Admission Shaping with Network Calculus

Anne Bouillard

***Abstract*—Several techniques can be used for computing deterministic performance bounds in FIFO networks. The most popular one, as far as Network Calculus is concerned, is Total Flow Analysis (TFA). Its advantages are its algorithmic efficiency, acceptable accuracy and adapted to general topologies. However, handling cyclic dependencies is mostly solved for token-bucket arrival curves. Moreover, in many situations, flows are shaped at their admission in a network, and the network analysis does not fully take advantage of it. In this paper, we generalize the approach to piece-wise linear concave arrival curves and to shaping several flows together at their admission into the network. We show through numerical evaluation that the performance bounds are drastically improved.**

***Index Terms*—Network calculus, FIFO multiplexing, performance evaluation, shaping.**

I. Introduction

With the development of time-sensitive networks, it becomes necessary to compute reliable delay bounds in networks. To this aim, Network Calculus is a theory able to guarantee delay upper bounds. In broad strokes, two types of results are useful for computing accurate bounds: a) refined modeling of the network elements, especially for multi-class switches; b) accurate FIFO analysis, to compute delay bounds within one class of traffic. In this paper, we focus on the later type.

In Network Calculus, several methods have been studied for computing performance bounds in FIFO networks: Least Upper Delay Bounds (LUDB) [1], Total Flow Analysis (TFA) [6], [9], Linear Programming (LP) [4], [2]. While LUDB applies to very restrictive settings (token-bucket arrival curves), TFA and LP can handle more general arrival curves, link-shaping [6] and cyclic dependencies [2], [9].

TFA is the most popular method, due to its algorithmic efficiency, and can, in some cases, be quite accurate compared to the more complex LP methods. While any arrival curve can be used in feed-forward networks, using tools dedicated to the fast computation of the Network Calculus operations, like Nancy [10], analyzing networks with cyclic dependencies is usually restricted to token-buckets arrival curves. Indeed, a fixed point on the arrival curves needs to be computed, and in case of token-bucket, this boils down to computing a fixed point of the burst parameters, as FP-TFA in [7] or using LP in [2]. Recently, FP-TFA has been upgraded to very general shapes of curves in [8].

However, these works do not consider that several flows may be shaped together at the admission in the network, either due to some previous analysis or some admission control. Only considering this shaping as an arrival curve at the entrance of the network may not be enough to get accurate performances.

In this paper we generalize the LP-based fixed point approach in two directions: 1) when the arrival curve is a combination of token-buckets (that is, piece-wise linear concave); 2) when the admission into the network is regulated, hence several flows are shaped together. We show that, especially with the admission shaping, the performance bounds are drastically improved.

The paper is organized as follows: in Section II we describe our model as well as the state-to-the-art TFA method, then in Section III, we present our contribution, and in Section IV, we illustrate how our approach improves the delay guarantees through numerical evaluation.

II. Network Calculus and Total Flow Analysis

Network Calculus [5], [3] is a theory developed for computing worst-case performance upper bounds, such as end-to-end delays or backlogs. This section is devoted to a brief presentation of the model and of the TFA analysis.

A. Process and server model

A process $A : \mathbb{R}_+ \to \mathbb{R}_+$ is a non-decreasing function where $A(t)$ represents the amount of data transiting through a point of a network before time t. It is constrained by an *arrival curve* α, if for all $0 \leq s \leq t$, $A(t) - A(s) \leq \alpha(t-s)$. In other words, $\alpha(t)$ represents the maximum of data that can transit during a time interval of length t. The most classical family of functions for the arrival curves is the token-bucket: $\gamma_{b,r} : \mathbb{R}_+ \to \mathbb{R}; t \mapsto b + rt$[1]. The parameter b is called the *burst* and r the arrival rate.

