A residual service curve computable in quadratic time for network calculus

Marc Boyer, Pierre Roux

A residual service curve computable in quadratic time for network calculus

Marc Boyer, Pierre Roux
ONERA / DTIS - Université de Toulouse
F-31055 Toulouse – France
Hugo Daigmorte, David Puechmaille
RealTime-at-Work
France

***Abstract*—Computing response times for resources shared by periodic workloads (tasks or data flows) can be very time consuming as it depends on the least common multiple of the periods. In a previous study, a quadratic algorithm was provided to upper bound the response time of a set a periodic tasks with fixed-priority scheduling. This paper generalises this result by giving not only a response time but a residual curve, that can be used in other contexts. It also provides a formal proof in the Coq language.**

I. Introduction

Network calculus is a theory designed to compute upper bounds on delays and memory usage in distributed real-time systems. Given such a system, the network calculus offers different ways to model it and different algorithms, producing different bounds at different computation costs.

Even if network calculus is able to analyse realistic industrial configurations in a few seconds [1], some operations have an exponential worst case complexity, related to the least common multiple (lcm) of the periods of the involved flows.

Currently, when modelling periodic or sporadic flows, one often use either an affine (*i.e.* fluid) model, with linear complexity, or a staircase model, with exponential complexity. This paper presents a quadratic solution for a very common operation, involved in the computation of a residual service for common scheduling policies.

This paper is inspired by [2], that gave a quadratic algorithm for the response time of a set of periodic real-time tasks on a CPU with fixed-priority scheduling. Since network calculus also offers methods to compute upper bounds on the response time of such systems, we had a look on the proof itself, and we found that it relies on the computation of the CPU capacity that is left to some task by the higher priority flows. This notion also exists in network calculus, where it is called "residual service" or "left-over capacity". This paper adapts the result in [2] to the network calculus framework and generalises it.

Since the proof is quite long, a formal proof, checked by the Coq proof assistant [3], is also provided.

After a presentation of a relevant subset of network calculus in Section II, and an overview of related work in Section III, the result itself is presented in Section IV, and evaluated on benchmarks in Section V.

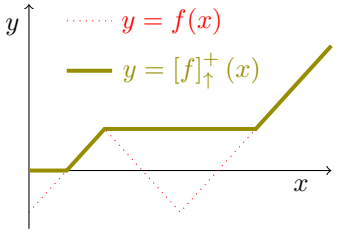


Fig. 1: Non-negative non-decreasing closure

II. Network calculus

This section provides a recall of network calculus formalism in Section II-A, with a focus on sporadic workload and rate-latency servers in Section II-B.

Notations: Let $\mathbb{R}$ denote the set of real numbers, $\mathbb{R}^+$ the subset of non-negative real numbers, $\mathbb{Z}$ the set of integers, for any $i, j \in \mathbb{Z}$, $[\![i, j]\!] = \{i, i+1, \ldots, j\}$, $\lceil\cdot\rceil : \mathbb{R} \to \mathbb{Z}$ the ceiling function ($\lceil 1.2\rceil = 2, \lceil 4\rceil = 4, \lceil -1.2\rceil = -1$). For any set X, $|X|$ denotes its cardinal. For any number $x \in \mathbb{R}$, $[x]^+ = \max(x, 0)$. For any function $f : \mathbb{R}^+ \to \mathbb{R}$, its non-decreasing non-negative closure (illustrated in Figure 1) is defined by

$$[f]^+_\uparrow(t) = \max_{0 \le s \le t} [f(s)]^+ .$$

A. Generic results

Network calculus is a theory for deriving deterministic upper bounds in networks. Network calculus mainly manipulates non decreasing functions to model flows, workload and server capacity. This section provides a short introduction. A more thorough treatment can be found in [4], [5], [6].

In network calculus, input and output flows of data are modelled by cumulative functions which represent the amount of data produced by the flow up to time t. Servers are just relations between input and output flows: a server S receives an arrival/input flow, $A(t)$, and delivers the data after some delay, as a departure/output flow, $D(t)$. We always have the relation $D \le A$, meaning that data can only go out after its arrival. However, the exact input/output data flows are in general unknown at design time, or too complex, and the calculus of these cumulative functions cannot be obtained. Nevertheless, the evolution of input/output data flows can be

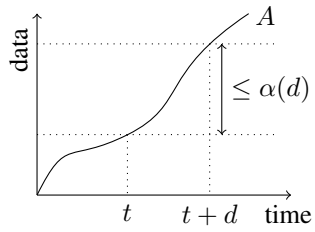


Fig. 2: Arrival curve

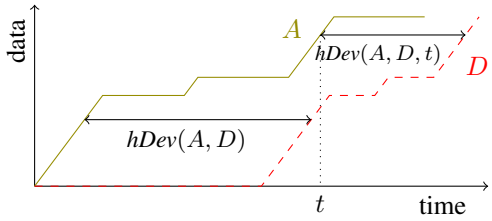


Fig. 3: Delay of the flow A

bounded considering contracts on the traffic and the services in the network. For this purpose, network calculus provides the concepts of arrival curve (illustrated in Figure 2) and service curve.

Definition 1 (Arrival curve)**.** *Let A be a flow, and α a function. Then, α is said to be an arrival curve for flow A, iff*

$$\forall (t,d) \in \mathbb{R}^+ \times \mathbb{R}^+,\ A(t+d) - A(t) \leq \alpha(d). \tag{1}$$

Definition 2 (Minimal service)**.** *A server S offers a strict minimal service curve β iff for all input/output A,D and for any backlogged period $(s,t]$ (*i.e. *such that $\forall x \in (s,t] : A(t) > D(t)$)*

$$D(t) - D(s) \geq \beta(t-s). \tag{2}$$

Let us now present the main network calculus result which allows, considering contracts, to compute bounds on delay.

Theorem 1 (Delay bound)**.** *Let S be a server transforming an arrival A into a departure D. If the order of data within the flow is preserved, the* delay *at time t is defined as $hDev(A,D,t)$, and the worst delay is $hDev(A,D)$, with*

$$hDev(A,D,t) \stackrel{def}{=} \inf\left\{d \in \mathbb{R}^+ \mid A(t) \leq D(t+d)\right\}, \tag{3}$$

$$hDev(A,D) \stackrel{def}{=} \sup_{t \in \mathbb{R}^+} hDev(A,D,t) \tag{4}$$

(see Figure 3 for an illustration).

A key point in network calculus is that arrival and service curves do not have to be tight. Mathematically they only have to be, respectively, upper and lower bounds (cf. eq. (1), eq. (2)). From a modelling point of view, they are not the exact behaviour, but only contracts. It has two complementary consequences. On one hand, if the contract is too far away from the real behaviour, the computed bounds will be large w.r.t. the real worst case. On the other hand, a complex contract can be approximated by a simpler one and all results still hold.

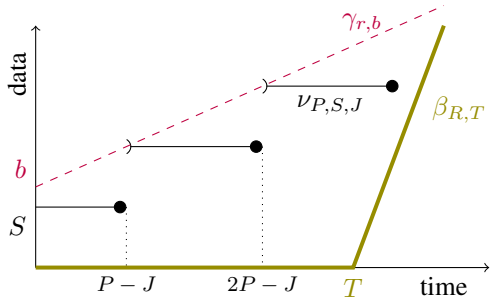


Fig. 4: Common arrival and service curves

B. Sporadic workload, rate-latency servers and NP-SP policy

This paper focuses on periodic or sporadic flows and rate-latency servers.

Given a flow sending frames of maximal size $S \in \mathbb{R}^+$ with a period or minimal inter-arrival time $P \in \mathbb{R}^+$ and a jitter $J \in \mathbb{R}^+$, it admits the arrival curve $\nu_{T,S,J} : \mathbb{R}^+ \to \mathbb{R}^+$, $t \mapsto S\left\lceil \frac{t+J}{P} \right\rceil$ but also $\gamma_{r,b} : \mathbb{R}^+ \to \mathbb{R}^+$, $t \mapsto rt + b$ ([1]), with $r = \frac{S}{P}$ and $b = r(J+P)$, as illustrated in Figure 4. Using $\nu_{P,S,J}$ is called "staircase modelling" while using $\gamma_{r,b}$ is called "fluid modelling".

Servers often offer a rate-latency service, *i.e.* a constant rate R (a data link bandwidth for example) after some latency T (some switching delay for example), modelled by a function $\beta_{R,T} : t \mapsto R\,[t-T]^+$.

When several flows share a server, its capacity β is shared between the flows, and to compute an upper bound on the delay for a flow of interest, network calculus offers to compute a residual service (aka. left-over service). The expression depends on the scheduling policy.

Theorem 2 (NP-SP residual service)**.** *Let S be a server shared by n flows. If S offers a strict minimal service curve $\beta_{R,0}$, and each flow i has arrival curve α_i and a maximal frame size S_i, then each flow j receives a residual service*

$$\beta_j = \left[\beta_{R,S_j^{\max}/R} - \sum_{k \in hp(j)} \alpha_k\right]^+_{\uparrow} \tag{5}$$

with $S_j^{\max} = \max_{k \in lp(j)} S_k$, and $hp(j)$ (resp. $lp(j)$) the set of flows with higher (resp. lower) priority than j.

