A Correlation Preserving Performance Analysis for Stream Processing Systems

Gideon Smeding, Gregor Goessler

A Correlation Preserving Performance Analysis for Stream Processing Systems

Gideon Smeding
Pop Art project
INRIA, France

Gregor Gössler
Pop Art project
INRIA, France

***Abstract*—For the design of real-time embedded systems, analysis of performance and resource utilization at an early stage is crucial to evaluate design choices. Network Calculus and its variants provide the tools to perform such analyses for distributed systems processing streams of tasks, based on a max-plus algebra. However, the underlying model employed in Network Calculus cannot capture correlations between the availability of different resources and between the arrivals of tasks, leading to overly conservative performance bounds for some frequently used system topologies. We present a model based on timing constraints relative to pairs of streams, endowed with an analysis technique that can handle such correlations.**

I. Introduction

Modern stream-processing systems, such as multimedia applications and embedded automatic control systems, are often realized on heterogeneous, multi-processor architectures. Such architectures have a large design space, due to the choice in processors, the networks to connect them, the partitioning of software on the hardware, and the choice of scheduling regimes to allocate resources. As development progresses, architecture changes become increasingly expensive. It is therefore essential to evaluate the feasibility of a design at an early stage. In particular, the performance characteristics of a design, such as the throughput and required buffer sizes needed to fulfill the system requirements.

Let us review two illustrative examples, that, by the end of the paper, we will be able to fully analyse. For now, we will only informally introduce them. The exact meaning of the components in the examples will be introduced in the sequel.

The first example, depicted in Figure 1, models a multimedia decoder that decodes audio and video parts of an incoming multimedia stream separately and then joins the decoded streams at the output.

Each double arrow in the diagram represents a lossless channel with FIFO semantics (a stream). The PJD and BD denote sources of streams and resources, respectively. The GPC component denotes the actual decoding of audio or video streams consuming processor resources and producing a stream of decoded frames. The AND component synchronously joins the two streams by combining incoming audio and video frames in the output.

A second example, depicted in Figure 2, shows a server that is connected to the outside world by a bus. The server responds to update requests that arrive over a bus and returns a reply over the very same bus. Conflict-free access to the bus from both sides is achieved through a time-division, multiple-access (TDMA) policy. That is, exclusive access to the bus is given periodically to the requester and the server. The server has a certain delay before it responds. If this delay corresponds to the TDMA period, then the bus does not introduce any additional delay since the reply to a received request can immediately be transmitted.

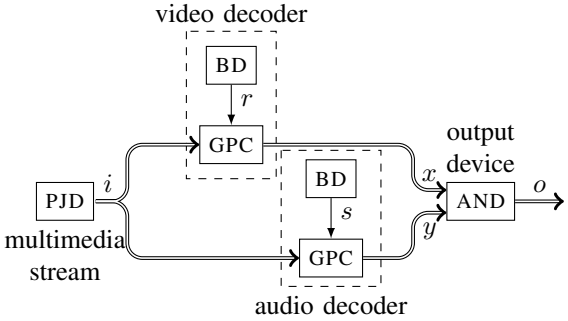


Fig. 1. The greedy processing components (GPC) model the decoding of the input stream i, arriving periodically with bounded jitter and minimum inter-arrival time (PJD), on two separate processors, utilizing resources r and s with fixed bandwidth and bounded delay (BD), and the AND component joins the decoded streams resulting the output stream o.

Two new components are used here: the TDMA component that splits a resources according to a fixed schedule, and the DEL component that imposes a fixed delay on a stream. In this example, the GPC components do not model processors or decoders, but the transmission of messages over the bus.

The performance of systems as the ones sketched above can be analyzed by different approaches. Simulation-based techniques are efficient but in general non exhaustive, and therefore may not reveal corner cases. On the other hand, formal analytical methods can offer strong guarantees for safety-critical systems. In this paper we present such an analytical method to discover absolute performance bounds.

Existing formal verification techniques for real-time systems are based on model-checking of timed automata [9], algebraic techniques like Network calculus [12] and real-time calculus [3], scheduling theory [8], or combinations thereof [10].

In network calculus streams of events, such as the arrival of consecutive frames of data to be processed or the periodic

Accepted for publication at MEMOCODE 2012.
Research supported by the Chair INRIA-Schneider, associated with Digiteo.

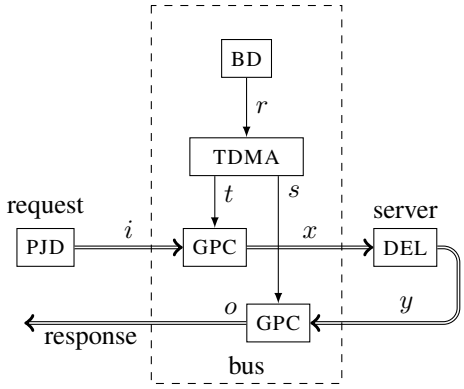


Fig. 2. The stream i of update requests passes a bus, represented by resource r, to arrive at a server, which, after a fixed delay (DEL), sends a reply y back over the bus. The resource is shared with a fixed TDMA schedule, over the resources t and s.

activation of a task, are modelled as a cumulative function R over time. While one such function represents one concrete execution trace, a "curve" α characterizes an abstract event stream of all behaviors R, such that for all time instants t and u where $t \leq u$

$$R(t) - R(u) \leq \alpha(t-u)$$

The availability of a resource is modelled by a similar cumulative function C and abstraction β such that $\beta(t-u) \leq C(t) - C(u)$.

