Figure 1. Figure 2 presents the underlying graph of this network, as described in Section II-A.

As discussed in the mentioned section, the underlying graph is actually a representation of the burst-propagation effect throughout the network. If the graph is acyclic, the network is feed-forward.

a) LCAN Operations with per-flow regulators: Assume output port C_{south} implements a PFR that regulates the flows coming from B_{south}. We configure it with $\alpha_f^R(B_{\text{south}}, C_{\text{south}}) = \alpha_f^0$ and $\alpha_{f'}^R(B_{\text{south}}, C_{\text{south}}) = \alpha_{f'}^0$: the arrival curves of f and f' at the output of the PFR equal their initial arrival curves at their respective sources. Figure 7 presents the inside of port C_{south}. The flows competing for the scheduler are f and f' with regulated arrival curves α_f^0 and $\alpha_{f'}^0$. Hence, the computation of the delay within the scheduler of port C_{south} neither depends on $\alpha_f(C_{\text{south}})$ nor on $\alpha_{f'}(C_{\text{south}})$. The burst propagation from B_{south} to C_{south} is blocked. This corresponds to removing edge $(B_{\text{south}}, C_{\text{south}})$ from the graph in Figure 2.

The shaping-for-free property described in Section II-B is kept. For the flow f regulated by the PFR $(B_{\text{south}}, C_{\text{south}})$, the FIFO system associated with the regulator is the entire suite of ports from the ingress node of f up to and including the output port B_{south} (Figure 8).

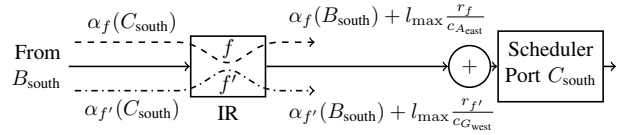

Fig. 9. Output port C_{south} of the network example with an IR

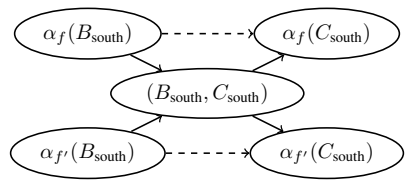

Fig. 10. Fraction of the *arrival curve dependency graph* (defined in Section V-A) for the network example

Finding the optimal positions for the PFRs is now translated into a MFAS problem: the objective is to remove the cycles from the underlying graph by removing the edges with the fewest weight sum. Each removed edge corresponds to a PFR. The weights are equal to the costs of the regulators. In the case of the toy example in Figure 1, if the weights all equal 1, then removing edge $(B_{\text{south}}, C_{\text{south}})$ from the underlying graph in Figure 2 is the optimal solution to make it acyclic.

b) LCAN Operations with interleaved regulators: Assume that we implement an IR in place of the PFR (Figure 9). We configure the IR such that $\alpha_f^R(B_{\text{south}}, C_{\text{south}}) = \alpha_f(B_{\text{south}}) + l_{\max}\frac{r_f}{c_{A_{\text{east}}}}$ and $\alpha_{f'}^R(B_{\text{south}}, C_{\text{south}}) = \alpha_{f'}(B_{\text{south}}) + l_{\max}\frac{r_{f'}}{c_{G_{\text{west}}}}$. The flows f and f' exiting the regulator do not retrieve their fresh arrival curves α_f^0 and $\alpha_{f'}^0$ rather they retrieve those they had after the packetizers that precede the previous output port (Figure 11). The packetization effect is computed as per Section VI-A. The regulator removed the effect of the last output port scheduler (port B_{south}) on the arrival curves of f and f'.

$\alpha_f^R(B_{\text{south}}, C_{\text{south}})$ and $\alpha_{f'}^R(B_{\text{south}}, C_{\text{south}})$ are not entirely independent from the past ports, which means the action of the IR on the network cannot be modeled by removing edge $(B_{\text{south}}, C_{\text{south}})$ from the graph Figure 2. We need to define a new graph that we call the *arrival curve dependency graph*. A fraction of it is given for the example in Figure 10, where we focus on the link between B and C. The graph is made of two types of nodes:

- The *state* nodes: Node $\alpha_f(B_{\text{south}})$ represents the arrival curve of flow f before entering the device that holds output port B_{south}, *i.e.* before entering device B.
- The *contention* nodes: Node $(B_{\text{south}}, C_{\text{south}})$ represents the

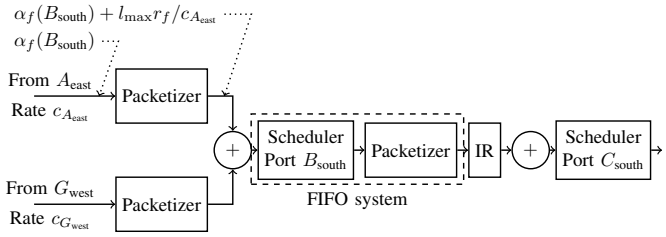

Fig. 11. Meeting of the requirements for the shaping-for-free property with an IR

propagation of the burst due to the contention on the scheduler B_{south} to the next port in the path of the flows, *i.e.* port C_{south}.

It also includes three types of edges:

- The dashed edges: $\alpha_f(B_{\text{south}}) \rightarrow \alpha_f(C_{\text{south}})$ represents the propagation of the arrival curve of f from B_{south} to C_{south}.
- The edge $\alpha_f(B_{\text{south}}) \rightarrow (B_{\text{south}}, C_{\text{south}})$ represents the contention that takes place in B_{south}, in which f participates, and the propagation of the effects of this contention in terms of burstiness increase to the next output port: C_{south}.
- Finally, edge $(B_{\text{south}}, C_{\text{south}}) \rightarrow \alpha_f(C_{\text{south}})$ represents the propagation of these effects on the arrival curve of f before port C_{south}.

Placing an IR between ports B_{south} and C_{south} removes the influence of the contention in the scheduler of port B_{south} on all the arrival curves, before the scheduler of port C_{south}. This corresponds to removing node $(B_{\text{south}}, C_{\text{south}})$ (and all its edges) from the graph in Figure 10. The arrival curves of f and f' at the output of the IR continue to depend on their previous arrival curves: this is captured by not removing the *state nodes* and dashed edges in Figure 10.

The shaping-for-free property described in Section II-B is kept. For the flow aggregate regulated by the IR $(B_{\text{south}}, C_{\text{south}})$, the FIFO system associated with the regulator in the direct previous scheduler at output port B_{south}, after the packetizers (Figure 11). This system is FIFO for the aggregate $\{f, f'\}$.

Finding the optimal positions for the IRs is now translated into a minimum feedback vertex set (MFVS) problem: the objective is to remove the cycles from the *arrival curve dependency graph* by removing the contention nodes with the fewest total weight. Each removed contention node corresponds to an IR.