The same kind of result holds, with some variations, with other type of service curves, or preemptive static priority server, FIFO or even EDF [6][2].

C. Illustrative example

Consider a bus with a bandwidth of 125kb/s, a non-preemptive static priority arbitration rule, and a latency of

[1]Readers with some background in network calculus may notice that in our definition, $\gamma_{r,b}(0) = b$ whereas the common practice is to set $\gamma_{r,b}(0) = 0$. But the results are simpler to prove with this definition, and can be easily extended to the case where the function is null at origin.

[2]Readers with a background in network calculus may have noticed only strict minimal service is presented, whereas applications of these results also involve min-plus minimal service. Since the contribution of this paper is independent of the service type, only one notion has been presented.

i	P_i	S_i	r_i	b_i
1	2.5 ms	125 b	50 kb/s	125 b
2	3.5 ms	125 b	35.72kb/s	125 b
3	3 ms	100 b	33.33 kb/s	125 b

TABLE I: Flow parameters of illustrative example

0.83ms. Three periodic flows, with period and packets sizes given in Table I are sharing this bus. Flow 3 has the lowest priority, then $hp(3) = \{1, 2\}$.

To compute the delay of this flow 3, one may choose to apply eq. (5). One may then either set $\alpha_i = \nu_{P_i,S_i,J_i}$ (staircase modelling) or $\alpha_i = \gamma_{r_i,b_i}$ (fluid modelling). In the top plot of Figure 5 are plotted the two staircase arrival curves, ν_1, ν_2 and the corresponding fluid arrival curves γ_1, γ_2. Eq. (5) involves the sum $\alpha_1+\alpha_2$, being either $\nu_1+\nu_2$ or $\gamma_1+\gamma_2$ (both are in the second plot of Figure 5), and the two residual services $\beta_3^{stc} = [\beta - \nu_1 - \nu_2]^+_\uparrow$, $\beta_3^{fluid} = [\beta - \gamma_1 - \gamma_2]^+_\uparrow$ are also plotted (given in the third and fourth plots of Figure 5). The latency is then bounded either by $hDev(\alpha_3, \beta_3^{stc}) = h_1$ or $hDev(\alpha_3, \beta_3^{fluid}) = h_1 + h_2 + h_3$.

As expected, the staircase modelling, that captures in a more accurate way the behaviour of the flows, gives a smaller bound than the fluid modelling.

D. Problem statement

Whereas the staircase modelling computes better bounds, it has several drawbacks.

One problem is the cost of the addition: whereas the addition with a fluid model is easy to compute (there is a closed-form formula, $\sum_{i\in I} \gamma_{r_i,b_i} = \gamma_{\sum_i r_i, \sum_i b_i}$, whose cost is linear w.r.t. $|I|$), on the contrary, the addition with a staircase model is hard: there exists no closed-form formula, only algorithms [7], and the computation requires to unroll the function up to the least common multiple of the periods[3], leading to exponential complexity.

Another problem is the absence of a closed-form formula. Closed formulae, and especially those involving linear terms, allow to perform explicit and efficient optimisations.

A last problem is related to the implementation: not all tools are able to handle staircase functions, and several only consider linear arrival curves, as presented in next section.

The contribution of this paper is to give a rate-latency residual service that lies between the staircase and the fluid residual service curves, denoted $\beta_{R',C/R'}$ in Figure 5.

III. Related work

A. Implementation of algebraic operators for network calculus

Practical application of network calculus requires an implementation of algebraic operations on functions.

For years, work has concentrated exclusively on linear functions, using closed-form formulae [5], and some tools were even only using affine arrival curves and rate-latency service curves [8].

The subclass of concave or convex piecewise linear functions has also received some attention [9], [10] and is the class currently used in the DISCO tool [11], [12].

A big step was the development of the (min,plus) library for the RTC toolbox [13], representing piecewise linear functions (called VCCs) as a collection of segments [14, Sec. 7].

A major breakthrough has been achieved with the definition of the class of ultimately pseudo periodic functions, generalising VCCs, and the development of the algorithms allowing effective computation [7].

The problem of computation time has not yet received a lot of attention in academia.

In [15], the idea is to maintain a staircase arrival curve per flow, but to approximate it by a concave piecewise linear function of two segments before doing the sum, to keep linear complexity.

The notion of a "container" is developed in [16], with $O(n \log n)$ complexity on operations.

Another line of work is based on the fact that the computation of the bounds (the horizontal deviation, *hDev*) is based only on the prefix of the involved functions, and that one can maintain only a prefix and approximate the remainder of the function by an affine segment [17], [18], [19].

Lastly, another way to reduce the computation time (and to get larger upper bounds) is to replace some periods T_i by a smaller value T'_i but such that the lcm of the T'_i is smaller than the lcm of the T_i [20], [21].

B. Coq for real-time systems

Coq is a proof assistant [3], *i.e.*, a tool offering a language to state theorems and describe their proofs as well as a software[4] verifying the proofs. It can also be used to develop software whose execution is proved to be conform to their (formal) specification such as the CompCert C compiler [22] or the CertiKOS operating system [23]. When used as a proof checker, Coq will complain when attempting to use a lemma without providing a proof for one of its hypotheses or if the proved hypotheses do not match the expected ones.

Proving that a systems guarantees some real-time property is often a complex task, requiring long and complex proofs. One way to build correct analyses is to use a proof assistant, like Coq [24] or Isabelle/HOL.

IV. Contribution

This section details the main contribution of the paper: given a rate-latency curve $\beta_{R,T}$ and a set of staircase functions ν_{T_i,C_i,J_i}, there exists a rate-latency function $\beta_{R',C/R'}$ which is a lower bound, as shown in eq. (6), that can be used to compute residual service.

Theorem 3 (Quadratic rate-latency bound)**.** *Let $R, T, C_1, \ldots, C_n, T_1, \ldots, T_n$ (resp. $J_1, \ldots, J_n$) be a set*

[3]In practice, periods are often integers or rational numbers that can be mapped to integers once a common denominator is found.

[4]Think of it as a compiler (in practice it is indeed a compiler for a very strongly typed language).

of positive real (resp. non negative real) values such that $\sum_{i=1}^{n} \frac{C_i}{T_i} < R$. *Then*

$$\left[\beta_{R,T} - \sum_{i=1}^{n} \nu_{T_i,C_i,J_i}\right]^{+}_{\uparrow} \geq \beta_{R',C/R'} \tag{6}$$

with

$$R' = R - \sum_{i=1}^{n} \frac{C_i}{T_i}, \qquad C = RT + W - \frac{\max\{L,Q\}}{R},$$

$$W = \sum_{i=1}^{n} \left(T_i + J_i - \frac{C_i}{R}\right) \frac{C_i}{T_i},$$

$$L = \min_{k \in [\![1,n]\!]} C_k \left(\sum_{i=1}^{n} \frac{C_i}{T_i} - \max_{i \in [\![1,n]\!]} \frac{C_i}{T_i}\right),$$

$$Q = \sum_{i=1}^{n} \sum_{j=1}^{i-1} \min\{T_i, T_j\} \frac{C_i C_j}{T_i T_j}.$$

The expression of the function $\beta_{R',C/R'}$ involves only simple sums (sub-terms R', W and L) and one double sum (sub-term Q) leading to quadratic complexity $O(n^2)$. To obtain a linear complexity, one may omit the term Q, leading to a smaller curve (*i.e.* a worst service) but in a shorter time.

Two proofs are given. In appendix A is given a "pen and paper" proof. This proof being non trivial, we chose to get a high level of confidence in its soundness. We formalized and verified it with Coq. A feedback of this use is given at the end of the current section and an overview of the Coq proof is given in appendix B.

The next theorem states that the previous result is an enhancement w.r.t. a fluid modelling.

Theorem 4. *Let* $R, T, C_1, \ldots, C_n, T_1, \ldots, T_n, J_1, \ldots, J_n, R', C$ *be as in Theorem 3. Then*

$$\beta_{R',C/R'} > \left[\beta_{R,T} - \sum_{i=1}^{n} \gamma_{r_i,b_i}\right]^{+}_{\uparrow} \tag{7}$$

with $r_i = \frac{C_i}{T_i}$, $b_i = r_i(J_i + T_i)$.