Given a processor with the resource characterized by a curve β that handles tasks characterized by α that arrive over a FIFO queue, network calculus shows that delay is bounded by the maximum horizontal distance between α and β and backlog, i.e., the maximum number of queued tasks at any time during execution, is bounded by their vertical distance. Moreover, it permits us to calculate the curve α' that characterizes the stream of output events that correspond to the completion of tasks. This forms the basis of a compositional analysis for networks of processing components.

Unfortunately, the abstraction of streams can lead to pessimistic bounds for some systems, as is the case for our examples.

In the multimedia decoder of Figure 1, the application of network calculus, with an AND component as described in [7], will overestimate the needed buffer space, in particular the inputs x and y of the AND component. This originates in the model of streams: while the maximum number of arrivals of each individual stream are exact, the model does not show that peaks in the number of arrivals occur simultaneously in x and y. The AND component's inputs are considered only independently, thus leading to a pessimistic estimate of the output's variability.

In the example of Figure 2, even if the arrivals and resources are constant, network calculus will overestimate the needed buffer size between the server and the bus, because the cause-effect relation of the request and response in combination with the strict schedule of the TDMA cannot be modelled.

We propose an analysis based on a relative event model, where events and resource availability are modelled by functions that give the cumulative number of events at the time of each occurrence of another event. This leads to an event model that preserves correlations between events, and allows a natural integration of an abstraction similar to network calculus' curves, with bounds that model causal relations such as data-flow constraints.

A. Contributions

The main novelty presented in this paper is an event model based on a combination of two kinds of bounds on relative counter functions: simple *clock bounds* to model e.g. data-flow dependencies and *drift bounds* that limit relative variability to model e.g. burstiness of a flow. We derive general properties of such bounds and properties associated with the processing components, leading to an analysis method based on a fixpoint algorithm. Finally, we provide experimental evidence that our relative-time framework can find better bounds for the needed buffer space in the above examples.

B. Previous Work

Data-flow networks are represented naturally through relational constraints. The first such a notion is synchronic distance [14] for Petri nets. The synchronic distance is the maximum number of times a transition may fire before another must be fired. If the Petri net models a data-flow system, typically modelled by conflict-free nets, the synchronous distance of a producer from a consumer (both modelled by transitions) corresponds to the maximum buffer occupation between them.

The affine clock calculus used in the validation and compilation of real-time applications [16] programmed with a combination of SIGNAL and ALPHA, The calculus relates two clocks through a base clock. Typically, the arrival of complex tasks is related to their completions through a system clock that determines execution speed. These relations are expressed by affine transformations over the occurrence-times of events. The calculus then serves to determine synchronizability of two event streams.

The more recent work on n-synchrony [5] develops a similar concept to verify synchronizability of programs written in a synchronous data-flow language extended with statically scheduled sample (a periodic selection of elements in a flow) and merge (a combination of two flows defined by a static schedule) operations and buffers with FIFO semantics. The verification is based on a type-system and determines whether the system can be executed with finite buffer sizes. The clock envelopes introduced in [4] further develop this system, using clock abstractions called envelopes that permit more efficient verification at the cost of some over-estimated buffer sizes.

The clock bounds, introduced in this paper, express the same essential relations as the data-flow relations of the mentioned work. The formalization of the relations as bounds on relative counter functions however, is new. And it is precisely this formalization that allows us to combine clock bounds with

drift bounds. Clock bounds are more general than the synchronic distance on conflict-free nets, affine relations and clock envelopes, all of which can be represented by linear clock bounds in our model. Clock bounds are equal in expressiveness to the basic model of n-synchrony.

Drift bounds are essentially a relational variant of network calculus' arrival and resource curves. The problem of correlating variability of event streams, such as coincident bursts of arrival streams, has been studied before in various incarnations of network calculus.

In [15], [13] the case where streams are transported over a network as a single, joined stream, to be separated at arrival, is treated by tracking the correlations between sub-streams and the aggregated stream. The event count curves of [13] are a special case of our drift bounds: a bound on a sub-stream with respect to the aggregate stream.

Correlated streams are also used in [11] for the analysis of a fork-join scenario, where the load of a single event stream is split over several processing components and then merged in a scenario similar to the multimedia decoder of Figure 1. However, [11] distribute frames over the processors for load distribution, like splitting traffic over two lanes and then merging it again, rather than the synchronous split join of our example.

Finally [19] exploits correlations between the workload imposed by a task traversing a chain, which occur e.g. for tasks with variable payloads sizes that traverse a chain of processors: tasks that require many resources on the first processor because of a large payload, will also require more work for the next and vice versa. None of the above approaches uses a general relational model as presented in this paper.

II. A Relative Model of Clocks

This section introduces the behavioral model of systems, which, like network calculus, is based on cumulative functions. Then it introduces a first abstraction that relates all clocks to each other rather than to real-time.

In the model *streams* of periodically recurring events as *clocks*. For example, the arrival of audio frames in a multimedia stream is represented by a clock that ticks once for each arriving frame. If a resource is a processor, then each clock tick corresponds to a cycle that would be used to process a task. If the resource is a bus, each tick corresponds to the transfer of a task. The behavior of clocks is described by *counter functions*.