In graph theory, MFVS and MFAS are equivalent problems [34] and MFVS can easily be transformed into a MFAS with minor graph manipulations. This step is captured in the *Transform the problem* box in Figure 6.

B. Solving the MFAS Problem

Finding a MFAS is a well-known NP-hard problem in graph theory [35]. We use Baharev's algorithm [34]: it relies on a cover-set problem formulation with a lazy constraints generation. It provides an optimum and is well suited for graphs containing up to one million cycles, a reasonable limit for industrial cases. LCAN invokes Baharev's algorithm to remove the cycles in the graphs computed in the previous subsection. Lastly, LCAN configures the regulators and updates the network representation.

VI. FP-TFA: A NOVEL ALGORITHM FOR SMALL DELAY BOUNDS IN NETWORKS WITH CYCLIC DEPENDENCIES

LCAN can be used to achieve partial deployment of regulators. To compare its performance against the no-deployment approach (right column of Table I), we need an algorithm for

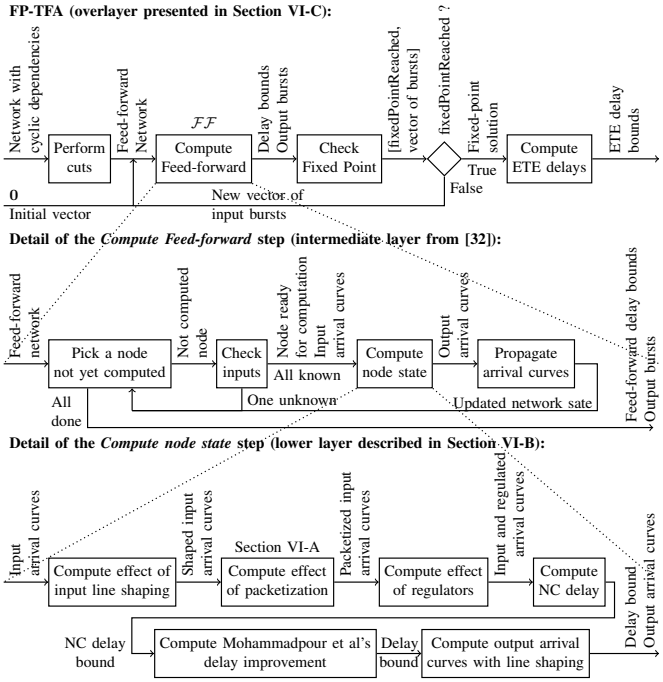

Fig. 12. Overview of the FP-TFA algorithm

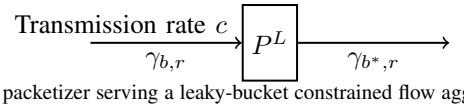

Fig. 13. A packetizer serving a leaky-bucket constrained flow aggregate with an input transmission line of rate c

providing small delay bounds in the latter case. We present fixed-point total-flow analysis (FP-TFA), a novel algorithm for computing delay bounds in networks with cyclic dependencies. Up to Subsection VI-D, we assume that no regulator has been deployed in the network.

An overview of FP-TFA is available in Figure 12. FP-TFA is primary based on TFA++ [32] and provides

- a new result on the effect of packetizers,
- the computation of a tighter delay bound within nodes by implementing the aforementioned result, as well as recent work in network calculus [22], and
- an extension of the algorithm to topologies with cyclic dependencies by using an iterative fixed point described in [11, Chap. 12].

A. A Novel Result on Packetizers

The input ports in our model contain packetizers (Figure 5). To compute delays within nodes, we need to assess both their delay and their effect on the aggregate arrival curves.

Previous results on packetizers [12] concluded that they do not directly participate in the ETE-delay bound of flows, but that they can increase the burstiness of the flow aggregate. This burstiness increase is bounded by $l_{\max}$, the maximum packet size of all the flows. In this subsection, we provide a tighter bound when the packetizer is connected to an input line of fixed transmission rate.

Theorem 1. *If a packetizer P^L is placed on a line of fixed transmission rate c and if it serves a leaky-bucket constrained*

Fig. 14. Timeline of the reception of several packets at packetizer's input

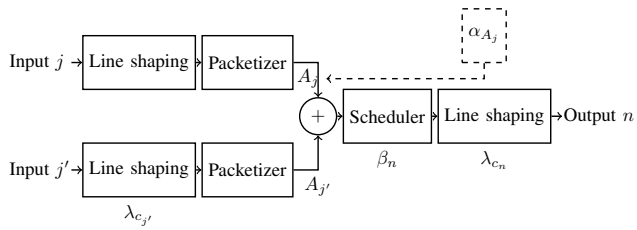

Fig. 15. Detailed model for the computation of the delay bound within a node

flow aggregate $\gamma_{b,r}$, then an arrival curve for the aggregate at its output is the leaky-bucket arrival curve $\gamma_{b^*,r}$ with $b^* = b + \frac{r}{c} l_{\max}$, where $l_{\max}$ is the maximum packet length of the flow aggregate.

Proof. We note R the cumulative arrival function of the aggregate at packetizer's input. For a given packet q, we note A_q the arrival time of the first bit of the packet in P^L. We note D_q the arrival time of the last bit of packet q and the subsequent release of the entire packet q from P^L. The transmission rate of the medium is c, therefore $D_q - A_q = l_q/c$, with l_q the size of packet q.

Let us take any two packet indices $q \leq s$ (Figure 14). By definition of the maximum arrival curve $\gamma_{b,r}$ for the input aggregate, the amount of bits received during time interval $D_s - A_q$ is bounded by $\gamma_{b,r}(D_s - A_q)$, i.e.

$$l_q + \ldots + l_s \leq r \cdot (D_s - A_q) + b$$

Using the previous relation for $D_q - A_q$, we obtain:

$$l_q + \ldots + l_s \leq (D_s - D_q) r + b + r l_{\max}/c \qquad (1)$$

Equation (1) is the max-plus representation of a packetized flow aggregate constrained by an arrival curve $\gamma_{b^*,r}$ with $b^* = b + r\frac{l_{\max}}{c}$ [12]. □

B. Delay within a Node

In this subsection, we combine the aforementioned burstiness improvement with recent work in network calculus [22] and present the delay computations within the lowest layer in Figure 12. We are interested in the delay suffered by the flows within output port n of the device model in Figure 5. The computational model associated is presented in Figure 15, where j and j' are two of several input ports without any regulator.

According to the total-flow analysis (TFA)++ model [32], neither the input shaping nor the output shaping participate in the ETE delay, under the assumption that the medium transmission-rate is higher than the service rate of the scheduler ($c_n \geq R_n$). We also mention, in Section VI-A, that packetizers do not directly participate in the ETE delay but they can increase the burstiness.

In conclusion, only the scheduler in Figure 15 participates in the ETE-delay bound of the flows. To compute a delay bound, we need to obtain the arrival curve of the flow aggregate at its input.