Proof. The first step consists in an expression of linear residual service. First, $\sum_{i=1}^{n} \gamma_{r_i,b_i} = \gamma_{\sum_{i=1}^{n} r_i, \sum_{i=1}^{n} b_i}$, then for any $t \in \mathbb{R}^+$:

$$\left[\beta_{R,T}(t) - \sum_{i=1}^{n} \gamma_{r_i,b_i}(t)\right]^{+} \tag{8}$$

$$= \left[[R(t-T)]^{+} - \left(\sum_{i=1}^{n} r_i\right) t - \sum_{i=1}^{n} b_i\right]^{+} \tag{9}$$

$$= \left[\left(R - \sum_{i=1}^{n} r_i\right) t - \left(RT + \sum_{i=1}^{n} b_i\right)\right]^{+} \tag{10}$$

$$= \beta_{R',C'/R'} \tag{11}$$

with $C' = RT + (\sum_{i=1}^{n} b_i) = RT + \sum_{i=1}^{n} (J_i + T_i)\frac{C_i}{T_i}$. Just looking at this expression, $C' > C$ so $\beta_{R',C/R'} > \beta_{R',C'/R'}$. □

Figure 5 illustrates the differences between the functions and highlights the influence of the non-decreasing closure. As

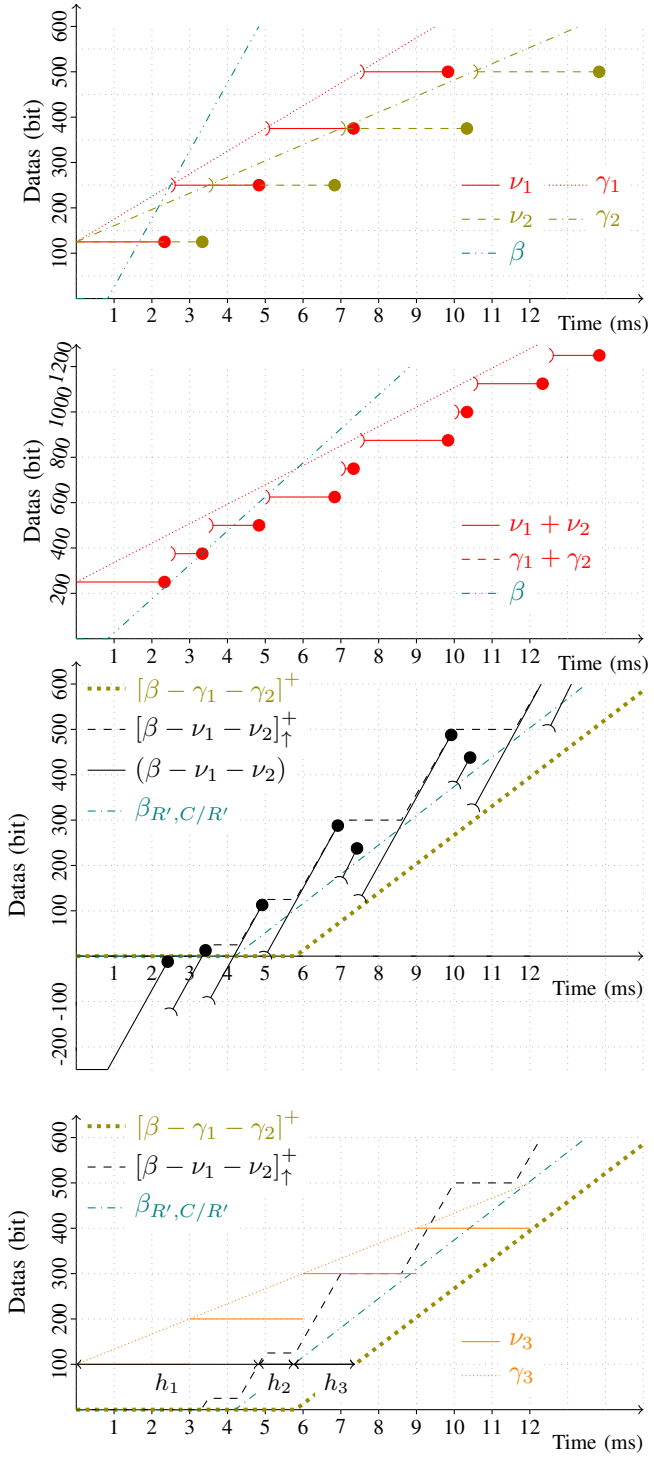


Fig. 5: Illustration of different curves, with $R = 125kb/s$, $T = .83ms$ and $\forall i \in \{1,2,3\}, \nu_i = \nu_{P_i,S_i,0}$, $\gamma_i = \gamma_{r_i,b_i}$, for the values of P_i, S_i, r_i and b_i given in Table I.

expected, since fluid modelling gives a larger arrival curve than staircase modelling ($\gamma_i \geq \nu_i$), then the fluid residual curve is less than or equal to the staircase one: $\beta - \sum_i \gamma_i \leq \beta - \sum_i \nu_i$. As stated by the Theorem, $\beta_{R',C/R'} \leq [\beta - \sum_i \gamma_i]^+_\uparrow$ but $\beta_{R',C/R'}$ is not smaller than $\beta - \sum_i \gamma_i$.

Comparison with [2]: This result is of course closely related to the one in [2], and once the equation is given, the amount of generalisation can be detailed. Using network calculus, the response time of a task of execution time C_0 and period T_0 on a CPU with speed one ($R = 1$) and no latency ($T = 0$) can be bounded by $\mathit{hDev}(\gamma_{C_0/T_0,C_0}, \beta_{R',C/R'}) = C/R' + C_0/R' = \frac{C_0+W-\max\{L,Q\}}{R'}$ whereas the expression in [2, Thm. 1] is $\frac{C_0+W-Q}{R'}$. The contribution of this paper is then: the modelling of the speed R and the latency T of a server, the introduction of the linear term L and the extraction of the residual curve, that can be used in more contexts that the fixed priority scheduling.

Feedback on the use of Coq: The use of Coq gave us the opportunity to fix a few small mistakes in a preliminary version of the proof of Theorem 3 and one of its hypotheses.

One of the last steps of the proof consists is showing that a value s is non-negative (step 11 in Appendix A). It was claimed as an evidence, even with negative values J_i of the jitters. While trying to encode this "evidence" is Coq, we had to assume that the J_i values must be non-negative, and the hypotheses have been updated. We do not know currently whether the property holds with negative J_i values.

One step of the proof (an index permutation, step 9.c in Appendix A) was using a wrong argument, doing a confusion between values and indexes. The proof has been corrected.

Regarding the cost of the development, it can be considered reasonable as only 1400 lines of Coq code were needed[5], requiring about two man×weeks of development[6], (including above mentioned proof fixes). This was made possible thanks to the availability of a formalization of the real numbers in Coq's standard library as well as the nice Mathematical Components library [25] and particularly its big operators [26] to conveniently manipulate the Σ notation for sums.

V. Evaluation

This section evaluates the quality of the approximation provided in this paper, in terms of accuracy of the result and computational cost.

To do so, we test the expression on a large set of configurations. Each configuration represents a non-static priority server, with a constant rate of 1Mb/s, no latency, and a set of randomly generated sporadic flows. Let c_i be a configuration, each flow $f_{i,j}$ has priority j, a fixed packet size $C_{i,j}$ chosen uniformly between 8 and 16 bytes, a period $T_{i,j}$ also randomly chosen in a subset of values, and a jitter $J_{i,j}$ also randomly chosen. New flows are added up to reaching a global load of 90%, and n_i denotes the number of flows.

One hundred configurations are generated picking periods values from S1 of Table II and with no jitter, another hundred using set S2 and also with no jitter, and another hundred using set S3 of the same table and also no jitter. Three others sets are generated in a similar way, but with a jitter uniformly distributed between 0 and the flow period (excluded).

TABLE II: Periods of flows (in ms)

Set name	S1	S2	S3
Period values	2,5,10,20,25,40,50	2,3,4,5,6,7,8,9,10	2,3,5,7,11,13
lcm	200	2520	30030

For each configuration c_i, let $f_{i,1}, \ldots, f_{i,n_i}$ be the set of flows. For each flow $f_{i,j}$, four bounds on the delay bound are computed using different methods. The two first have been used in the illustrative example in Section II-C.

1) $d^{\mathit{fluid}}_{i,j} = \mathit{hDev}(\alpha_i, \beta^{\mathit{fluid}}_{i,j})$, where $\beta^{\mathit{fluid}}_{i,j}$ is computed using eq. (5) with $\alpha_k = \gamma_{C_k/T_k,C_k(1+J_k/T_k)}$. It is called the *fluid* modelling.
2) $d^{\mathit{stc}}_{i,j} = \mathit{hDev}(\alpha_i, \beta^{\mathit{stc}}_{i,j})$, where $\beta^{\mathit{stc}}_{i,j}$ is computed using eq. (5) with $\alpha_k = \nu_{T_k,C_k,J_k}$. It is called the *staircase* modelling.
3) $d^{\mathit{lin}}_{i,j} = \mathit{hDev}(\alpha_i, \beta^{\mathit{lin}}_{i,j})$ where $\beta^{\mathit{lin}}_{i,j}$ is computed using Theorem 3 but only with the linear term L (*i.e.* setting $Q = 0$). It is called the *linear* modelling.
4) $d^{\mathit{quad}}_{i,j} = \mathit{hDev}(\alpha_i, \beta^{\mathit{quad}}_{i,j})$ where $\beta^{\mathit{quad}}_{i,j}$ is computed using Theorem 3 but only with the quadratic term Q (*i.e.* setting $L = 0$). It is called the *quadratic* modelling.