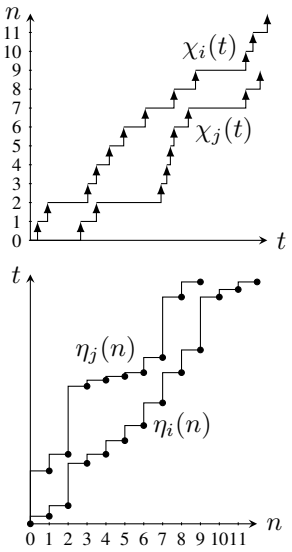
Definition 1 (Counter function (χ))**.** *Let $\chi \in \mathbb{R} \to \mathbb{N}$ be a counter functions such that it*

- *starts at zero ($\chi(0) = 0$);*
- *is non-decreasing ($t \leq u \implies \chi(t) \leq \chi(u)$); and*
- *is surjective ($range(\chi) = \mathbb{N}$).*

Equivalently, we could choose to describe clocks by *dater functions* that give the time at which each tick occurs.

Definition 2 (Dater function)**.** *The dater functions $\eta \in \mathbb{N} \to \mathbb{R}$ corresponding to the counter function χ is defined as*

$$\eta(n) = \min\{t \in \mathbb{R} \mid \chi(t) \geq n\}$$



n	$\eta_i(n)$	$\eta_j(n)$
0	0.00	0.00
1	1.00	7.00
2	2.40	9.20
3	8.00	18.2
4	9.20	19.0
5	11.0	19.5
6	13.0	20.0
7	16.0	30.0
8	20.0	32.0
9	23.0	
10	30.0	
11	31.0	
12	32.0	

Fig. 3. Two (partial) counter and corresponding dater functions and the exact times of each tick. Ticks 4 and 2 of respectively i and j occur simultaneously, as do the pairs (8, 6), and (10, 7).

Figure 3 shows two counter functions and the corresponding dater functions. It also shows in a table how the dater function provides a trace of timestamps for each successive tick.

Lemma 1 (Properties of dater functions)**.** *The dater function η starts at zero ($\eta(0) = 0$), is strictly increasing ($n < m \implies \eta(n) < \eta(m)$), is total ($domain(\eta) = \mathbb{N}$) and diverges ($\lim_{n\to\infty} \eta(n) = \infty$).*

Proof: As zero is an eliminator for minimization over the domain of positive numbers, the dater function must also start at zero. By contradiction it must be strictly increasing; if there were some $n < m$ such that $\eta(n) \geq \eta(m)$, then we arrive at the contradiction that χ is decreasing. Totality is a consequence of the surjectivity of the counter function, which can be shown through contradiction. Finally, the dater function diverges, because the minimum also diverges if the time and the tick count go to infinity. ∎

The dater function is not a proper inverse of the counter function because in general, $\eta \circ \chi \neq id$. However, we do have the inequalities $\eta(\chi(t)) \leq t$, $\eta(\chi(t) + 1) \geq t$, and $\chi(\eta(n)) = n$ for all times t. This can be understood from the informal description of counter and dater functions: $\chi(t)$ gives the number of ticks of i up to, and including, time t, and $\eta(n)$ gives the time when the n-th tick occurs.

In our model we concentrate on the logical behavior of clocks, as opposed to the temporal behavior. That is, how ticks of different clocks interleave with one-another. To this end, the clocks are modelled by the relations between them, concentrating on the relative order of events rather than their absolute timing.

Given two clocks i and j described by their counter functions χ_i and χ_j, their relative clock $X_{i/j}(n)$ gives the tick count of i at the n-th tick of j. Figure 4 depicts the relative counter function $R_{i/j}$ as well as $R_{j/i}$ from the counter and

dater functions depicted in Figure 3.

Definition 3 (Relative counter function ($X_{i/j}$))**.** *Let χ_i and η_j respectively be the counter and dater functions of two clocks i and j, then the relative counter function $X_{i/j} \in \mathbb{N} \to \mathbb{N}$ is defined as*

$$X_{i/j}(n) = (\chi_i \circ \eta_j)(n)$$

As the notation already gives away, the behavior of a whole system, consisting of a set K of clocks, is described by a column vector $\boldsymbol{\chi} \in (\mathbb{R} \to \mathbb{N})^{K\times 1}$ and the corresponding row vector of dater functions $\boldsymbol{\eta} \in (\mathbb{N} \to \mathbb{R})^{1\times K}$. Thus we naturally obtain a matrix $X \in (\mathbb{N} \to \mathbb{N})^{K\times K}$ of relative counter functions:

$$X = \begin{pmatrix}\chi_1\\ \vdots \\ \chi_n\end{pmatrix} \circ (\eta_1 \cdots \eta_n) = \begin{pmatrix}\chi_1 \circ \eta_1 & \cdots & \chi_1 \circ \eta_n \\ \vdots & \ddots & \vdots \\ \chi_n \circ \eta_1 & \cdots & \chi_n \circ \eta_n\end{pmatrix}$$

The diagonal of the relative counter function matrix, consists of identity functions since $X_{i/i}(n) = \chi_i(\eta_i(n)) = n$ for all clock i. A note on notation: anywhere where $X_{i/j}$ is used, it can be replaced directly by the composition $\chi_i \circ \eta_j$.