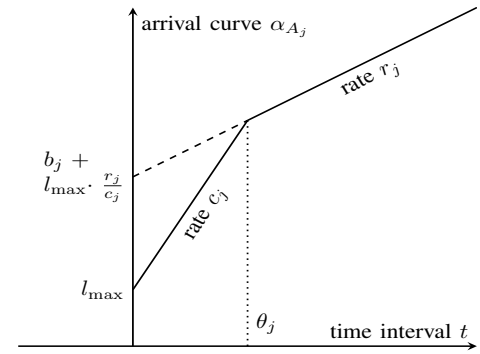

Fig. 16. Arrival curve α_{A_j} for one unregulated input j

A flow aggregate of arrival curve $\sum_{f \ni (j,n)} \alpha_f(n)$ enters input port j and is first submitted to the shaping created by the input transmission line, as stated in TFA++ [32]. The packetizer input arrival-curve hence equals $\lambda_{c_j} \otimes \sum_{f \ni (j,n)} \alpha_f(n)$ and is the convolution of two leaky-bucket arrival curves. We apply to each component the impact of the packetizer as in Therorem 1. The aggregate arrival curve at location A_j in Figure 15 equals

$$\alpha_{A_j} = \left(\lambda_{c_j} + l_{\max}\right) \otimes \left(\sum_{f \ni (j,n)} \alpha_f(n) + l_{\max} \frac{\sum_{f \ni (j,n)} r_f}{c_j} \right)$$

where $\otimes$ represents the min-plus convolution [12].

If we sum the incoming aggregate arrival curves from all the input ports, an arrival curve of the aggregate that the scheduler has to serve is

$$\alpha_{\text{sched. } n} = \sum_{j \in \mathcal{I}(n)} \left(\underbrace{\left(\lambda_{c_j} + l_{\max}\right) \otimes \left(\sum_{f \ni (j,n)} \alpha_f(n) + l_{\max} \frac{\sum_{f \ni (j,n)} r_f}{c_j} \right)}_{\alpha_{A_j}} \right) \qquad (2)$$

With the aggregate, we compute the network calculus delay-bound as the maximal horizontal distance between $\alpha_{\text{sched. } n}$ and β_n. In the following part of the subsection, we address where and how this horizontal distance is computed. Because the flows have leaky-bucket initial arrival curves, all the arrival curves $\alpha_f(n)$ are leaky-bucket arrival curves.

Figure 16 presents the shape of α_{A_j}, a piecewise linear function with $b_j = \sum_{f \ni (j,n)} b_f(n)$ [resp $r_j = \sum_{f \ni (j,n)} r_f(n)$] the total burstiness [resp. the total rate] of the flows $f \ni (j,n)$. We note $\theta_j = \frac{b_j + l_{\max}(r_j/c_j - 1)}{c_j - r_j}$ the time value of the intersection of both slopes.

Next step of the algorithm is to add all the A_j terms for all input j such that $\{f \ni (j,n)\}$ is non-empty. Call $\#J$ the number of such inputs. The sum outputs the piecewise-linear function presented in Figure 17, with m the input index such that $\theta_m = \max_j \theta_j$, $r_{\text{tot}} = \sum_j r_j$, the total rate of the inputs and $b_{\text{tot}} = \sum_j b_j$, the total burstiness of the inputs, without the line shaping or the packetization effects.

The network calculus delay is finally computed. The highest horizontal distance is reached on θ_m because $r_{\text{tot}} \leq R_n$ (local-stability assumption), whereas $r_{\text{tot}} - r_m + c_m \geq c_m \geq R_n$

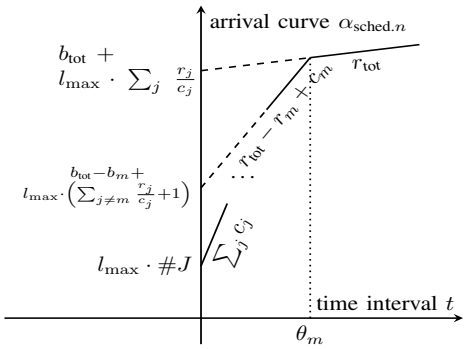

Fig. 17. Arrival curve $\sum_j \alpha_{A_j}$ for all unregulated inputs $\{j\}$

(assumption that the service rate is lower than all transmission rates). With the above notations, an upper bound on the delay of the flows in output port n equals

$$D_n = \frac{b_{\text{tot}} + l_{\max} \cdot \sum_j \frac{r_j}{c_j}}{R_n} + T_n + \theta_m \cdot \left(\frac{r_{\text{tot}}}{R_n} - 1\right) \quad (3)$$

with R_n the service rate of the scheduler n and T_n its latency.

The algorithm then applies the delay bound improvement of Mohammadpour et al., achieved when an output port n is followed by a line with transmission rate c_n [22]: $D_n^* = D_n - l_{\min}\left(\frac{1}{R_n} - \frac{1}{c_n}\right)$ is also a bound on the delay suffered by the flows within port n, where $l_{\min}$ is the minimal packet size of the flows.

The last step of the algorithm consists in computing the output arrival curve for each individual flow by taking into account the shaping effect of the output transmission rate c_n. This step is performed following the results in [32]: for a given flow f, its output arrival curve $\alpha_f^*(n)$ equals

$$\alpha_f^*(n) = (\alpha_f(n) \otimes \lambda_{c_n}) \oslash \delta_{D_n^*} \quad (4)$$

where $\oslash$ represents the min-plus deconvolution [12], $\delta_{D_n^*}$ is the service curve with no minimum service before D_n^* and infinite service after. The retrieved arrival curves are leaky-bucket functions, with same rates as at the sources, but with larger bursts. Before propagation, we apply a ceiling function to the bursts. Hence, they are expressed by integer values of bits.

C. Delay Bounds in Networks with Cyclic Dependencies

Combining the two low layers of FP-TFA provides an analysis tool for feed-forward networks. For a network with cyclic dependencies, the main idea is to virtually perform cuts in the topology so as to make it acyclic [11]. A cut is here a separation between two output ports.

Cuts can be selected freely as long as the remaining network is feed-forward. However, to keep the following steps tractable, having as few cuts as possible is an interesting strategy. Hence, the identification of cuts of Figure 12 can be performed using LCAN. In the toy example (Figure 2), a cut between B_{south} and C_{south}, splitting f and f', is sufficient.

The resulting virtual network is feed-forward. Given the knowledge of the bursts after the cuts (vector $[b_f(C_{\text{south}}), b_{f'}(C_{\text{south}})]$ in the toy example), the iterative application of the method in the previous subsection provides an algorithm, $\mathcal{FF}$, to compute a bound for the bursts before the cuts (vector $[b_f^*(B_{\text{south}}), b_{f'}^*(B_{\text{south}})]$).