Experiments have run on a laptop with 4GB of memory and a 2.7GHz Intel Core i5.

Figure 6a plots, for a given configuration c^k with periods chosen in S1 (harmonic periods) and no jitter, the bounds $d^{\mathit{fluid}}_{k,j}$, $d^{\mathit{stc}}_{k,j}$, $d^{\mathit{quad}}_{k,j}$ $d^{\mathit{lin}}_{k,j}$ computed by the four methods for each flow. Since flows are sorted by priority, the plots are non decreasing. As expected, the fluid modelling gives the larger, *i.e.* worse, bounds, whereas the linear approximation is smaller, the quadratic approximation even smaller, and staircase modelling leads to the smallest bounds. Only one configuration is plotted, but they all have the same shape.

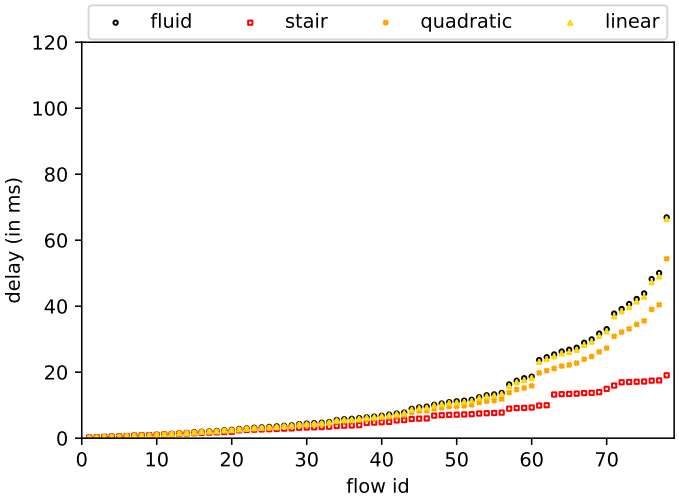
Now considering the fluid modelling as the reference value, Figure 6c plots, for each configuration with harmonic periods, the sum of all bounds computed by a method divided by the sum of all bounds computed by fluid modelling: $\frac{\sum_{j=0}^{n_i} d^X_{i,j}}{\sum_{j=0}^{n_i} d^{\mathit{fluid}}_{i,j}}$ with $X \in \{\mathit{fluid}, \mathit{stc}, \mathit{lin}, \mathit{quad}\}$. Figure 6e plots the computation time required to analyse each configuration, depending on the modelling, with a log-scale on time axis.

In the same figure group are also plotted the same graphs but considering the jitter of each flow picked up between 0 and the flow period. As expected, the jitter increases the delay of the affine models, but has no influence on the staircase one (cf. Figure 6b). Then, the gain obtained by the staircase model w.r.t. the fluid model increases, whereas the gain of the quadratic model is less (12% instead of 16%).

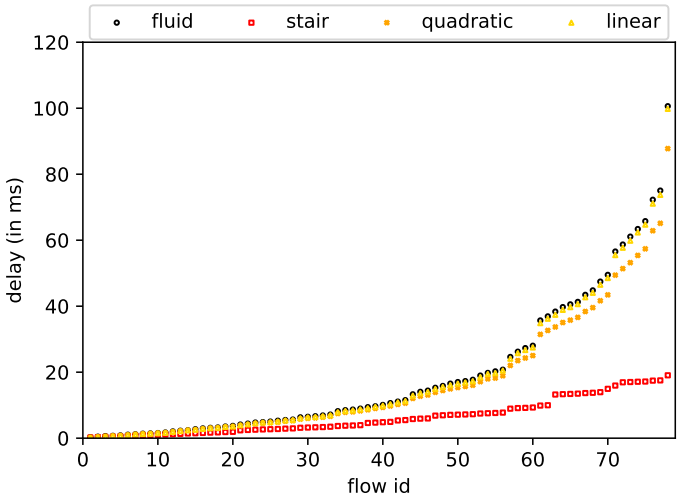
The Table III summarises, for each set of hundred configurations, the mean bound on delays for all flows, and the mean computing time for a single configuration. For the staircase modelling is added this computation time divided by the lcm of the periods, showing that this computation time is almost linear w.r.t. this lcm.

[5]214 lines for statements, 989 lines for proofs and 49 lines of comment (the remaining being blank lines).

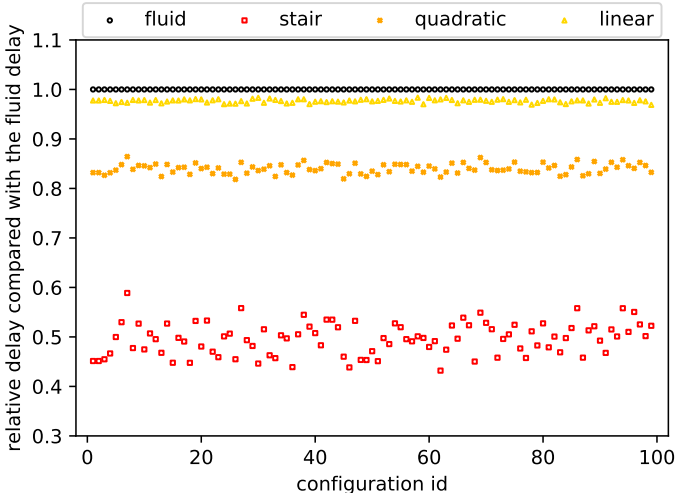
[6]For a developper with a few years of experience with the tool.



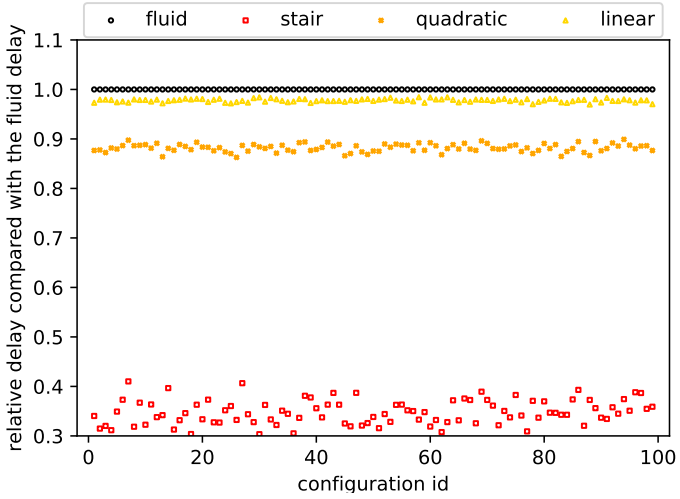
(a) Per flow delay bound, for one configuration, null jitter.



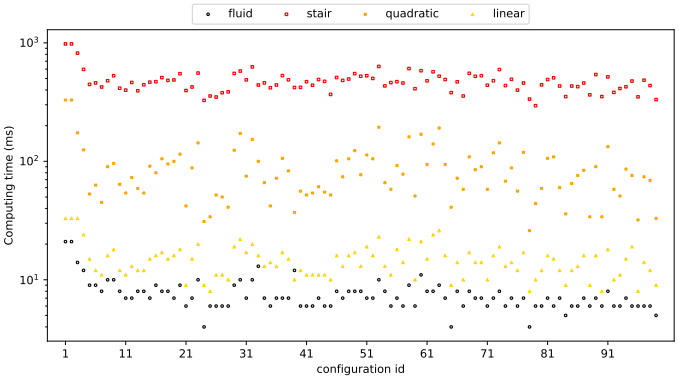
(b) Per flow delay bound, for one configuration, random jitter.



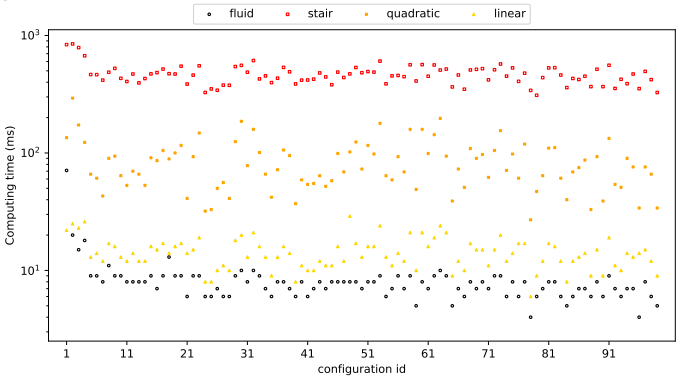
(c) Per configuration mean delay w.r.t. to fluid modelling, null jitter.



(d) Per configuration mean delay w.r.t. to fluid modelling, random jitter.



(e) Per configuration computing time, null jitter.



(f) Per configuration computing time, random jitter .