A server is a relation between an arrival process A and departure process D. It is described by a *service curve* β and a *shaper* σ. On the one hand, the service curve β is a guarantee on the amount of data that can be served during time intervals: $D \geq A * \beta$, with for all $t \geq 0$, $f * g(t) = \inf_{0 \leq s \leq t} f(s) + g(t-s)$. On the other hand, the shaper σ represents the physical limitation, for example the link capacity, and bounds the maximum amount of data that can be transmitted to the next server (that is, σ is an arrival curve for D). The most classical family of functions for the service curve is the *rate-latency* curve: $\beta_{R,T} : \mathbb{R}_+ \to \mathbb{R}; t \mapsto R(t-T)_+$, and token-bucket curve $\gamma_{L,C}$ for the shaper, where in particular C represents the link capacity, and L is usually the maximum packet length.

Usually, several processes are multiplexed in a server. We assume that data is served in its arrival order (FIFO). The aggregation of processes $A_1, \ldots, A_m$ is $\sum_{i=1}^m A_i$. We will also

A. Bouillard is a researcher at the Paris Research Center of Huawei Technologies France, 18, quai du Point du Jour, 92100 Boulogne-Billancourt, France e-mail: anne.bouillard@huawei.com.

[1]This function is usually defined with $\gamma_{b,r}(0) = 0$. Here the results will not be changed by taking the simplified version, but will be easier to express.

assume *admission-shaping*: several flows, *e.g.,* $A_1, \ldots, A_m$ can be constrained by an arrival curve α at their entrance in the network. This means that $\sum_{i=1}^{m} A_i$ has arrival curve α.

We are interested in computing a) a worst-case delay bound in a server; b) an arrival curve for the departure processes from the server (of individual flows or groups of flows).

Theorem 1 ([3])**.** *Consider a FIFO server offering a service curve β and shaper σ. Assume an arrival process A to the server with arrival curve α, that A is the aggregation of two processes A_1 and A_2: $A = A_1 + A_2$, and that α_1 is a constraint for process A_1. Then,*

- *the horizontal distance between α and β is d, an upper bound of the worst-case delay of the process: $d = h(\alpha, \beta) := \sup\{u \geq 0 \mid \exists t \geq 0,\ \alpha(t) > \beta(t+u)\}$;*
- *an arrival curve for the departure process of process A_1 is $\alpha_1(\cdot + d) \wedge \sigma$, with $f(\cdot + d) : x \mapsto f(x + d)$.*

The horizontal distance, hence the delay, can be computed by the mathematical program $\max\{\mathbf{t} - \mathbf{s} \mid \alpha(\mathbf{s}) > \beta(\mathbf{t})\}$. When α and β are respectively piece-wise linear concave and convex, this is a LP, the strict inequality is not needed, and it becomes

$$\max\{\mathbf{t} - \mathbf{s} \mid \alpha(\mathbf{s}) \geq \mathbf{A} = \mathbf{D} \geq \beta(\mathbf{t})\}. \tag{1}$$

We introduce the variables $\mathbf{A}$ and $\mathbf{D}$ respectively representing the values of the processes at some times, $A(s)$ and $D(t)$, they will be used in the next LPs.

B. Network Model

- There are n FIFO servers numbered from 1 to n. Server j offers service curve β_j and is a σ_j-shaper.
- There are m flows numbered from 1 to m. Each flow i has an arrival curve α_i and follows a path (sequence of servers) $\pi_i = \langle \pi_i(1), \ldots, \pi_i(|\pi_i|) \rangle$. We denote by $\alpha_{i,j}$ the arrival curve at the output of server $j \in \pi_i$.
- There are p admission-shapers. We denote $\mathcal{P}_{adm} = \{(I_k, j_k), k \leq p\}$, where I_k the set of shaped flows and j_k the server at which the flows enter the network. The sets I_k are disjoint. The arrival curve of group k is α'_k.