If the real network, with no cut, is stable, then for every cut tuple (n, h) and every split flow f there exists a lowest possible bound on the burstiness of f after the cut (before h). Thus, the vector $\boldsymbol{b}$ of these lowest burst bounds must verify $\mathcal{FF}(\boldsymbol{b}) \geq \boldsymbol{b}$. By Tarski's fixed-point theorem [36] and the monotonicity of $\mathcal{FF}$, it follows that $\boldsymbol{b}$ is upper-bounded by the (possibly infinite) largest fixed point of $\mathcal{FF}$. In Theorem 2, we prove a stronger result: for any non-negative and finite fixed point $\overline{\boldsymbol{b}}$ of $\mathcal{FF}$ (i.e., $\mathcal{FF}(\overline{\boldsymbol{b}}) = \overline{\boldsymbol{b}}$) and if the real network is initially empty, then the real network is stable and $\boldsymbol{b} \leq \overline{\boldsymbol{b}}$, i.e., the fixed point $\overline{\boldsymbol{b}}$ is a valid bound for the bursts of the flows at the cuts.

To find such a fixed point, FP-TFA iterates the feed-forward algorithm $\mathcal{FF}$, starting with the empty vector $\mathbf{0}$ (Figure 12). By Tarski's fixed-point theorem [36], this gives the smallest fixed point. After each call to $\mathcal{FF}$, Algorithm 1 is called to check if the fixed point is reached for the current iteration. This algorithm compares the input and output vectors. For the fixed point to be reached, strict equality must be achieved for each term of the vectors, since burst values are integer (in bits). Algorithm 1 outputs a boolean and the tentative fixed point. The boolean is used in Figure 12 to decide if another iteration must be taken. If so, the output bursts are used as input to a new call to $\mathcal{FF}$.

Algorithm 1 Detail of the "Check Fixed Point" block in Figure 12

Require: $\boldsymbol{b}_{\text{in}}$ is the input vector of $\mathcal{FF}$. It contains the burst after each cut, for each flow. n is its size. $\boldsymbol{b}_{\text{out}}$ is the result $\mathcal{FF}(\boldsymbol{b}_{\text{in}})$.

```
procedure CHECKFIXEDPOINT(b_in, b_out)
    b_new ← empty vector size n.
    fixedPointReached ← True
    for i = 1, ..., n do
        fixedPointReached ← fixedPointReached and b_in,i = b_out,i
        b_new,i ← b_out,i
    end for
    return [fixedPointReached, b_new]
end procedure
```

Theorem 2. *If the network with cyclic dependencies is empty at $t = 0$, then any nonnegative fixed point, i.e. a vector $\overline{\boldsymbol{b}}$ such that $\mathcal{FF}(\overline{\boldsymbol{b}}) = \overline{\boldsymbol{b}}$, is a valid burst bound for the network with cyclic dependencies at the cuts. If $\mathcal{FF}$ has a finite nonnegative fixed point, then the network is stable.*

Proof. Consider the view of the cyclic network in Figure 18 where points U and W are located respectively after and before the cuts. The network between U and W is feed-forward. We fix some θ such that $0 < \theta < t_{\text{prop}}$ and we consider V, the point that is exactly θ seconds before W. This point is on the same link as W because t_{prop} is the minimal propagation delay of links. We consider the true network, i.e.

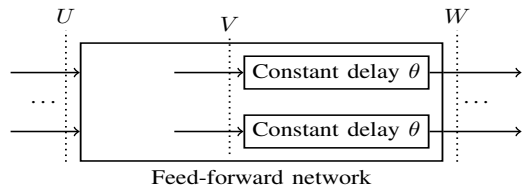

Fig. 18. Illustration of the proof principle. The network is feed-forward between U and W, and V is exactly θ seconds before W.

with U and W connected together. In the rest of the proof, take any $\tau \geq 0$.

For $M = U, V, W$ call $\boldsymbol{R}_M$ [resp. $\boldsymbol{R}_M^\tau(t)$] the vector of cumulative arrival functions [resp. stopped at time τ i.e., $\boldsymbol{R}_M^\tau(t) = \min(\boldsymbol{R}_M(t), \boldsymbol{R}_M(\tau))$]. Also for $M = V, W$ call $\boldsymbol{R}_M'^\tau$ the vector of cumulative arrival functions that are obtained at points V, W when the inputs at U are stopped at time τ, i.e. $\boldsymbol{R}_M'^\tau(t) = \min(\boldsymbol{R}_M(t), \boldsymbol{R}_U(\tau))$. Last, call $\boldsymbol{b}_M^\tau$ and $\boldsymbol{b}_M'^\tau$ the corresponding best burst bounds. For example, for any component i of vector $\boldsymbol{b}_M'^\tau$, $b_{M,i}'^\tau = \sup_{t' \geq t}(R_{M,i}'^\tau(t') - R_{M,i}'^\tau(t) - r(t'-t))$, where r is the leaky-bucket rate of the flow corresponding to component i.

Lemma 1. *At point V, $\boldsymbol{b}_V^\tau \leq \boldsymbol{b}_V'^\tau$*

Proof of Lemma 1. Note that for $t \leq \tau, \boldsymbol{R}_M^\tau(t) = \boldsymbol{R}_M'^\tau(t)$ and for $t > \tau, \boldsymbol{R}_M^\tau(t)$ is a constant. The result is thus obtained by splitting the sup of the definition of $\boldsymbol{b}_V'^\tau$, $\boldsymbol{b}_V^\tau$ into the three options: either $t, t' \leq \tau$, or $t \leq \tau, t' > \tau$ or $t, t' > \tau$. □

As $\mathcal{FF}$ computes a bound on the output bursts for a given input, we know that $\boldsymbol{b}_W'^\tau \leq \mathcal{FF}(\boldsymbol{b}_U^\tau)$. As the delay between V and W is constant equal to θ, a change of variable $(s, s') \leftarrow (t+\theta, t'+\theta)$ in the definition of $\boldsymbol{b}_V'^\tau$ gives $\boldsymbol{b}_V'^\tau = \boldsymbol{b}_W'^\tau \leq \mathcal{FF}(\boldsymbol{b}_U^\tau)$. Using Lemma 1, we get:

$$\forall \tau \geq 0, \boldsymbol{b}_V^\tau \leq \mathcal{FF}(\boldsymbol{b}_U^\tau) \tag{5}$$

Using $\boldsymbol{R}_W(t+\tau) = \boldsymbol{R}_V(t)$ for any $t \geq 0$, we obtain $\boldsymbol{b}_W^{\tau+\theta} = \boldsymbol{b}_V^\tau$. Also, as W and U are connected together, Equation 5 gives $\boldsymbol{b}_U^{\tau+\theta} = \boldsymbol{b}_W^{\tau+\theta} = \boldsymbol{b}_V^\tau \leq \mathcal{FF}(\boldsymbol{b}_U^\tau)$. Apply this to $\tau = k\theta$ for $k \in \mathbb{N}$ and obtain:

$$\forall k \in \mathbb{N}, \boldsymbol{b}_U^{(k+1)\theta} \leq \mathcal{FF}(\boldsymbol{b}_U^{k\theta}) \tag{6}$$