Fig. 6: Plots related to configurations with periods in S1 set, null w.r.t. random jitters

The same kind of information is plotted in the group of Figures 6 when the periods are taken from the set S3. The relations between the methods in terms of accuracy of results are in the same order of magnitude (from 16% to 22% without jitter, from 11% to 17% with jitter), but the computation time of the staircase methods is three orders of magnitude larger (50s vs. 21ms).

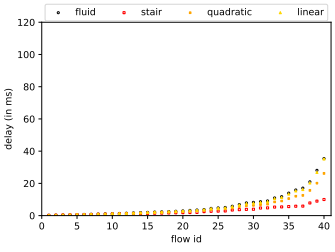
The results for the set S2 are not plotted but are summarised in Table III.

VI. Conclusion

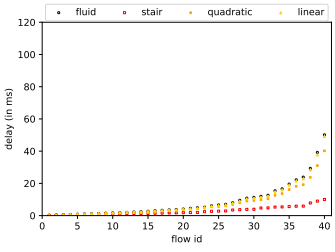
In network calculus, the computation of residual services with staircase arrival curves has exponential complexity,

TABLE III: Mean computed bounds and computing time

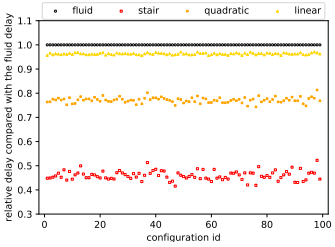
Configuration Periods	Configuration Jitters	Method Fluid	Linear	Quadratic	Staircase
Mean computed bounds, per flow, in ms, and gain w.r.t. fluid modelling					
S1	Null	12.3	12.0 (-2%)	10.3 (-16%)	6.1 (-50%)
S1	Rand.	17.6	17.2 (-2%)	15.5 (-11%)	6.1 (-65%)
S2	Null	7.7	7.4 (-3%)	5.7 (-25%)	3.4 (-56%)
S2	Rand.	10.6	10.2 (-3%)	8.6 (-19%)	3.4 (-68%)
S3	Null	7.2	6.9 (-3%)	5.6 (-22%)	3.3 (-54%)
S3	Rand.	9.9	9.5 (-3%)	8.2 (-17%)	3.3 (-66%)
Mean computing time, per configuration, in ms (and ratio w.r.t. lcm for staircase)					
S1	Null	9	15	96	567 (2.8)
S1	Rand.	10	18	101	597 (3.0)
S2	Null	6	7	26	5239 (2.1)
S2	Rand.	6	7	24	4935 (2.0)
S3	Null	6	6	21	51657 (1.7)
S3	Rand.	6	6	21	50226 (1.7)



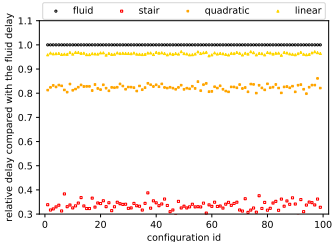
(a) Per flow delay bound, for one configuration, null jitter.



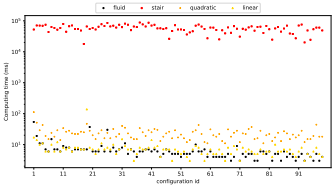
(b) Per flow delay bound, for one configuration, random jitter.



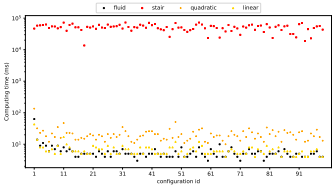
(c) Per configuration mean delay w.r.t. to fluid modelling, null jitter.



(d) Per configuration mean delay w.r.t. to fluid modelling, random jitter.



(e) Per configuration computing time, null jitter.



(f) Per configuration computing time, random jitter .

Fig. 7: Plots related to configurations with periods in S3 set, null w.r.t. random jitters

whereas fluid arrival curves offer a linear complexity but give larger, *i.e.* worse, upper bounds.

This paper generalises a result from [2], and develops a residual service curve with either linear or quadratic computational complexity. The correctness of the result is enforced by providing a formal Coq proof. The different approaches are evaluated on 600 systems with sporadic workload and non-preemptive static priority scheduling.

Whereas the staircase model computes bounds that are half of those of the fluid model[7], at the expense of a computation time from 10^2 to 10^4 times larger, the quadratic approach already enhances the results by about 20% while being only 10 times slower. The linear model offers a limited enhancement (2%-5%). Having accurate results in short computation times helps real-time system designers when exploring several configurations.

Moreover, the analytic formula of the residual service curves opens some opportunities. First, having a residual curve allows to use the Pay Burst Only Once principle, to compute an end-to-end network delay smaller than the sum of per switch delays. Second, a closed form formula gives opportunities for optimisation. Third, getting rid of least common multiple allows the use of directed rounding floating-point arithmetic that could lower the computation cost by one or two additional orders of magnitude.

REFERENCES

[1] M. Boyer, N. Navet, and M. Fumey, "Experimental assessment of timing verification techniques for AFDX," in *Proc. of the 6th Int. Congress on Embedded Real Time Software and Systems*, Toulouse, France, 2012.

[2] E. Bini, A. Parri, and G. Dossena, "A quadratic-time response time upper bound with a tightness property," in *Proc. of the 36th IEEE Real-Time Systems Symposium (RTSS 2015)*, San Antonio, USA, December 2015.

[3] *The Coq proof assistant reference manual*, The Coq development team, 2019, version 8.11. [Online]. Available: https://coq.inria.fr

[4] C.-S. Chang, *Performance Guarantees in communication networks*, ser. Telecommunication Networks and Computer Systems. Springer, 2000.

[5] J.-Y. Le Boudec and P. Thiran, *Network Calculus*, ser. LNCS. Springer Verlag, 2001, vol. 2050, http://lrcwww.epfl.ch/PS_files/NetCal.htm.

[6] A. Bouillard, M. Boyer, and E. Le Corronc, *Deterministic Network Calculus – From theory to practical implementation*. Wiley, 2018.

[7] A. Bouillard and E. Thierry, "An algorithmic toolbox for network calculus," *Discrete Event Dynamic Systems*, vol. 18, no. 1, pp. 3–49, october 2008.

[8] L. Bisti, L. Lenzini, E. Mingozzi, and G. Stea, "DEBORAH: a tool for worst-case analysis of FIFO tandems," in *Proc. of the 4th Int. Symp. On Leveraging Applications of Formal Methods, Verification and Validation (ISoLA 2010)*, ser. LNCS. Springer, 2010.

[9] H. Sariowan, R. L. Cruz, and G. C. Polyzos, "SCED: A generalized scheduling policy for guaranteeing quality-of-service," *IEEE/ACM transactions on networking*, vol. 7, no. 5, pp. 669–684, October 1999.

[10] M. Boyer and C. Fraboul, "Tightening end to end delay upper bound for AFDX network with rate latency FCFS servers using network calculus," in *Proc. of the 7th IEEE Int. Workshop on Factory Communication Systems Communication in Automation (WFCS 2008)*. IEEE industrial Electrony Society, May 21-23 2008, pp. 11–20.

[11] J. B. Schmitt and F. A. Zdarsky, "The DISCO network calculator - a toolbox for worst case analysis," in *Proc. of the First International Conference on Performance Evaluation Methodologies and Tools (VALUETOOLS'06), Pisa, Italy*. ACM, Nov. 2006.

[12] S. Bondorf and J. B. Schmitt, "The DiscoDNC v2 – a comprehensive tool for deterministic network calculus," in *Proc. of the 8th Int. Conf. on Performance Evaluation Methodologies and Tools (VALUETOOLS 2014)*, Dec. 2014.

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

[14] E. Wandeler, "Modular performance analysis and interface-based design for embedded real-time systems," Ph.D. dissertation, ETH Zurich, September 2006.

[15] M. Boyer, J. Migge, and N. Navet, "An efficient and simple class of functions to model arrival curve of packetised flows," in *Proc. of the 1st Int. Workshop on Worst-Case Traversal Time (WCTT'2011)*. ACM, 2011, pp. 43–50.

[7]It must be mentioned that a previous study on a realistic avionic configuration, based on Ethernet and 2 priority levels, has shown a gain related to staircase of only 6% [27] and another on a more loaded configuration gave a gain of 18% [1].

[16] E. Le Corronc, B. Cottenceau, and L. Hardouin, "Container of (min,+)-linear systems," *Journal of Discrete Event Dynamic Systems*, vol. 14, no. 1, pp. 15–52, March 2014.

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

[18] K. Lampka, S. Bondorf, and J. Schmitt, "Achieving efficiency without sacrificing model accuracy: Network calculus on compact domains," in *Proc. of the 24th IEEE Int. Symp. on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS 2016)*, 2016.

[19] U. Suppiger, S. Perathoner, K. Lampka, and L. Thiele, "A simple approximation method for reducing the complexity of modular performance analysis," Computer Engineering and Networks Laboratory – Swiss Federal Institute of Technology (ETH), TIK-Report 329, August 2010.