The underlying graph of the network is a graph whose nodes are the servers, and the set of edges consists of the pairs of servers crossed in a row by any flow: $E = \{(\pi_i(k), \pi_i(k+1)) \mid i \leq m, k < |\pi_i|\}$. We denote by ${}^\bullet j$ (resp. $j^\bullet$) the set of predecessors (resp. successors) of j in the graph. The set of flows crossing server j is $\mathrm{Fl}(j)$ and can be decomposed into $\mathrm{Fl}(h, j)$ the set of flows following the edge (h, j) and $\mathrm{New}(j)$ the set of flows entering the network at server j.

C. Total Flow Analysis

Let us present the TFA analysis from the state of the art, that is, when there is no admission-shaping. It relies on the two results presented in Theorem 1.

For each server j, we compute

1) the arrival curve $\alpha^{(j)}$ of the arrival process of the network:
$$\alpha^{(j)} = \sum_{h \in {}^\bullet j} (\sigma_h \wedge \sum_{i \in \mathrm{Fl}(h,j)} \alpha_{i,h}) + \sum_{\mathrm{New}(j)} \alpha_i; \tag{2}$$
2) the worst-case delay bound in the server $d_j = h(\alpha^{(j)}, \beta_j)$;
3) the arrival curve for the departure process of each flow: for all $h \in {}^\bullet j$ and all $i \in \mathrm{Fl}(h, j)$, $\alpha_{i,j} = \alpha_{i,h}(\cdot + d_j)$.

Once a delay bound has been computed for all servers, an end-to-end delay bound for each flow is the sum of the delays of the servers on its path. If the network is feed-forward, these operations can be performed in the topological order of the servers. Otherwise, the functions $\alpha_{i,j}$ are computed as a fixed point. When α_i are token-buckets, $\alpha_{i,j}$ as the same arrival rate as α_i, and only the burst parameters need to be computed. It was shown in [9] that the smallest fixed point of the equation gives valid arrival curves. There can be multiple equivalent iterative ways to compute it, including using parallel programming [7]. In [2] it is proved that there is in fact a unique fixed point, that can be computed using a linear program.

III. TFA for concave arrival curves and admission-shaping

The section presents the contributions: the generalization to piecewise linear concave arrival curves and admission-shaping.

A. Piecewise linear concave arrival curves

This first generalization is rather straightforward, and it is enough to introduce additional burst parameters: if $\alpha_i = \min_{\ell \leq \ell_i} \gamma_{b_\ell, r_\ell}$, then for all $d \geq 0$, $\alpha_i(\cdot + d) = \min_{\ell \leq \ell_i} \gamma_{b_\ell + r_\ell d, r_\ell}$. So the only difference is that ℓ_i burst parameters need to be maintained instead of only one for each flow and server.

B. Admission-shaping propagation

Let us first focus on how the admission-shaping can be propagated to the subsequent servers. Consider $(I_k, j_k) \in \mathcal{P}_{adm}$, and let d_{j_k} be the delay bound of j_k. In Eq. (2), the last term, focusing on the entering flows can be changed to

$$\sum_{i \in \mathrm{New}(j_k) \setminus I_k} \alpha_i + (\alpha'_k \wedge \sum_{i \in I_k} \alpha_i). \tag{3}$$

It may also be beneficial to propagate these constraints. However, all flows do not follow the same path. Assume for example that $I_k = J_1 \cup J_2$, and that the second server of the flows in J_1 (resp. J_2) is j_1 (resp. j_2). From Theorem 1, an arrival curve for the flows in J_1 (resp. J_2) at server j_1 (resp. j_2) is then also $\alpha'_k(\cdot + d_{j_k})$. Algorithm 1 gives the flows that can be shaped together due to the propagation of constraints for each server (in short, they share a common prefix-path). They are represented by pairs: the subset of flows concerned and the server. We also construct an underlying graph. Edges $\mathcal{E}$ record how the pairs are constructed, to allow the propagation of the constraints.

Initially the pairs $\mathcal{P}_{adm}$ are given by the model. In loop (4-10), for each pair, the algorithm looks at the constraints that can be propagated to the next server: for each successor h of node j, the constraints can be propagated to the subset of flows following the edge (j, h) (lines 6-9).