The network is empty at $t = 0$ so $\boldsymbol{b}_U^0 = \boldsymbol{0} \leq \overline{\boldsymbol{b}}$. Now $\mathcal{FF}$ can be assumed to be wide-sense increasing as per [11, Chap. 12]. By monotonicity of $\mathcal{FF}$, the fact that $\mathcal{FF}(\overline{\boldsymbol{b}}) = \overline{\boldsymbol{b}}$ and a simple induction argument, it follows that $\boldsymbol{b}_U^{k\theta} \leq \overline{\boldsymbol{b}}$ for all $k \in \mathbb{N}$. For any $\tau \in [0, +\infty)$, we have $\boldsymbol{b}_U^\tau \leq \boldsymbol{b}_U^{k\theta}$ with $k = \lceil \frac{\tau}{\theta} \rceil$, thus $\boldsymbol{b}_U^\tau \leq \overline{\boldsymbol{b}}$ for all $\tau \geq 0$. Now $\boldsymbol{b}_U = \sup_{\tau \geq 0} \boldsymbol{b}_U^\tau$, thus $\overline{\boldsymbol{b}}$ is a finite bound for $\boldsymbol{b}_U$ and the network is stable. □

D. FP-TFA Adaptations for Networks with Regulators

We group LCAN and FP-TFA together. Our objective is to use FP-TFA on topologies with a partial regulator deployment.

If the deployment has been performed based on LCAN recommendations, the network will be feed-forward and the upper stage of FP-TFA will not be required. In all cases, slight modifications of the lowest stage have to be carried out to account for the presence of regulators.

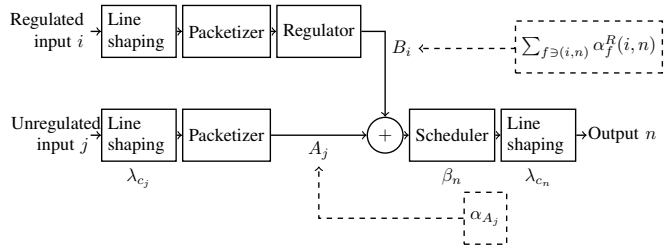

Fig. 19. Detailed model for the computation of the delay bound within a node with partial regulator deployment

Figure 19 presents the modified computational model when some inputs (i in the Figure) hold a regulator placed after the packetizer (as per the device model presented in Figure 5). The regulator element in the model does not participate in the ETE-delay bound, as LCAN ensures the shaping-for-free property is kept (Section V).

An aggregate arrival curve at location B_i, the output of the regulator for regulated input i, can directly be obtained by summing all the configured arrival curves $\alpha_f^R(i,n)$ for any flow $f \ni (i,n)$. Each individual arrival curve being a leaky-bucket arrival curve, the sum is also a leaky-bucket arrival curve. Call r_i its rate and b_i its burstiness.

With the same previous notations, we redefine b_{tot} and r_{tot} by $b_{\text{tot}} = \sum_{j \text{ unregulated}} b_j + \sum_{i \text{ regulated}} b_i$ and $r_{\text{tot}} = \sum_{j \text{ unregulated}} r_j + \sum_{i \text{ regulated}} r_i$. Then, the delay within a node with partial regulator deployment equals

$$D_n = \frac{b_{\text{tot}} + l_{\max} \cdot \sum_{j \text{ unregulated}} \frac{r_j}{c_j}}{R_n} + T_n + \theta_m \cdot \left(\frac{r_{\text{tot}}}{R_n} - 1 \right) \tag{7}$$

The remaining steps of the algorithm are performed as per Figure 12.

VII. Synthetic Use-Case

We evaluate the performance of the five approaches: the no-deployment approach, the partial-deployment approaches using either PFRs or IRs with a cost of 1 for every regulator, and the full-deployment approaches using either IRs or PFRs.

A. The Grid Topology

We consider the toy example, in Figure 1, to be two columns and one line of a basic cell. By operating axial symmetries on axes (G, I) or (E, I), we can control the size of the network. We keep the same path length for all flows. We note $\mathcal{L}$ [resp. $\mathcal{C}$] the number of rows [resp. columns].

We consider high-priority flows with the same leaky-bucket initial arrival curve $\gamma_{b_0,r}$ and constant packet size l. The network is homogeneous, each port provides the same rate-latency service curve $\beta_{R,L}$. For any $\mathcal{L} \geq 2$, the maximum number of flows per link equals 4, we thus note $a = 4r/R$ the network load; c is the transmission rate of the lines.

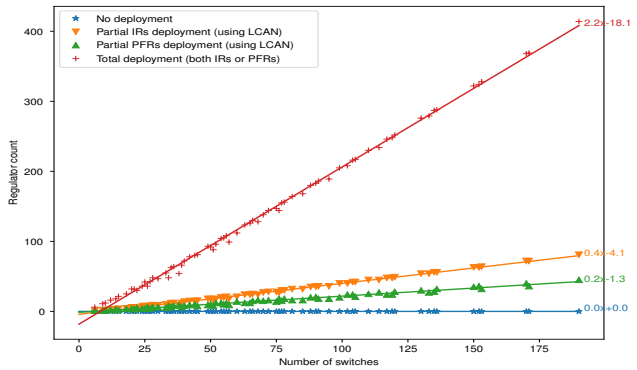

Fig. 20. Number of regulators versus the grid network size for the different approaches.

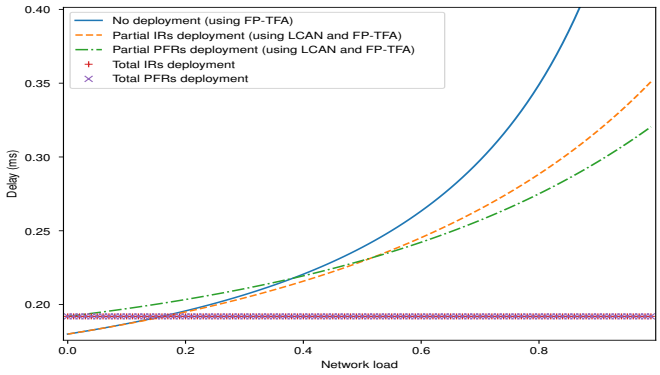

Fig. 21. ETE delay bound of the flows on the grid versus the network load. Size $\mathcal{L} = \mathcal{C} = 8$, initial burstiness $b_0 = 12\mathrm{E}3$bits, service rate $R = 1\mathrm{E}9$bits/s, service latency $T = 1.2\mathrm{E}{-5}$s, packet length $l = 12\mathrm{E}3$bits, line shaping $c/R = 1$.