[20] N. Navet, T. L. Mai, and J. Migge, "Using machine learning to speed up the design space exploration of Ethernet TSN networks," http://hdl.handle.net/10993/38604, University of Luxembourg, Tech. Rep. 10993/38604, January 2019.

[21] N. Navet, J. Migge, J. Villanueva, and M. Boyer, "Pre-shaping bursty transmissions under IEEE802.1Q as a simple and efficient QoS mechanism," *SAE Int. Journal of Passenger CarsElectronic and Electrical System*, vol. 11, no. 3, 2018.

[22] X. Leroy, "Formal verification of a realistic compiler," *Commun. ACM*, vol. 52, no. 7, pp. 107–115, 2009. [Online]. Available: https://doi.org/10.1145/1538788.1538814

[23] R. Gu, Z. Shao, H. Chen, X. N. Wu, J. Kim, V. Sjöberg, and D. Costanzo, "Certikos: An extensible architecture for building certified concurrent OS kernels," in *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*. Savannah, GA: USENIX Association, Nov. 2016, pp. 653–669. [Online]. Available: https://www.usenix.org/conference/osdi16/technical-sessions/presentation/gu

[24] F. Cerqueira, F. Stutz, and B. B. Brandenburg, "PROSA: A case for readable mechanized schedulability analysis," in *Proc. of the 28th Euromicro Conference on Real-Time Systems (ECRTS 2016)*, Toulouse, France, July 5-8 2016, pp. 273–284. [Online]. Available: http://dx.doi.org/10.1109/ECRTS.2016.28

[25] G. Gonthier, A. Mahboubi, and E. Tassi, "A Small Scale Reflection Extension for the Coq system," INRIA, Research Report RR-6455, 2008. [Online]. Available: http://hal.inria.fr/inria-00258384

[26] Y. Bertot, G. Gonthier, S. Ould Biha, and I. Pasca, "Canonical big operators," in *Theorem Proving in Higher Order Logics*, O. A. Mohamed, C. Muñoz, and S. Tahar, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 86–101.

[27] M. Boyer, J. Migge, and M. Fumey, "PEGASE, a robust and efficient tool for worst case network traversal time," in *Proc. of the SAE 2011 AeroTech Congress & Exhibition*, Toulouse, France, 2011.

[28] L. Lamport, "How to write a 21st century proof," *Journal of Fixed Point Theory and Applications*, vol. 11, no. 1, pp. 43–63, Mar 2012.

APPENDIX A
PROOF OF THEOREM 3

Regarding only correctness issues, we may have omitted this section since the Coq proof already provides a formal correctness insurance. Nevertheless, this section can be considered as the documentation of the Coq proof. But the main justification of this section relies in the opportunity to adapt or generalise the results. The same way as we have converted and extended the result on response time presented in [2] by a study of its proof, we provide a human-oriented proof, as a complement of the formal Coq proof.

The proof presentation is inspired by [28]. Each sub-step of the proof will start with some ordering value, followed by the statement of the sub-step, using bold font. Thereafter will come the proof of the sub-step itself.

For the proof, let $V \stackrel{def}{=} \sum_{i=1}^{n} \nu_{T_i,C_i,J_i}$ *i.e.* $V(t) = \sum_{i=1}^{n} C_i \left\lceil \frac{t+J_i}{T_i} \right\rceil$, and $\rho = \sum_{i=1}^{n} \frac{C_i}{T_i}$ the long term rate of V, and recall that $\rho < R$.

1) **Definitions of s^M and first properties**: For any $M \in \mathbb{R}$, let

$$s^M \stackrel{def}{=} \min \{t \in \mathbb{R} \mid V(t) + M = R(t-T)\}. \quad (12)$$

This s^M is the minimal solution to $V(t)+M = R(t-T)$. The first step consists in showing that s^M exists (there are solutions, and there exists a minimal one), and the second on their relative positions (cf. Figure 8).

a) **The minimum exists:** By definition of the ceiling function, $x \leq \lceil x \rceil < x+1$. Then, for any i, $t\frac{C_i}{T_i} + J_i\frac{C_i}{T_i} \leq C_i \left\lceil \frac{t+J_i}{T_i} \right\rceil < t\frac{C_i}{T_i} + J_i\frac{C_i}{T_i} + C_i$. Making the sum for all $i \in [1,n]$ leads to

$$\forall t \in \mathbb{R} : \rho t + b \leq V(t) < \rho t + b', \quad (13)$$

with $b = \sum_{i=1}^{n} J_i\frac{C_i}{T_i}$, $b' = b + \sum_{i=1}^{n} C_i$.
These are affine functions, and since $R > \rho$, there exists $x < x'$ such that $\rho x + b = R(x-T) - M$ and $\rho x' + b' = R(x'-T) - M$ (cf. Figure 8). Set $y = \rho x + b$, $y' = \rho x' + b'$.
From eq. 13, for any $t \in [x, x'] : y \leq V(t) \leq y'$. Set $Y = \{V(t) \mid t \in [x, x']\}$ the set of values of V on $[x, x']$. This set is non-empty and finite. If $(v_i)_{i\in\mathbb{N}}$ is the ordered set of values of the function V, there exists $k \leq k'$ such that $Y = \{v_k, v_{k+1}, \ldots, v_{k'}\}$ ($Y = \{v_3, v_4\}$ on the example in Figure 8), and to each one corresponds one s_k^M such that $v_k = R(s_k^M - T) - M$. Then the set $\{t \in \mathbb{R} \mid V(t) + M = R(t-T)\} = \{s_k^M, \ldots, s_{k'}^M\}$ is non empty and finite, and its minimum, s^M exists.

b) **Before s^M, $R(t-T) - M$ is below $V(t)$ *i.e.***

$$\forall t < s^M : R(t-T) - M < V(t): \quad (14)$$

By contradiction, assume there exists $t < s^M$ such that $R(t-T) - M \geq V(t)$. The case $R(t-T) - M = V(t)$ leads to $t \geq s^M$ by definition of s^M. In case of $R(t-T)-M > V(t)$, then $R(t-T)-M > \rho t$, so $t > x$.

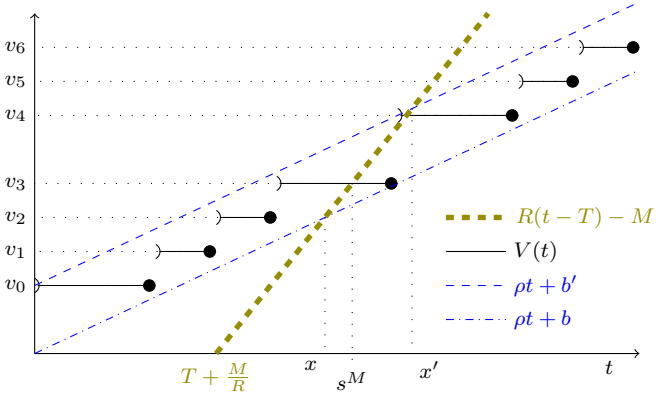


Fig. 8: Illustration of s^M definition, with $n = 2$, $C_1 = C_2 = \frac{1}{2}$, $T_1 = 2$, $T_2 = 3$, $J_1 = J_2 = 0$

c) **A lower bound on s^M:** In step 1a, x has been defined such that $s^M \in [x, x']$, with $\rho x + b = R(x - T) - M$, then

$$s^M \geq x = \frac{M + RT + \sum_{i=1}^{n} J_i \frac{C_i}{T_i}}{R'}. \tag{15}$$

This relation will be used in one of the last step of the proof.

2) **Definitions of q_i^M, r_i^M and first properties:** Let introduce for any $i \in [\![1, n]\!]$,

$$q_i^M \stackrel{def}{=} \left\lceil \frac{s^M + J_i}{T_i} \right\rceil \quad r_i^M \stackrel{def}{=} T_i q_i^M - (s^M + J_i) \tag{16}$$

keep in mind that $q_i^M = \frac{s^M + J_i + r_i^M}{T_i}$ and that $T_i > r_i^M \geq 0$ (from $\frac{x}{T} \leq \left\lceil \frac{x}{T} \right\rceil < \frac{x}{T} + 1$ comes $0 \leq T \left\lceil \frac{x}{T} \right\rceil - x < T$, and setting $x = s^M + J_i$).