Note that since the sets $(I_k)_{k \leq p}$ are disjoint, the graph $(\mathcal{P}, \mathcal{E})$ is a forest with p trees, whose roots are $\mathcal{P}_{adm}$. We

Algorithm 1: Building the admission-shaping pairs

```
begin
    P ← P_adm; E ← ∅;
    P_aux ← P;
    while P_aux ≠ ∅ do
        Let (I, j) ∈ P_aux;
        foreach h ∈ j• do
            if I ∩ Fl(j, h) ≠ ∅ then
                P ← P ∪ {(I ∩ Fl(j, h), h)};
                E ← E ∪ {(I, j), (I ∩ Fl(j, h), h)};
        P_aux ← P_aux \ {(I, j)};
    Return P
```

denote by $\mathcal{P}_k$ and $\mathcal{E}_k$ the nodes and edges in the tree with root (I_k, j_k).

C. A fixed point equation

In this paragraph, we show how to compute the burst parameters from one another. Let us first introduce some notations for the constants and variables of the linear program to come:

- the arrival curve for flow i is $\alpha_i = \min_{\ell \le \ell_i} \gamma_{b_{i,\ell}, r_{i,\ell}}$. Then for all $j \in \pi_i$, $\alpha_{i,j} = \inf_{\ell \le \ell_i} \gamma_{x_{i,j,\ell}, r_{i,\ell}}$: the variables $x_{i,j,\ell}$ represent the burst parameters of every linear piece of arrival curve of flow i after server j;
- for each admission-shaping constraint $(I_k, j_k) \in \mathcal{P}_{adm}$, the shaping constraint is $\alpha'_k = \min_{\ell \le \ell'_k} \gamma_{b'_{k,\ell}, r'_{k,\ell}}$. Then for all $(I, j) \in \mathcal{P}_k$, $\alpha'_{I,j} = \inf_{\ell \le \ell'_k} \gamma_{y_{I,j,\ell}, r'_{k,\ell}}$. The variables $y_{I,j,\ell}$ represent the burst parameter of every linear piece of the propagation of the admission-shaping after server j;
- n servers with service curve $\beta_j = \max_{q \le q_j} \beta_{R_{j,q}, T_{j,q}}$ and shaping curve $\sigma_j = \min_{q \le q'_j} \gamma_{L_{j,q}, C_{j,q}}$.

To compute the delay bounds in the network, we need to compute values for the variables of type x and y. If x and y at the input of a server are known, the two paragraphs above allow to compute the values x and y at the output of the server. This defines a function $F : \mathbb{R}_+^N \to \mathbb{R}_+^N$, where N be the total number of variables. From [3, Theorem 12.1], finding a solution boils down to finding the largest fixed point of the equation $F(z) = z$.

Table I details the computation of function F with a linear program. The variables of the LP (in bold) are the time variables $\mathbf{s}_j, \mathbf{t}_j$ and process variables $\mathbf{A}_{i,j}$ and $\mathbf{D}_j$. It follows the simple mathematical programming of Eq. (1). The main difference is that the arrival process must be decomposed into single flows (Arr.), according to their incoming edges (Shap.), or to the admission shaping pairs (A.-Sh.). Note that from one set of constraints for server j, one can compute its delay bound, as well as its related variables (burst parameters at the output of the server). Only the objective is modified. The next theorem gives a single LP for computing all delays in the network.