B. Number of Placed Regulators

Figure 20 presents the number of regulators used versus the number of switches in the network. Compared to the full deployment approach, the IR partial deployment reduces the number of required regulators by 81% and the PFR partial deployment by 89%. Figure 20 shows that cyclic dependencies can be removed with very few regulators within the network.

C. Latency Bounds with Respect to the Utilization

We evaluate the delay bounds versus the network load. To do so, we use the $(\mathcal{C} = 8, \mathcal{L} = 8)$ grid configuration (corresponding to 153 switches). ETE delay bounds are presented in Figure 21.

The full-deployment scheme shows a performance penalty compare to the other approaches when the utilization is low. Up to a utilization threshold $a \approx 15\%$, the best results are obtained without any regulator. At high utilization, the best bound is obtained with the full-deployment schemes, either with PFRs or IRs and both obtain the same delay bound.

Partial-deployment approaches are never the best ones.

D. Effect of the Network Size

Figure 22 presents the ETE delay bound of the flows as a function of the network size. We observe that each approach obtains a delay bound independent from the network size after approximately 60 switches. Note that the lengths of flow paths

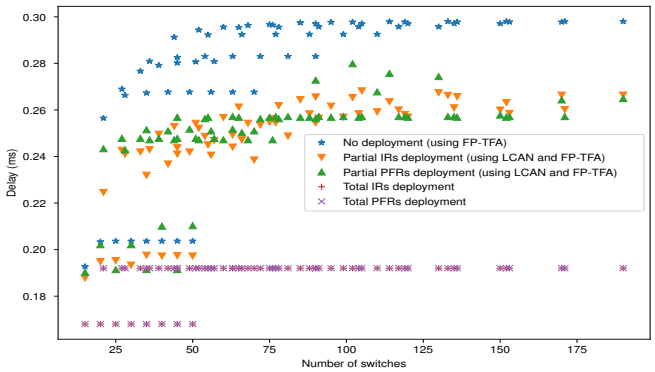

Fig. 22. ETE delay bound of the flows versus the grid size. Initial burstiness $b_0 = 12\mathrm{E}3$bits, utilization $a = 0.7$, service rate $R = 1\mathrm{E}9$bits/s, service latency $T = 1.2\mathrm{E}{-5}$s, packet size $l = 12\mathrm{E}3$bits, line shaping $c/R = 1$.

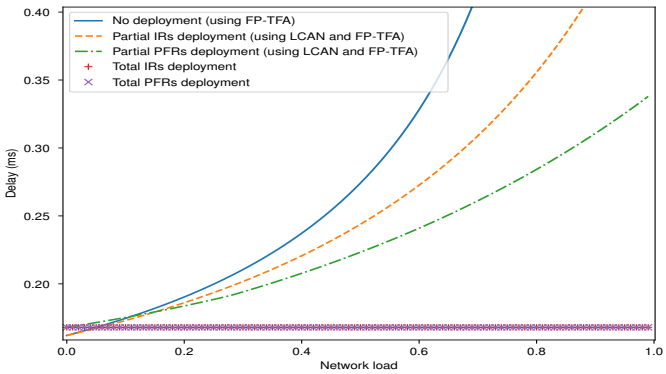

Fig. 23. ETE delay bound of the flows versus the network load with line shaping $c/R = 2$. Size $\mathcal{L} = \mathcal{C} = 8$, initial burstiness $b_0 = 12\mathrm{E}3$bits, service rate $R = 1\mathrm{E}9$bits/s, service latency $T = 1.2\mathrm{E}{-5}$s, packet length $l = 12\mathrm{E}3$bits.

do not depend on the network size. The network complexity does not influence the bound obtained by FP-TFA. The values obtained for large sizes is consistent with the ones obtained for $a = 0.7$ in Figure 21. It emphasizes that the previous observations are independent from the network size.

E. Effect of the Line Shaping

In Figure 23 we set the ratio line rate / scheduler guaranteed rate to $c/R = 2$. This diminishes "line shaping", which is the beneficial effect on latency of the bit-by-bit serialization of the packets when transmitted on the line.

The value of the utilization threshold is reduced (down to $a \approx 5\%$) due to the combination of two effects: On one hand, all partial- and no-deployment approaches have a performance worse than with $c/R = 1$. On the other hand, the delay bound of the full-deployment approach is decreased because it is only affected by Mohammadpour et al.'s delay-bound improvement. This improvement has a more positive effect when the transmission rate c is higher.

We also noted that the no-deployment approaches show some instability. Complementary tests showed that, without regulators, FP-TFA is no longer able to obtain finite latency bounds for the whole utilization spectrum (from 0 to 1) when $c/R \geq 3$.

Stability is retrieved for any partial deployment. However, the partial deployment of IRs shows very large bounds, even

TABLE III
TRADE-OFF OF THE APPROACHES

Criteria + = good properties, - = bad properties	Total (IR or PFR)	Partial PFR	Partial IR	No deployment
Regulator count	-	+	∼	++
Delay bound at low utilization	-	∼	+	++
Delay bound at high utilization	++	∼	∼	-
Sensitivity w/r utilization	++	∼	∼	-
Sensitivity w/r trans. rate	++	-	∼	- -

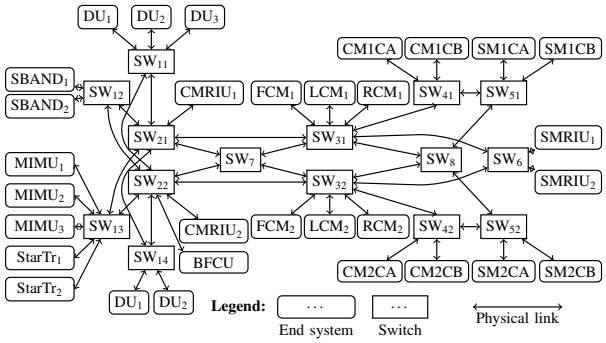

Fig. 24. Physical topology of the Orion CEV network. From [37].

though LCAN places more IRs than PFRs. With fewer PFRs, we achieve a better performance. At a high utilization or for a high transmission rate, IRs need to be placed everywhere to provide noticeable improvements.

F. Conclusions of Synthetic Use-Case

Partial deployments never provide the best delay bounds. However, as summarized in Table III, they provide an interesting compromise between performance, stability, and number of hardware elements.

The partial deployment of IRs is an interesting option only at low utilization and transmission rates. When one of these values increases, this approach shows performance worse than partial deployment of PFRs, and IRs need to be placed everywhere to provide noticeable improvements.

VIII. REPRESENTATIVE INDUSTRIAL USE-CASE

We show the applicability of the different approaches and our algorithms on a representative industrial case. We consider the Orion crew exploration vehicule (CEV) network. Its architecture is detailed in [38, p.328] and relies on TT-Ethernet.