3) **Expression of s^M in terms of r_i^M:** Since s^M is a minimum, it satisfies $R(s^M - T) = V(s^M) + M$ *i.e.*

$$R(s^M - T) = \sum_{i=1}^{n} C_i \left\lceil \frac{s^M + J_i}{T_i} \right\rceil + M = \sum_{i=1}^{n} C_i q_i^M + M \tag{17}$$

$$= \sum_{i=1}^{n} \frac{C_i}{T_i}(s^M + J_i + r_i^M) + M \tag{18}$$

$$\iff s^M(R - \sum_{i=1}^{n} \frac{C_i}{T_i}) = M + RT + \sum_{i=1}^{n} \frac{C_i}{T_i}(J_i + r_i^M) \tag{19}$$

$$\iff R' s^M = M + RT + \sum_{i=1}^{n} \frac{C_i}{T_i} J_i + \sum_{i=1}^{n} \frac{C_i}{T_i} r_i^M \tag{20}$$

4) **Two definitions for a reordering l_i^M and σ:**

$$\forall i \in [\![1, n]\!] : l_i^M \stackrel{def}{=} (q_i^M - 1)T_i - J_i. \tag{21}$$

Remark that $s^M > l_i^M$ (from $0 \leq r_i^M < T_i$ comes $0 \leq T_i q_i^M - (s^M + J_i) < T_i$ and $T_i(q_i^M - 1) - J_i < s^M$). Now, let $\sigma : [\![1, n]\!] \to [\![1, n]\!]$ be a permutation such that the sequence $l_{\sigma(i)}^M$ is non-increasing, *i.e.* $s^M > l_{\sigma(1)}^M \geq l_{\sigma(2)}^M \geq \cdots \geq l_{\sigma(n)}^M$.

5) **Forall $k \in [\![1, n]\!]$ it holds**

$$C_k \left\lceil \frac{s^M + J_k}{T_k} \right\rceil = C_k \left\lceil \frac{l_k^M + J_k}{T_k} \right\rceil + C_k \tag{22}$$

By definition q_k^M is an integer, so $q_k^M - 1$ also is and $q_k^M = \lceil q_k^M - 1 \rceil + 1$. By definition of l_k^M, we then get $q_k^M = \left\lceil \frac{l_k^M + J_k}{T_k} \right\rceil + 1$ hence the result by definition of q_k^M.

6) **Forall $i \in [\![1, n]\!] : s^M \geq l_{\sigma(i)}^M + \sum_{k=1}^{i} \frac{C_{\sigma(k)}}{R}$:**
Let $i \in [\![1, n]\!]$, $\mathbf{S}_i = \{\sigma(1), \ldots, \sigma(i)\}$ and $\bar{\mathbf{S}}_i = [\![1, n]\!] \setminus \mathbf{S}_i$.
From previous relation, for any k in $\mathbf{S}_i$, $C_k \left\lceil \frac{s^M + J_k}{T_k} \right\rceil \geq C_k \left\lceil \frac{l_k^M + J_k}{T_k} \right\rceil + C_k$. But by definition, $(l_{\sigma(m)}^M)_{m \in [\![1,n]\!]}$ is non-increasing sequence, so for all $k \in \mathbf{S}_i$, $l_k^M \geq l_{\sigma(i)}^M$, which yields $C_k \left\lceil \frac{l_k^M + J_k}{T_k} \right\rceil + C_k \geq C_k \left\lceil \frac{l_{\sigma(i)}^M + J_k}{T_k} \right\rceil + C_k$. To conclude

$$\forall k \in \mathbf{S}_i : C_k \left\lceil \frac{s^M + J_k}{T_k} \right\rceil \geq C_k \left\lceil \frac{l_{\sigma(i)}^M + J_k}{T_k} \right\rceil + C_k \tag{23}$$

Now, consider $k \in \bar{\mathbf{S}}_i$. By the definition of l_j^M (cf. proof step 4), $\forall j \in [\![1, n]\!] : s^M > l_j^M$, and in particular, for $j = \sigma(i)$. Then, it holds

$$\forall k \in \bar{\mathbf{S}}_i : C_k \left\lceil \frac{s^M + J_k}{T_k} \right\rceil \geq C_k \left\lceil \frac{l_{\sigma(i)}^M + J_k}{T_k} \right\rceil \tag{24}$$

Summing over eq. (23) and eq. (24), it comes

$$\sum_{k \in \mathbf{S}_i \cup \bar{\mathbf{S}}_i} C_k \left\lceil \frac{s^M + J_k}{T_k} \right\rceil \geq \sum_{k \in \mathbf{S}_i \cup \bar{\mathbf{S}}_i} C_k \left\lceil \frac{l_{\sigma(i)}^M + J_k}{T_k} \right\rceil + \sum_{k \in \mathbf{S}_i} C_k$$

$$\textit{i.e. } V(s^M) \geq V(l_{\sigma(i)}^M) + \sum_{j=1}^{i} C_{\sigma(j)}$$

By the definition of s^M, one has $V(s^M) = R(s^M - T) - M$. Conversely, since $s^M \geq l_{\sigma(i)}^M$, from eq. 14, it comes $V(l_{\sigma(i)}^M) \geq R(l_{\sigma(i)}^M - T) - M$, so

$$R(s^M - T) - M \geq R(l_{\sigma(i)}^M - T) - M + \sum_{j=1}^{i} C_{\sigma(j)} \tag{25}$$

$$\iff s^M \geq l_{\sigma(i)}^M + \sum_{j=1}^{i} \frac{C_{\sigma(j)}}{R} \tag{26}$$

7) **Forall $i \in [\![1, n]\!] : r_{\sigma(i)}^M \leq T_{\sigma(i)} - \sum_{j=1}^{i} \frac{C_{\sigma(j)}}{R}$:** This is a direct consequence of definition of l_i^M, q_i^M and previous relation.

$$s^M \geq l_{\sigma(i)}^M + \sum_{j=1}^{i} \frac{C_{\sigma(j)}}{R} \tag{27}$$

$$\iff s^M \geq (q_{\sigma(i)}^M - 1)T_{\sigma(i)} - J_{\sigma(i)} + \sum_{j=1}^{i} \frac{C_{\sigma(j)}}{R} \tag{28}$$

$$\iff s^M + J_{\sigma(i)} - T_{\sigma(i)} q_{\sigma(i)}^M \geq -T_{\sigma(i)} + \sum_{j=1}^{i} \frac{C_{\sigma(j)}}{R} \tag{29}$$

$$\iff r_{\sigma(i)}^M \leq T_{\sigma(i)} - \sum_{j=1}^{i} \frac{C_{\sigma(j)}}{R} \tag{30}$$

8) $\sum_{i=1}^{n} r_i^M \frac{C_i}{T_i} \leq \sum_{i=1}^{n} \left(T_i - \frac{C_i}{R}\right) \frac{C_i}{T_i} -$

$\frac{1}{R}\sum_{i=1}^{n}\sum_{j=1}^{i-1}\frac{C_{\sigma(i)}C_{\sigma(j)}}{T_{\sigma(i)}}$ **:**

$$\sum_{i=1}^{n} r_i^M \frac{C_i}{T_i} = \sum_{i=1}^{n} r_{\sigma(i)}^M \frac{C_{\sigma(i)}}{T_{\sigma(i)}} \text{ since } \sigma \text{ is a permutation} \tag{31}$$

$$\leq \sum_{i=1}^{n}\left(T_{\sigma(i)} - \sum_{j=1}^{i}\frac{C_{\sigma(j)}}{R}\right)\frac{C_{\sigma(i)}}{T_{\sigma(i)}} \tag{32}$$

$$= \sum_{i=1}^{n}\left(T_{\sigma(i)} - \frac{C_{\sigma(i)}}{R} - \sum_{j=1}^{i-1}\frac{C_{\sigma(j)}}{R}\right)\frac{C_{\sigma(i)}}{T_{\sigma(i)}} \tag{33}$$

$$= \sum_{i=1}^{n}\left(T_{\sigma(i)} - \frac{C_{\sigma(i)}}{R}\right)\frac{C_{\sigma(i)}}{T_{\sigma(i)}} - \frac{1}{R}\sum_{i=1}^{n}\sum_{j=1}^{i-1}\frac{C_{\sigma(j)}C_{\sigma(i)}}{T_{\sigma(i)}} \tag{34}$$

$$= \sum_{i=1}^{n}\left(T_i - \frac{C_i}{R}\right)\frac{C_i}{T_i} - \frac{1}{R}\sum_{i=1}^{n}\sum_{j=1}^{i-1}\frac{C_{\sigma(j)}C_{\sigma(i)}}{T_{\sigma(i)}} \tag{35}$$

The next step consists in having a lower bound on $\sum_{i=1}^{n}\sum_{j=1}^{i-1}\frac{C_{\sigma(j)}C_{\sigma(i)}}{T_{\sigma(i)}}$.