Maximize	$\mathbf{t}_j - \mathbf{s}_j$	(d_j)
or	$x_{i,h,\ell'} + r_{i,\ell'}(\mathbf{t}_j - \mathbf{s}_j)$	$(x_{i,j,\ell'})$
or	$y_{I,h,\ell'} + r_{k',\ell'}(\mathbf{t}_j - \mathbf{s}_j)$	$(y_{I',j,\ell'})$
such that	$0 \le \mathbf{s}_j \le \mathbf{t}_j$	(Time)
$\forall i \in \mathrm{New}(j), \ell \le \ell_i$	$\mathbf{A}_{i,j} \le b_{i,\ell} + r_{i,\ell}\mathbf{s}_j$	(Arr.)
$\forall i \in \mathrm{Fl}(h,j), \ell \le \ell_i$	$\mathbf{A}_{i,j} \le x_{i,h,\ell} + r_{i,\ell}\mathbf{s}_j$	
$\forall (I,j) \in \mathcal{P}_{adm}, \ell \le \ell'_k$	$\sum_{i \in I} \mathbf{A}_{i,j} \le b'_{k,\ell} + r_{k,\ell}\mathbf{s}_j$	(A.-Sh.)
$\forall ((I,h),(I',j)) \in \mathcal{E}_k$		
$\ell \le \ell'_k$	$\sum_{i \in I'} \mathbf{A}_{i,j} \le y_{I,h,\ell} + r_{k,\ell}\mathbf{s}_j$	
$\forall q \le q_j$	$\mathbf{D}_j \ge R_{j,q}\mathbf{t} - R_{j,q}T_{j,q}$	(Serv.)
	$\mathbf{D}_j \ge 0$	
$\forall h \in {}^\bullet j, q \le q'_h$	$\sum_{i \in \mathrm{Fl}(h,j)} \mathbf{A}_{i,j} \le L_{h,q} + C_{h,q}\mathbf{s}_j$	(Shap.)
	$\sum_i \mathbf{A}_i = \mathbf{D}_j$	(hor. d.)

TABLE I: LP for the delay bound of server j, variables $x_{i,j,\ell'}$ and $y_{I,j,\ell'}$, with $(I,j) \in \mathcal{P}_{k'}$.

Theorem 2. *F has a unique fixed point. The delay bounds for each server in the network can be computed by extracting the values of the variables $\mathbf{d}_j$ of the LP of Table II.*

Proof. Let us first have a look at the LP of Table I for computing each coordinate of F. We notice that variables x and y appear a) at most once, as upper bounds in the linear constraints b) positively in the objective. From [2, Th. 7], this means that F has a unique fixed point, and can be found as the solution of $\max\{\sum_{i=1}^N z_i \mid \forall i \le N,\ F_i(z) \le z_i\}$. This is the linear program of Table II, where the objective is replaced by the sum of all x and y variables: the first set of constraints corresponds to the computation of F_i and the second set of constraints to $F_i(z) \le z_i$. We see from the different objectives of Table I that these variables depend positively and linearly in $\mathbf{d}$. Then the objective can be replaced by optimizing the sum of the delays. □

Since the fixed point is unique, it can also be computed with iterative methods, starting from any initial value for variables of type x and y. The parallelization of iterative scheme, as described in [7], should hold similarly, starting from null values for x and y, because function F is non-decreasing.