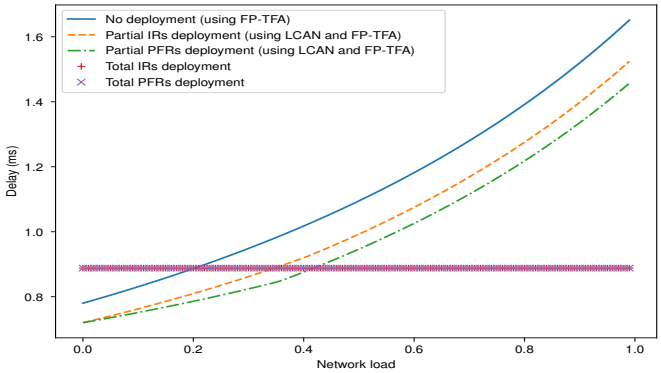

Fig. 25. Highest ETE delay bound of the flows within Orion versus the network usage. Initial burstiness $b_0 = 12\text{E}3$bits, service rate $R = 1\text{E}9$bits/s, service latency $T = 1.2\text{E}{-5}$s, packet length $l = 12\text{E}3$bits, line shaping $c/R = 1$.

We study it, however, in an asynchronous setting.The physical topology presented in Figure 24 is retrieved from [37]. We map on it 119 multicast flows of the TSN control-data traffic (CDT) class by using an algorithm that optimizes link utilization. This mapping creates a total of $293'912$ cyclic dependencies in the network. We assume that each output port provides the same rate-latency service curve and, as we consider the highest priority, we can assume that the service rate equals the transmission rate (case $c/R = 1$).

In this industrial case, a full-deployment approach would require a total of 249 regulators within the network. Our LCAN algorithm, configured with a cost of 1 for each regulator, is able to remove all cyclic dependencies with only either 14 IRs or 9 PFRs. This highlights that substantial cost savings could be achieved in a real industrial case.

Figure 25 shows the highest delay bound among all flows. We note that the no-regulator approach FP-TFA is no longer the best one at low utilization. This emphasizes the potential benefit of regulators in heterogeneous networks. Partial deployment is here an attractive solution up to a utilization of 40%.

IX. CONCLUSION AND FUTURE WORK

We have developed a tool, FP-TFA, for computing latency bounds in time sensitive networks with cyclic dependencies. We have also developed an algorithm, LCAN, for placing a set of PFRs [resp. IRs] of minimal cost such that cyclic dependencies are removed. The combination of FP-TFA and LCAN has been used in synthetic and industrial settings to evaluate the effect on latency of partial or full deployments of either type of regulator. Our analysis shows that when the effect of line shaping is large, FP-TFA computes low latency bounds for the configurations without shapers for a small utilization, after which there is a benefit in deploying regulators. Partial deployment of PFRs improves the latency bounds in the region of medium utilization or at high line transmission-rates while decreasing the deployment costs, with reference to full deployment solutions.

These conclusions, however, do not take into account the hardware-cost difference between a PFR and a IR. The latter probably uses fewer queues than the former. We plan to discuss with manufacturers to create a representative cost function able to capture these differences. Such a cost function could then be used to support deployments of mix of per-flow and interleaved regulators. Testing LCAN on other type of topologies constitutes another field of future work. We plan to discuss with industrials to create representative yet flexible topologies. LCAN's behavior relies mostly on cyclic dependencies, whose patterns might not be trivial even for simple configurations. Understanding how cyclic dependencies are generated is a key step to better understand our partial deployment approach. Lastly, sensitivity analyses of our approach with respects to other parameters such as the flows' characteristics represent another axis of future work.

REFERENCES

[1] "Time-Sensitive Networking (TSN) Task Group —." https://1.ieee802.org/tsn/.

[2] A. Committee *et al.*, "Aircraft Data Network Part 7, Avionics Full Duplex Switched Ethernet (AFDX) Network, ARINC Specification 664," *Annapolis, Maryland: Aeronautical Radio*, 2002.

[3] H. Kopetz, A. Ademaj, P. Grillinger, and K. Steinhammer, "The Time-Triggered Ethernet (TTE) Design," in *Object-Oriented Real-Time Distributed Computing, 2005. ISORC 2005. Eighth IEEE International Symposium on*, pp. 22–33, IEEE, 2005.

[4] "IEC/IEEE 60802 TSN Profile for Industrial Automation —." https://1.ieee802.org/tsn/iec-ieee-60802/.

[5] "IEEE Standard for Local and Metropolitan Area Networks - Timing and Synchronization for Time-Sensitive Applications in Bridged Local Area Networks," *IEEE Std 802.1AS-2011*, pp. 1–292, Mar. 2011. http://doi.org/10.1109/IEEESTD.2011.5741898.

[6] "IEEE Standard for Local and metropolitan area networks–Bridges and Bridged Networks–Amendment 29: Cyclic Queuing and Forwarding," *IEEE 802.1Qch-2017 (Amendment to IEEE Std 802.1Q-2014 as amended by IEEE Std 802.1Qca-2015, IEEE Std 802.1Qcd(TM)-2015, IEEE Std 802.1Q-2014/Cor 1-2015, IEEE Std 802.1Qbv-2015, IEEE Std 802.1Qbu-2016, IEEE Std 802.1Qbz-2016, and IEEE Std 802.1Qci-2017)*, pp. 1–30, June 2017. http://doi.org/10.1109/IEEESTD.2017.7961303.

[7] "IEEE Standard for Local and metropolitan area networks— Bridges and Bridged Networks - Amendment 24: Path Control and Reservation," *IEEE Std 802.1Qca-2015 (Amendment to IEEE Std 802.1Q-2014 as amended by IEEE Std 802.1Qcd-2015 and IEEE Std 802.1Q-2014/Cor 1-2015)*, pp. 1–120, Mar. 2016. http://doi.org/10.1109/IEEESTD.2016.7434544.

[8] *IEC 62439-3, Industrial Communication Networks - High Availability Automation Networks - Part 3: Parallel Redundancy Protocol (PRP) and High-availability Seamless Redundancy (HSR)*. 2016.

[9] *IEC 62439: High Availability Automation Networks : High Availability Automation Networks*. 2012.

[10] M. Andrews, "Instability of FIFO in the Permanent Sessions Model at Arbitrarily Small Network Loads," *ACM Trans. Algorithms*, vol. 5, pp. 33:1–33:29, July 2009. http://doi.acm.org/10.1145/1541885.1541894.

[11] A. Bouillard, M. Boyer, and E. Corronc, *Deterministic Network Calculus: From Theory to Practical Implementation*. Networks and Telecommunications, Wiley, 2018. http://doi.org/10.1002/9781119440284.