9) $\sum_{i=1}^{n}\sum_{j=1}^{i-1}\frac{C_{\sigma(j)}C_{\sigma(i)}}{T_{\sigma(i)}} \geq \max\{Q, L\}$ **The goal in this step is to get rid of the σ permutation, since it depends on M. :**

a) $\sum_{i=1}^{n}\sum_{j=1}^{i-1}\frac{C_{\sigma(j)}C_{\sigma(i)}}{T_{\sigma(i)}} \geq L$ **:**

$$\sum_{i=1}^{n}\sum_{j=1}^{i-1}\frac{C_{\sigma(i)}C_{\sigma(j)}}{T_{\sigma(i)}} \geq \sum_{i=1}^{n}\sum_{j=1}^{i-1}\frac{C_{\sigma(i)}\min_{k\in[\![1,n]\!]}C_k}{T_{\sigma(i)}}$$
$$= \min_{k\in[\![1,n]\!]}C_k\sum_{i=1}^{n}\frac{C_{\sigma(i)}}{T_{\sigma(i)}}\times(i-1)$$
$$\geq \min_{k\in[\![1,n]\!]}C_k\sum_{i=1}^{n}\frac{C_{\sigma(i)}}{T_{\sigma(i)}}\times\min(i-1,1)$$

and since one does not know the value of $\sigma(1)$ (*i.e.* when $i-1=0$)

$$\geq \min_{k\in[\![1,n]\!]}C_k\left(\sum_{i=1}^{n}\frac{C_i}{T_i} - \max_{i\in[\![1,n]\!]}\frac{C_i}{T_i}\right)$$
$$= L$$

b) $\sum_{i=1}^{n}\sum_{j=1}^{i-1}\frac{C_{\sigma(i)}C_{\sigma(j)}}{T_{\sigma(i)}} \geq \sum_{i=1}^{n}\sum_{j=1}^{i-1}\frac{C_{\sigma(i)}C_{\sigma(j)}}{T_{\sigma(i)}T_{\sigma(j)}}\min\left\{T_{\sigma(i)}, T_{\sigma(j)}\right\}$ **:**

$$\sum_{i=1}^{n}\sum_{j=1}^{i-1}\frac{C_{\sigma(i)}C_{\sigma(j)}}{T_{\sigma(i)}} = \sum_{i=1}^{n}\sum_{j=1}^{i-1}\frac{C_{\sigma(i)}C_{\sigma(j)}}{T_{\sigma(i)}T_{\sigma(j)}}T_{\sigma(j)}$$
$$\geq \sum_{i=1}^{n}\sum_{j=1}^{i-1}\frac{C_{\sigma(i)}C_{\sigma(j)}}{T_{\sigma(i)}T_{\sigma(j)}}\min\left\{T_{\sigma(i)}, T_{\sigma(j)}\right\}$$

c) $\sum_{i=1}^{n}\sum_{j=1}^{i-1}\frac{C_{\sigma(i)}C_{\sigma(j)}}{T_{\sigma(i)}T_{\sigma(j)}}\min\left\{T_{\sigma(i)}, T_{\sigma(j)}\right\} = \sum_{p=1}^{n}\sum_{q=1}^{p-1}\frac{C_pC_q}{T_pT_q}\min\{T_p, T_q\} = Q$**:**

Let $x_{i,j} \overset{def}{=} \frac{C_iC_j}{T_iT_j}\min\{T_i, T_i\}$, and $X \overset{def}{=} \left\{(i,j)\in[\![1,n]\!]^2 \;\middle|\; i>j\right\}$, and also

$$h:[\![1,n]\!]^2 \to [\![1,n]\!]^2$$
$$(i,j)\mapsto \begin{cases}(\sigma(i),\sigma(j)) \text{ when } (i>j)=(\sigma(i)>\sigma(j))\\(\sigma(j),\sigma(i)) \text{ otherwise.}\end{cases}$$

Note that for all i, j, we have $(i,j)\in X$ if and only if $h(i,j)\in X$ and $x_{\sigma(i),\sigma(j)} = x_{h(i,j)}$ since x is symmetric. We thus have

$$\sum_{i=1}^{n}\sum_{j=1}^{i-1}\frac{C_{\sigma(i)}C_{\sigma(j)}}{T_{\sigma(i)}T_{\sigma(j)}}\min\left\{T_{\sigma(i)}, T_{\sigma(j)}\right\}$$
$$= \sum_{(i,j)\in X} x_{(\sigma(i),\sigma(j))}$$
$$= \sum_{h(i,j)\in X} x_{h(i,j)}$$

One can then prove that h is injective, meaning it is bijective, which enables the following reindexing

$$\sum_{h(i,j)\in X} x_{h(i,j)} = \sum_{(i,j)\in X} x_{(i,j)} = Q$$

10) $s^M \leq \frac{M+C}{R'}$**:** This is just, going from equations (20), application of steps 8 and 9.

$$R's^M = M + RT + \sum_{i=1}^{n}\frac{C_i}{T_i}J_i + \sum_{i=1}^{n}\frac{C_i}{T_i}r_i^M$$
$$\leq M + RT + \sum_{i=1}^{n}\frac{C_i}{T_i}J_i + \sum_{i=1}^{n}\left(T_i - \frac{C_i}{R}\right)\frac{C_i}{T_i} - \frac{1}{R}\sum_{i=1}^{n}\sum_{j=1}^{i-1}\frac{C_{\sigma(i)}C_{\sigma(j)}}{T_{\sigma(i)}}$$
$$\leq M + RT + \sum_{i=1}^{n}\left(J_i + T_i - \frac{C_i}{R}\right)\frac{C_i}{T_i} - \frac{\max\{Q,L\}}{R}$$
$$\implies s^M \leq \frac{M+C}{R'}$$

11) **Here comes the M elimination:** Let $t\in\mathbb{R}^+$.
If $t \leq \frac{C}{R'}$, $\beta_{R',\frac{C}{R'}}(t) = 0$, so $\beta_{R',\frac{C}{R'}}(t) \leq [\beta_{R,T} - V]^+_{\uparrow}(t)$ trivially holds.
If $t \geq \frac{C}{R'}$. By definition of s^M, for any $M\in\mathbb{R}$,

$$M = R(s^M - T) - V(s^M) \tag{36}$$

so, for any I^M such that $s^M \in I^M$

$$M \leq \sup_{u\in I}\{R(u-T) - V(u)\}. \tag{37}$$

Set $M = R't - C$ (this can be done safely since there is no hidden M in $R, R', T, V(\cdot)$). From step 10, $s^M \leq \frac{M+C}{R'} = t$, so

$$R't - C \leq \sup_{s^M\leq u\leq t}\{R(u-T) - V(u)\}. \tag{38}$$

But from $t\geq\frac{C}{R'}$ and $M = R't - C$ comes $M\geq 0$, and introducing it in eq. 15 yields $s^M \geq 0$, so

$$R't - C \leq \sup_{0\leq u\leq t}\{R(u-T) - V(u)\}, \tag{39}$$

and by doing the maximum with 0

$$R'\left[t-\frac{C}{R'}\right]^+ \leq \sup_{0\leq u\leq t}\left[R(u-T)-V(u)\right]^+ \quad (40)$$

$$\iff \beta_{R',\frac{C}{R'}}(t) \leq \left[\beta_{R,T}-V\right]^+_\uparrow(t) \quad (41)$$

Appendix B
Coq statement of Theorem 3

While the Coq compiler checks that a theorem is well formed and that its proof is correct, it can not check that the theorem is conform to the author or reader intuition. We will then describe the formal statement of Theorem 3 in Coq's language.

The full proof is available, along with instructions to automatically recheck it with Coq, at http://doi.org/10.5281/zenodo.3881823 .

First comes the loading of the libraries.

```
Require Import mathcomp.(*...*).
Require Import Reals (*...*).
```

and Coq is instructed to interpret all standard notations, such as $+, -, \leq$, as real number ones

```
Local Open Scope R_scope.
```

We then give the hypotheses of the theorem

```
Section Theorem3.

Variable n' : nat.
Notation n := n'.+1.
(* Be sure that n is non zero *)

Variable R T : R+*.
Variable tC tT : R+* ^ n.
Variable tJ : R+ ^ n.
```

For convenience, the `i`-th element `(tC i)` of the `n`-tuple `tC` will then be denoted `C`_i`

```
Notation "'C`_' i" := (tC i).
Notation "'T`_' i" := (tT i).
Notation "'J`_' i" := (tJ i).

Hypothesis R_large_enough :
  \sum_i C`_i / T`_i < R.
```

And we define the various constants and functions

```
Definition R' := R - \sum_i C`_i / T`_i.
Definition W :=
  \sum_i (T`_i + J`_i - C`_i / R) * (C`_i / T`_i).
Definition L := (\min_k C`_k)
  * ((\sum_i C`_i / T`_i) - \max_i (C`_i / T`_i)).
Definition Q :=
  \sum_(i < n) \sum_(j < n | j < i)
    Rmin T`_i T`_j * (C`_i * C`_j) / (T`_i * T`_j).
Definition C := R * T + W - Rmax L Q / R.
Definition V t := \sum_i
  C`_i * IZR (Zceil ((t + J`_i) / T`_i)).
Definition beta R T :=
  fun t : R+ => R * '[ t - T ]+.
```

Before finally stating the theorem itself

```
Theorem theorem3 : forall t,
(beta R' (C / R') t
≤ '[fun t => beta R T t - V t]+^ t)%Rbar.
```

where `%Rbar` tells Coq that $\leq$ is the one on $\overline{\mathbb{R}} = \mathbb{R} \cup \{-\infty, +\infty\}$ since the non decreasing closure contains a least upper bound that could be infinite.