Relative counter functions inherit most of the counter functions' properties. The most important difference being that relative counter functions can increase by more than one in a single step. For example, in Figure 4 $X_{i/j}$ jumps from 4 to 7, because i ticks three times between the second and third ticks of j.

We can also see that only the (non-strict) order of ticks is preserved, in the sense that if e.g. $X_{i/j}(2) = 4$ we do not know if the fourth tick of i occurred simultaneously with the second tick of j, or if it preceded the second tick of j.

Lemma 2 (Properties of relative counter functions)**.** *Like normal counter functions, relative counter functions start at zero ($X_{i/j}(0) = 0$), are non-decreasing ($n \leq m \implies X_{i/j}(n) \leq X_{i/j}(m)$) and diverge.*

Proof: That $X_{i/j}(0) = 0$ can be calculated directly. Divergence and non-decreasingness are both preserved by the composition of the non-decreasing and divergent counter and dater functions. ∎

As a relative counter function $X_{i/j}$ characterizes the behavior of χ_i with respect to χ_j, we can compute a *pseudo-inverse* $X^{-1}_{i/j}$ characterizing the behavior of χ_j with respect to χ_i.

Definition 4 (Pseudo-inverse)**.** *The pseudo-inverse $X^{-1}_{i/j} \in \mathbb{N} \to \mathbb{N}$ of a relative counter function $X_{i/j} \in \mathbb{N} \to \mathbb{N}$ is defined as*

$$X^{-1}_{i/j}(n) = \min\{m \in \mathbb{N} \mid X_{i/j}(m+1) \geq n\}$$

We define the pseudo-inverse X^{-1} of a matrix as the pointwise pseudo-inversion and a transposition, i.e., $[X^{-1}]_{i/j} = X^{-1}_{j/i}$. Subscription takes precedence over the inversion operator.

Lemma 3 (Properties of the pseudo-inverse)**.** *The pseudo-inverse $X^{-1}_{i/j}$ has the same properties as a relative counter function. That is, $X^{-1}_{i/j}$ goes through the origin, is non-decreasing and diverges. Furthermore, the inversion operator is symmetric ($[f^{-1}]^{-1} = f$) and antitone ($f(n) \leq g(n) \iff f^{-1}(n) \geq g^{-1}(n)$ for monotonic f and g).*

Furthermore, for any $n \in \mathbb{N}$, (1) $X^{-1}_{i/j}(X_{i/j}(n)) < n$, (2) $X_{i/j}(X^{-1}_{i/j}(n)) < n$ for $n > 0$, (3) $X^{-1}_{i/j}(X_{i/j}(n)+1) \geq n$, and $X_{i/j}(X^{-1}_{i/j}(n)+1) \geq n$.

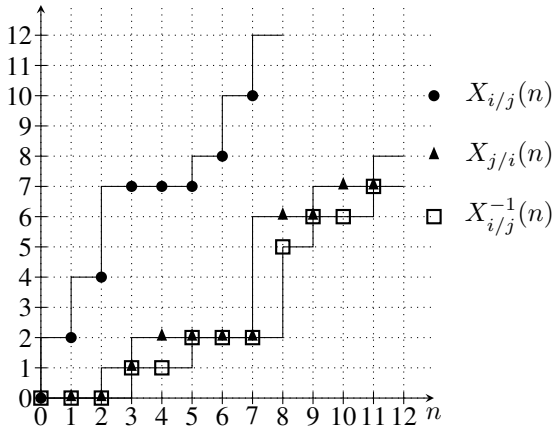


Fig. 4. The relative counter functions $X_{i/j}$ and $X_{j/i}$ created by composing the counter and dater functions of Figure 3, and the pseudo-inverse $X^{-1}_{i/j}$. Note that $X_{j/i}(n) = X^{-1}_{i/j}(n) + 1$ occurs exactly when the clocks tick simultaneously.

Intuitively we might expect the pseudo-inverse $X^{-1}_{i/j}$ to relate strongly to $X_{j/i}$ and maybe even be equal. The example in Figure 4 reinforces this, but it also shows that, for the few instances where the clocks i and j tick simultaneously, $X^{-1}_{i/j} \neq X_{j/i}$. As the following lemma shows equality almost holds, with a difference of one.

Lemma 4 (Inverse of relative counter function matrix)**.** *Let X be a matrix of relative counter functions, then $X^{-1} \leq X \leq X^{-1} + 1$.*

Proof: We will prove this by deriving equal upper and lower bounds of the converse. First note that $X^{-1}_{i/j}(n)$ $= \min\{m \in \mathbb{N} \mid (\chi_i \circ \eta_j)(m+1) \geq n\} = \min\{m \in \mathbb{N} \mid \eta_j(m+1) \geq \eta_i(n)\}$ because $\chi_i(\eta_i(n)) = n$ and $\eta_i(\chi_i(t)) \leq t$ imply $\chi_i(x) \geq y \Leftrightarrow x \geq \eta_i(y)$. Then, since $\eta_j(\chi_j(t)+1) \geq t$ implies $x \geq \chi_j(y) \Rightarrow \eta_j(x+1) \geq y$, we have the lower bound

$$\begin{aligned}&\min\{m \in \mathbb{N} \mid \eta_j(m+1) \geq \eta_i(n)\}\\ \leq &\min\{m \in \mathbb{N} \mid m \geq (\chi_j \circ \eta_i)(n)\}\\ = &\min\{m \in \mathbb{N} \mid m \geq X_{j/i}(n)\}\\ = &X_{j/i}(n)\end{aligned}$$

and, since $\chi_j(\eta_j(n)) = n$ implies $\eta_j(x) \geq y \Rightarrow x \geq \chi_j(y)$,

$$\begin{aligned}&\min\{m \in \mathbb{N} \mid \eta_j(m+1) \geq \eta_i(n)\}\\ \geq &\min\{m \in \mathbb{N} \mid m+1 \geq (\chi_j \circ \eta_i)(n)\}\\ = &\min\{m \in \mathbb{N} \mid m \geq X_{j/i}(n) - 1\}\\ = &X_{j/i}(n) - 1\end{aligned}$$