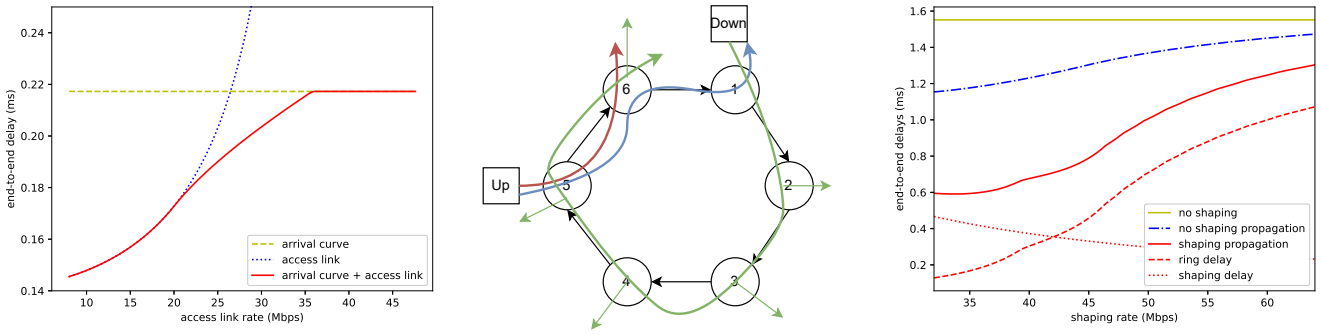
Maximize	$\sum_j \mathbf{d}_j$
such that for all server j	$0 \le \mathbf{s}_j \le \mathbf{t}_j$
	$\mathbf{d}_j = \mathbf{t}_j - \mathbf{s}_j$
$\forall i \in \mathrm{New}(j), \ell \le \ell_i$	$\mathbf{A}_{i,j} \le b_{i,\ell} + r_{i,\ell}\mathbf{s}_j$
$\forall i \in \mathrm{Fl}(h,j), \ell \le \ell_i$	$\mathbf{A}_{i,j} \le \mathbf{x}_{i,h,\ell} + r_{i,\ell}\mathbf{s}_j$
$\forall (I,j) \in \mathcal{P}_{adm}, \ell \le \ell'_k$	$\sum_{i \in I} \mathbf{A}_{i,j} \le b'_{k,\ell} + r_{k,\ell}\mathbf{s}_j$
$\forall ((I,h),(I',j)) \in \mathcal{E}_k, \ell \le \ell'_k$	$\sum_{i \in I'} \mathbf{A}_{i,j} \le \mathbf{y}_{I,h,\ell} + r_{k,\ell}\mathbf{s}_j$
$\forall q \le q_j$	$\mathbf{D}_j \ge R_{j,q}\mathbf{t} - R_{j,q}T_{j,q}$
	$\mathbf{D}_j \ge 0$
$\forall h \in {}^\bullet j, q \le q'_h$	$\sum_{i \in \mathrm{Fl}(h,j)} \mathbf{A}_{i,j} \le L_{h,q} + C_{h,q}\mathbf{s}_j$
	$\sum_i \mathbf{A}_i = \mathbf{D}_j$
$\forall i \in \mathrm{Fl}(h,j), \forall \ell \in \ell_i$	$\mathbf{x}_{i,j,\ell} \le \mathbf{x}_{i,h,\ell} + r_{i,\ell}\mathbf{d}_j$
$\forall ((I,h),(I',j)) \in \mathcal{E}_k, \forall \ell \le \ell'_k$	$\mathbf{y}_{I',j,\ell} \le \mathbf{y}_{I,h,\ell} + r'_{k,\ell}\mathbf{d}_j$

TABLE II: Linear program for computing all delays.

The number of variables grows linearly with the number of token-buckets in the description of the system, so the LP of Table II can also be solved in polynomial time.

IV. Numerical evaluation

Let us consider a unidirectional ring of n servers. The edges of the underlying graph are $\{(i, i+1), i < n\} \cup \{(n, 1)\}$. End-stations connected to that ring send packets along the whole



(a) (b) (c)

Fig. 1: Numerical experiments with $n = 20$ nodes in the ring: (a) Usecase 1: end-to-end delay bound of a flow, with $\gamma_{4\ell,4r}$, $\gamma_{\ell,C}$ and $\gamma_{4\ell,4r} \wedge \gamma_{\ell,C}$, in function of the shaping rate C. (b) Usecase 2: upstream traffic is shaped and directed from any node (here node 5) to nodes 1 and n; downstream traffic from nodes 1 and n (here node 1) to any other node. (c) Usecase 2: end-to-end delay bound comparison of an upstream flow.

ring. The service and shaping rate of the servers is 1 Gbps, and the latency of the servers is 5 μs.

In the first usecase, each end-station j sends 4 packets of length ℓ =1 kb every 1 ms along the path $\langle i, \ldots, n, 1, \ldots i-1 \rangle$, $i \leq n$. The arrival curve from each end-station is then a token-bucket with rate 4 Mbps and burst 4 kb. Assume that the links connecting the end-stations to the ring is C. Then the arrival curve to the ring can be refined to $\alpha = \gamma_{4\ell,4r} \wedge \gamma_{\ell,C}$, with r = 1 Mbps. Figure 1a shows the end-to-end delay bound of any flow if we assume that arrival curve is token-bucket ($\gamma_{4\ell,4r}$ or $\gamma_{\ell,C}$) or α. We see that considering α can capture the better of both token-buckets and also slightly improves the delay bounds.