[12] J.-Y. Le Boudec and P. Thiran, *Network Calculus: A Theory of Deterministic Queuing Systems for the Internet*. Lecture Notes in Computer Science, Lect.Notes Computer. Tutorial, Berlin Heidelberg: Springer-Verlag, 2001. https://www.springer.com/us/book/9783540421849.

[13] S. Perathoner, E. Wandeler, and et al., "Influence of Different Abstractions on the Performance Analysis of Distributed Hard Real-Time Systems," *Design Automation for Embedded Systems*, 2009.

[14] J. Grieu, *Analyse et évaluation de techniques de commutation Ethernet pour l'interconnexion des systèmes avioniques*. phd, Sept. 2004. http://ethesis.inp-toulouse.fr/archive/00000084/.

[15] A. Mifdaoui, F. Frances, and C. Fraboul, "Performance analysis of a Master/Slave switched Ethernet for military embedded applications," *IEEE Transactions on Industrial Informatics*, vol. 6, no. 4, pp. 534–547, 2010.

[16] T. Ferrandiz, F. Frances, and C. Fraboul, "A Network Calculus Model for SpaceWire Networks," RTCSA, 2011.

[17] B. Jonsson, S. Perathoner, L. Thiele, and W. Yi, "Cyclic Dependencies in Modular Performance Analysis," in *Proceedings of the 8th ACM International Conference on Embedded Software*, EMSOFT '08, (New York, NY, USA), pp. 179–188, ACM, 2008. http://doi.acm.org/10.1145/1450058.1450083.

[18] H. Schiøler, J. J. Jessen, J. D. Nielsen, and K. G. Larsen, "Network Calculus for Real Time Analysis of Embedded Systems with Cyclic Task Dependencies," in *Computers and Their Applications*, pp. 326–332, 2005.

[19] D. Starobinski, M. Karpovsky, and L. A. Zakrevski, "Application of network calculus to general topologies using turn-prohibition," *IEEE/ACM Transactions on Networking*, vol. 11, pp. 411–421, June 2003. http://doi.org/10.1109/TNET.2003.813040.

[20] E. Wandeler, A. Maxiaguine, and L. Thiele, "Performance analysis of greedy shapers in real-time systems," in *Proceedings of the Design Automation Test in Europe Conference*, vol. 1, pp. 6 pp.–, Mar. 2006. http://doi.org/10.1109/DATE.2006.243801.

[21] J.-Y. Le Boudec, "A Theory of Traffic Regulators for Deterministic Networks With Application to Interleaved Regulators," *IEEE/ACM Transactions on Networking*, vol. 26, pp. 2721–2733, Dec. 2018. http://doi.org/10.1109/TNET.2018.2875191.

[22] E. Mohammadpour, E. Stai, and J. L. Boudec, "Improved Delay Bound for a Service Curve Element with Known Transmission Rate," *IEEE Networking Letters*, pp. 1–1, 2019. http://doi.org/10.1109/LNET.2019.2927143.

[23] K. Wagner, "Short evaluation of linux's token-bucket-filter (tbf) queuing discipline," *http://www.docum.org/stef.coene/qos/docs/other/tbf02_kw.ps*, 2001.

[24] H. Ayed, A. Mifdaoui, and C. Fraboul, "Hierarchical traffic shaping and frame packing to reduce bandwidth utilization in the AFDX," in *Proceedings of the 9th IEEE International Symposium on Industrial Embedded Systems (SIES 2014)*, pp. 77–86, June 2014. http://doi.org/10.1109/SIES.2014.6871190.

[25] J. Specht and S. Samii, "Urgency-based scheduler for time-sensitive switched ethernet networks," in *Real-Time Systems (ECRTS), 2016 28th Euromicro Conference on*, pp. 75–85, IEEE, 2016.

[26] R. L. Cruz, "A calculus for network delay. I. Network elements in isolation," *IEEE Transactions on Information Theory*, vol. 37, pp. 114–131, Jan. 1991. http://doi.org/10.1109/18.61109.

[27] R. L. Cruz, "A calculus for network delay. II. Network analysis," *IEEE Transactions on Information Theory*, vol. 37, pp. 132–141, Jan. 1991. http://doi.org/10.1109/18.61110.

[28] C.-S. Chang, *Performance Guarantees in Communication Networks*. Telecommunication Networks and Computer Systems, London: Springer-Verlag, 2000. https://www.springer.com/gp/book/9781852332266.

[29] E. Mohammadpour, E. Stai, M. Mohiuddin, and J. Le Boudec, "Latency and Backlog Bounds in Time-Sensitive Networking with Credit Based Shapers and Asynchronous Traffic Shaping," in *2018 30th International Teletraffic Congress (ITC 30)*, vol. 02, pp. 1–6, Sept. 2018. http://doi.org/10.1109/ITC30.2018.10053.

[30] A. Charny and J.-Y. Le Boudec, "Delay Bounds in a Network with Aggregate Scheduling," in *Quality of Future Internet Services* (J. Crowcroft, J. Roberts, and M. I. Smirnov, eds.), Lecture Notes in Computer Science, pp. 1–13, Springer Berlin Heidelberg, 2000. https://link.springer.com/chapter/10.1007/3-540-39939-9_1.

[31] A. Amari and A. Mifdaoui, "Worst-case timing analysis of ring networks with cyclic dependencies using network calculus," in *2017 IEEE 23rd International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA)*, pp. 1–10, Aug. 2017. http://doi.org/10.1109/RTCSA.2017.8046319.

[32] 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), pp. pp. 1–8, Dec. 2017. https://hal.archives-ouvertes.fr/hal-01690096.

[33] Y. Li and H. Gu, "XY-turn model for deadlock free routing in honeycomb networks-on-chip," in *2009 15th Asia-Pacific Conference on Communications*, pp. 900–903, Oct. 2009. http://doi.org/10.1109/APCC.2009.5375521.

[34] A. Baharev, H. Schichl, and A. Neumaier, "An exact method for the minimum feedback arc set problem," Fakultät für Mathematik, Universität Wien, 2015. https://www.mat.univie.ac.at/~neum/papers.html.

[35] M. R. Garey and D. S. Johnson, *Computers and Intractability; A Guide to the Theory of NP-Completeness*. New York, NY, USA: W. H. Freeman & Co., 1990.

[36] A. Tarski *et al.*, "A lattice-theoretical fixpoint theorem and its applications.," *Pacific journal of Mathematics*, vol. 5, no. 2, pp. 285–309, 1955.

[37] L. Zhao, P. Pop, Z. Zheng, and Q. Li, "Timing Analysis of AVB Traffic in TSN Networks Using Network Calculus," in *2018 IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*, pp. 25–36, Apr. 2018. http://doi.org/10.1109/RTAS.2018.00009.

[38] R. Obermaisser, *Time-Triggered Communication*. Boca Raton, FL, USA: CRC Press, Inc., 1st ed., 2011.