∎

The most interesting property of relative counter functions is their transitivity: the relative counter functions $X_{i/j}$, $X_{i/k}$ and $X_{k/j}$ of any three clocks i, j and k are not independent.

Lemma 5 (Transitivity of relative counter functions)**.** *The matrix X of relative counter functions is transitively closed, i.e., for all clocks i, j and k in K*

$$X_{i/k} \circ X_{k/j} \leq X_{i/j} \leq X_{i/k} \circ (X_{k/j} + 1)$$

Proof: The proof of the lower bound follows from the fact that function composition is associative and that $(\eta_k \circ \chi_k)(t) \leq t$. Thus we derive $X_{i/k} \circ X_{k/j} = \chi_i \circ \eta_k \circ \chi_k \circ \eta_j \leq \chi_i \circ \eta_j = X_{i/j}$. The upper bound is proven similarly, but now $\eta_k(\chi_k(t) + 1) \geq t$, and therefore $X_{i/k}(X_{k/j}(n) + 1) = \chi_i(\eta_k(\chi_k(\eta_j(n)) + 1)) \geq \chi_i(\chi_j(n)) = X_{i/j}(n)$. ■

III. Modeling Clock Variability

Relative counter functions are a first step towards an abstract representation of streams. In this section we introduce bounds as a further abstraction to describe the behavior of clocks. More precisely we introduce two kinds of bounds: (1) clock bounds on the relative clocks to model for example data-flow dependencies between processes; and (2) drift bounds reminiscent of network calculus' curves, to model variability in the availability of resources or arrivals of tasks, for example the (relative) drift of processor's clocks or the availability of resources. We use clock and drift bounds as a finite representation of an infinite number of clock vectors.

A. Clock Bounds

Clock bounds limit the total number of ticks at every tick of another clock, e.g., the upper clock bound $C^u_{i/j}(n) = m$ states that at the n-th tick of j, i may have ticked at most m times. For example, the BD component, that describes the availability of the bus and decoders in the examples, is represented by a pair of upper and lower bounds on the relative counter function $X_{r/k}$ as depicted in Figure 5.

Definition 5 ($\mathcal{F}$)**.** *Let $\mathcal{F}$ be the set of functions $f \in (\mathbb{N} \cup \{\infty\}) \to (\mathbb{N} \cup \{\infty\})$ s.t. $f(0) = 0$ and f is non-decreasing.*

$\mathcal{F}$ is essentially the set of relative counter functions, extended with limits.

Definition 6 (Clock bound matrix C)**.** *The matrices $C^l, C^u \in \mathcal{F}^{K \times K}$ are the lower resp. upper* clock bounds *of a clock vector $\boldsymbol{\chi}$ of clocks K if for all clocks i and j, and $n \in \mathbb{N}$,*

$$C^l_{i/j}(n) \leq X_{i/j}(n) \leq C^u_{i/j}(n)$$

Clock bounds can be used to model general difference bounds such as $\chi_i(t) - \chi_j(t) \geq B$ through $C^u_{i/j}(n) = B + n$ as well as relative clock rates such as $\chi_i(t) \leq R\chi_j(t)$ through $C^u_{i/j}(n) = Rn$, or combinations of both difference bounds and relative rates. Even rates or difference bounds that change over time can be modelled, as long as the behavior is eventually periodic.

In similar work, such as the clock envelopes [4] and the affine clock calculus [16], $\mathcal{F}$ is often restricted to linear or convex functions. The bounds in n-synchronous programming [5], [4] including the clock envelope abstraction, have the additional restriction that $f(n+1) - f(n) \leq 1$, because clocks are only compared with their master clock in the clock hierarchy.

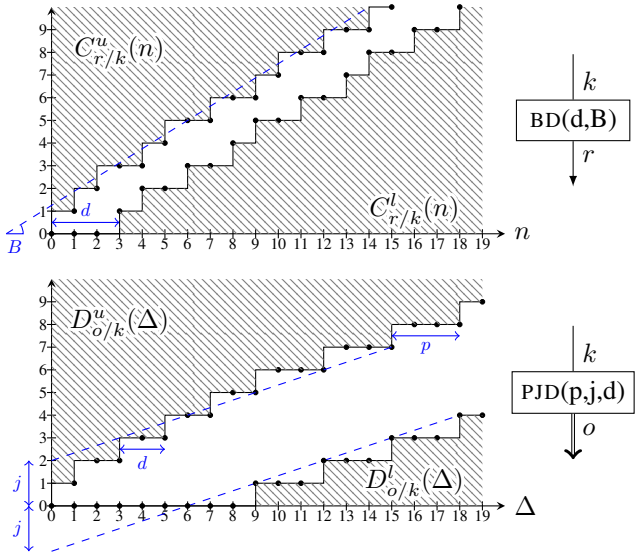


Fig. 5. *Top:* the clock bounds for the resource r resulting from a BD component with bandwidth B and delay d. *Bottom:* the drift bounds for an event stream o with period p, maximum jitter j and minimum inter-arrival time d.