The second use-case illustrates the power of propagating shaping constraints. Here, we rather consider the ring as an access-ring, and the upstream traffic arrives according to the scheme of Figure 1b traffic is generated at end-stations, and forwarded to the switches of the ring (we assume 8 end-stations per switch). Upstream traffic's destination is either server 1 or n. The downstream traffic is generated from nodes 1 and n to every node. Each switch in the ring has a token-bucket shaper to regulate the incoming traffic to the ring with rate C and burst ℓ =1 kb.

All flows have a token-bucket arrival curve with rate 4 Mbps and burst 4 kb. Admission-shaping for the upstream flows (2 flows) is token-bucket with rate C and burst ℓ. For the downstream ($2n$ flows from servers 1 and n), it is token-bucket with rate 160 Mbps and burst 2ℓ.

Figure 1c shows the end-to-end delay of the longest upstream flow obtained as a function of the shaping rate C (in red). The delay is the sum of the shaping delay (at the admission shaper, dotted) and the ring delay, after the shaping (dashed). Intuitively, the smaller the shaping rate, the larger the shaping delay and smaller the ring delay. We also compare the delays without admission-shaping (in blue) and when the admission-shaping constraints are not propagated (in yellow). We observe that 1) the delay bounds are greatly reduced by inserting shapers for admission in the network and are minimized for C = 33.64 Mbps; 2) propagating the shaping constraints also allows to compute more accurate performance bounds.

V. Conclusion

In this paper, we presented an extension of the TFA algorithms for the analysis of FIFO networks. It takes into account concave piece-wise linear arrival curves and shaping several flows at the admission in the network, and is valid for every topology. We demonstrate that in particular, shaping at the admission can be beneficial. The propagation of admission-shaping can also be generalized similarly to the shaping curves of the servers for flows following a common sub-path. That could be beneficial in particular for servers with a small link capacity.

References

[1] L. Bisti, L. Lenzini, E. Mingozzi, and G. Stea. Estimating the worst-case delay in FIFO tandems using network calculus. In *ValueTools '08*, pages 67:1–67:10, 2008.

[2] A. Bouillard. Trade-off between accuracy and tractability of network calculus in FIFO networks. *Perform. Evaluation*, 153:102250, 2022.

[3] A. Bouillard, M. Boyer, and E. Le Corronc. *Deterministic Network Calculus: From Theory to Practical Implementation*. ISTE, 2018.

[4] A. Bouillard and G. Stea. Exact Worst-Case Delay in FIFO-Multiplexing Feed-Forward Networks. *IEEE/ACM Trans. Netw.*, 23(5):1387–1400, 2015.

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

[6] A. Mifdaoui and T. Leydier. Beyond the Accuracy-Complexity Trade-offs of Compositional Analyses 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)*, pages 1–8, 2017.

[7] S. Plassart and J. L. Boudec. Equivalent versions of total flow analysis. *CoRR*, abs/2111.01827, 2021.

[8] S. M. Tabatabaee, M. Boyer, J.-Y. Le Boudec, and J. Migge. Efficient and accurate handling of periodic flows in time-sensitive networks. In *2023 IEEE 29th Real-Time and Embedded Technology and Applications Symposium (RTAS)*, pages 303–315, 2023.

[9] L. Thomas, J.-Y. Le Boudec, and A. Mifdaoui. On Cyclic Dependencies and Regulators in Time-Sensitive Networks. In *IEEE Real-Time Systems Symposium, RTSS*, pages 299–311, 2019.

[10] R. Zippo and G. Stea. Nancy: An efficient parallel network calculus library. *SoftwareX*, 19:101178, 2022.