B. Drift Bounds

Clock bounds cannot model time-invariant properties such as a bound on bursts or jitter. For those, we introduce drift bounds. Drift bounds limit the increase of ticks of e.g. $X_{i/j}$ for different intervals of $\chi_j(t)$. That is, the upper drift bound $D^u_{i/j}$ on the relative counter function $X_{i/j}$ requires that i ticks at least $D^u_{i/j}(\Delta)$ times for every Δ ticks of j.

Drift bounds limit the maximum incline for each interval, e.g., a bound $D^u_{i/j}(\Delta) = m$ states that i may tick at most m times, between any Δ consecutive ticks of j. The constraint imposed by a drift bound can be interpreted as a sliding window constraint, with multiple window-sizes. This allows us to model, for example, the PJD component also depicted in Figure 5, for an event stream that exhibits short term bursts and jitter, but is asymptotically stable. Moreover, it limits the time between each two consecutive events.

Definition 7 (Drift bound matrix D)**.** *The matrices $D^l, D^u \in \mathcal{F}^{K \times K}$ is a lower resp. upper* drift bound *of a clock vector $\boldsymbol{\chi}$ of clocks K if for all clocks i and j, $n \in \mathbb{N}$, and all interval sizes $\Delta \in \mathbb{N}$,*

$$D^l_{i/j}(\Delta) \leq X_{i/j}(n + \Delta) - X_{j/k}(n) \leq D^u_{i/j}(\Delta)$$

Although, as we will see in Section V, drift and clock bounds are not completely orthogonal, they capture different properties: clock bounds limit the relative tick count of clocks whereas drift bounds limit the variability of the relative tick count.

To our knowledge drift bounds have no parallel in models for data-flow constraints, they can only be compared to the arrival and resource curves of network calculus. The difference between our drift bounds and the curves of network calculus [12] is found in the bounded quantity: not the counter functions in $\boldsymbol{\chi}$, but the relative counter functions in X are bounded. Because the domain of a relative counter function $X_{i/j}$ is not real-time but the logical time of η_j, the domain of drift bounds is not the set of real-time intervals $(t, r]$ but the set of logical time intervals $(\chi_j(t), \chi_j(r)]$.

In the sequel we use $(\mathcal{C}, \mathcal{D})$ to denote a pair of lower and upper clock bounds $\mathcal{C} = (C^l, C^u)$ and drift bounds $\mathcal{D} = (D^l, D^u)$. We write $\boldsymbol{\chi} \models (\mathcal{C}, \mathcal{D})$ ("$\boldsymbol{\chi}$ satisfies C and D") if the clock vector $\boldsymbol{\chi}$ is bounded by the clock bounds $\mathcal{C}$ and drift bounds $\mathcal{D}$.

C. Interpreting Bounds

The bounds can be studied to obtain bounds on the maximum number of items in a buffer, or the minimum resource utilisation. For example, in the case of an audio or video decoder we would be interested in three key figures: the maximum buffer occupancy (backlog) for each decoder, the time between arrival of a frame and the decoded result (delay), and the minimum resource utilization (throughput). This paper only considers backlog.

The number of tokens in the buffer is the difference between the number consumed tokens and the number of produced tokens at any time. This difference translates to the more general concept of backlog, which at any time, is the difference $\chi_i(t) - \chi_o(t)$ between the number of arrivals at i and the number of departures at o. In the model of relative clocks, this translates to the maximum distance from the identity function, because $X_{j/i}(n)$ gives the number of ticks of the clock j the the n-th tick of i.

Lemma 6 (Backlog bound)**.** *Let $(\mathcal{C}, \mathcal{D})$ describe the behavior of some system, then the backlog $b_{i/j}$ for any behavior $\chi \models (\mathcal{C}, \mathcal{D})$ is bounded by*

$$b_{i/j} \leq \max\{n - C^l_{j/i}(n) \mid n \in \mathbb{N}\}$$

Proof: This follows from the observation that the difference $\chi_i(t) - \chi_j(t)$ is greatest when i has just ticked, it therefore suffices to observe the difference at each time instant $\eta_t(n)$ for all $n \in \mathbb{N}$. Thus $b_{i/j} = \max\{X_{i/i}(n) - X_{j/i}(n)) \mid n \in \mathbb{R}\} \leq \max\{n - C^l_{j/i}(n)) \mid n \in \mathbb{R}\}$. ∎

D. An Algebra of Relative Clock and Drift Bounds

Clock and drift bounds, as well as relative clocks, are ordered by the partial order defined by the point-wise comparison.

Definition 8 ($\leq$)**.** *Let $\leq$ be the partial order over $\mathcal{F}$ (Definition 5) such that for all $f, g \in \mathcal{F}$*

$$f \leq g \iff \forall n \in \mathbb{N} : f(n) \leq g(n)$$

With the partial order, we also define the point-wise minimum $\wedge$ and maximum $\vee$ for all $f, g \in \mathcal{F}$ such that

$$(f \vee g)(n) = \max(f(n), g(n))$$
$$(f \wedge g)(n) = \min(f(n), g(n))$$

The inequality expressing the upper drift bound matrix can be rewritten using the min-plus convolution $\otimes$ which is defined, for $f, g \in \mathcal{F}$ as follows

$$(f \otimes g)(n) = \inf\{f(n - \Delta) + g(\Delta) \mid 0 \leq \Delta \leq n\}$$

The convolution $f \otimes g$ is often explained as the minimum of f and g when sliding g over f, i.e., $(f \otimes g) = g + f(0) \wedge g + f(1) \wedge g + f(2) \wedge ...$, for a more thorough discussion we once again refer to [12]. In our case, we use the operator to reformulate the upper bound condition without any quantifications:

$$X \leq X \otimes D^u$$

Alternatively, we may rewrite the inequality in the definition, such that $h(\Delta) \geq f(n + \Delta) - h(\Delta)$ for any $n \in \mathbb{N}$ and any interval $\Delta \in \mathbb{N}$. This leads to the deconvolution operator, defined as

$$(f \oslash g)(\Delta) = \sup\{f(n + \Delta) - g(n) \mid n \in \mathbb{N}\}$$

This enables a third, equivalent expression of the upper bound condition

$$X \oslash X \geq D^u$$

More generally, the deconvolution is the dual of the convolution, i.e, $f \leq g \otimes h$ if, and only if, $f \oslash g \leq h$ for $f, g, h \in \mathcal{F}$.

For the lower bounds both operators have their analogues in the max-plus convolution $\overline{\otimes}$ and deconvolution $\overline{\oslash}$, where the only difference is the interchange of the infimum and supremum, viz.,

$$(f \overline{\otimes} g)(n) = \sup\{f(n - \Delta) + g(\Delta) \mid 0 \leq n \leq \Delta\}$$
$$(f \overline{\oslash} g)(n) = \inf\{f(\Delta + n) - g(\Delta) \mid \Delta \in \mathbb{N}\}$$

These operators have many long established properties [12]. To quickly recapitulate: all operators are associative and order-preserving (or reversing) and convolution, minimum and maximum are commutative and idempotent.

We liberally extend all operators to matrices where the minimum, maximum, convolution and deconvolution are all applied point-wise. For example given the matrices A and B of functions in $\mathcal{F}$

$$[A \otimes B]_{i/j} = A_{i/j} \otimes B_{i/j}$$

The exception is function composition which acts like matrix multiplication with either the minimum or the maximum as addition, giving for matrices $A \in \mathcal{F}^{P \times Q}$ and $B \in \mathcal{F}^{Q \times P}$

$$[A \circ B]_{i/j} = \bigvee\{A_{i/k} \circ B_{k/j} \mid k \in Q\}$$

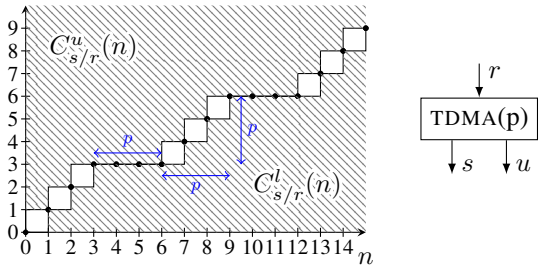


Fig. 6. The clock bound resulting from a TDMA that assigns access to resource r periodically to s with a period of $p = 3$.

IV. Modeling Stream Processing Components

We model a system by composing basic components, each of which defines relations between streams. Stream components are connected by equating streams, e.g., the output of the PJD component that generates the multimedia stream is the input stream of the two GPCs that model the decoders. The components presented here are inspired by components of the MPA-RTC toolbox [3]. In this section we define the behavior of the components informally introduced above and derive bounds on the relative counter functions.

A. Constant bounds

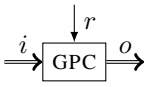
The simplest bounds are constant bound components PJD, BD and TDMA, that directly model the arrivals of input streams, the availability of a resource or indeed the schedule of a resource shared with a TDMA policy. The constant bound components directly impose bounds. The PJD and BD components have already been introduced in Figure 5.

The TDMA component gives access to a resource according to a static schedule. In our case, access is given to one of two users by a simple periodic schedule. For a TDMA component as depicted below, the first p resource units of resource r are assigned to s, the next p units to r, then again p units to s, etc.

The periodic schedule of a TDMA component can in fact be modelled by clock bounds, as is depicted in Figure 6. The equal upper and lower bounds force the clock s to tick synchronously with r for three ticks, and then pause for the same duration while u is given access, etc. The bounds $\mathcal{C}_{u/r}$ can then be defined as the remainder, i.e., $C^u_{u/r} = C^l_{u/r} = id - C^u_{s/r}$.

B. The greedy processing component

The most important component is the GPC which handles tasks according to the availability of a resource, producing the result as another stream. The GPC processes its input i as soon as resources r are available to produce output o.



We formalize the behavior of the GPC, introduced in [17], as a predicate over clock vectors. The original definition also provides a stream of remaining resources, which we have left out for simplicity.

Definition 9 (GPC)**.** *Let* GPC(i, o, r) *be a predicate over a clock vector with input stream i, output stream o, and resources r, that holds if, and only if,*

$$\chi_o(t) = \min\{\chi_i(u) + \chi_r(t) - \chi_r(u) \mid 0 \le u \le t\}$$

Lemma 7 (GPC)**.** *For all clocks k and all behavior $\boldsymbol{\chi}$ such that $\boldsymbol{\chi} \models (C, D)$ and* GPC(i, o, r)

$$C^l_{i/r} \otimes id \le X_{o/r} \le C^u_{i/r} \otimes id$$
$$C^l_{r/i} \wedge id \le X_{o/i} \le C^u_{r/i} \wedge id$$

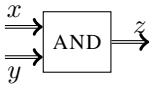
Proof: Let us translate the identity on χ_o to an identity on relative counter function $X_{o/r}$ by substituting real-time parameter t for relative time $\eta_r(n)$. Next, we substitute u by $\eta_r(\Delta)$ which is possible, because the minimum must occur exactly at each tick of r. Finally, $X_{r/r}$ is the identity function.

$$\begin{aligned} & X_{o/r}(n) \\ =& \min\{\chi_i(u) + X_{r/r}(n) - \chi_r(u) \mid 0 \le u \le \eta_r(n)\} \\ =& \min\{X_{i/r}(\Delta) + X_{r/r}(n) - X_{r/r}(\Delta) \mid 0 \le \Delta \le n\} \\ =& \min\{X_{i/r}(\Delta) + id(n - \Delta) \mid 0 \le \Delta \le n\} \\ =& X_{i/r} \otimes id \end{aligned}$$

The upper and lower bounds then follow by substituting X by C^u and C^l respectively. ∎

C. The synchronous join component

The purely logical AND component — logical, because it is not dependent on resources — synchronously *joins* two streams [18].



Essentially the AND component takes one arriving task of each stream x and y and combines them into a single output task on stream z. Combining the tasks is instantaneous, so the total number of events in the outputs stream at any time is the minimum of the total number of events of the input streams.

Definition 10 (AND)**.** *Let* AND(x, y, z) *denote a predicate over clock vectors with input streams x and y and output stream z, such that it holds if, and only if, $\chi_z(t) = \min(\chi_x(t), \chi_y(t))$.*

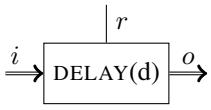
Lemma 8 (AND)**.** *For all clocks $k \in \{x, y, z\}$ and all behavior $\boldsymbol{\chi}$ such that $\boldsymbol{\chi} \models (C, D)$ and* AND(x, y, z) *holds for $\boldsymbol{\chi}$,*

$$C^l_{x/k} \wedge C^l_{y/k} \le X_{z/k} \le C^u_{x/k} \wedge C^u_{y/k}$$

Proof: First we derive a relation on relative counter functions by substituting t by $\eta_k(n)$, which yields $X_{z/k} = X_{x/k} \wedge X_{y/k}$. The upper and lower bounds are then obtained by substituting X by its bounds. ∎

D. The delay component

The DELAY component is a synchronous buffer [5] that holds items for a number of ticks, thus it imposes a fixed delay. The following diagram represents the component that delays events of the input stream i for d ticks of the reference clock r resulting in the output clock o.



Definition 11 (DELAY)**.** *Let* DELAY(i, o, d, r) *denote a predicate over clock vectors with an input stream* i*, an output stream* o *and reference clock* r*, such that it holds if, and only if,* $\chi_o(t) = \chi_i(\eta_r(\chi_r(t) - d))$.

Lemma 9. *For all clock vectors* $\boldsymbol{\chi}$ *such that* $\boldsymbol{\chi} \models (\mathcal{C}, \mathcal{D})$ *and* DELAY(i, o, d, r) *holds for* $\boldsymbol{\chi}$

$$C^l_{i/r}(n-d) \leq X_{o/r}(n) \leq C^u_{i/r}(n-d)$$

Proof: Substituting t by $\eta_r(n)$ yields $X_{o/r}(n) = X_{i/r}(n-d) \leq C^u_{i/r}(n-d)$ because $\chi_r(\eta_r(n)) = n$. ∎

V. Intrinsic Bound Properties

In this section, we identify a number of inequalities that are intrinsic properties of drift bounds; properties that hold for any bounds regardless of the modelled system. They arise from interactions between the different bounds. Later on, the properties will serve to find, given some bounds $(\mathcal{C}, \mathcal{D})$ more precise bounds $(\mathcal{C}', \mathcal{D}')$ that characterize the exact same behaviors. That is, bounds such that $\boldsymbol{\chi} \models (\mathcal{C}, \mathcal{D}) \iff \boldsymbol{\chi} \models (\mathcal{C}', \mathcal{D}')$.

A. Interaction of Lower and Upper Bounds

We use the relation between $X_{i/j}$ and $X^{-1}_{j/i}$ of Lemma 4, to derive lower bounds $C^l_{i/j}$ and $D^l_{i/j}$ from the upper bounds $C^u_{j/i}$ and $D^u_{j/i}$. This is a natural phenomenon: for example, if the clock i may have ticked at most thrice at the second tick of j, we can also conclude that j must have ticked at least twice at the third tick of i.

Theorem 1. *Let* $(\mathcal{C}, \mathcal{D})$ *be a pair of bound matrices, then* $\boldsymbol{\chi} \models (\mathcal{C}, \mathcal{D})$ *implies* $[C^u]^{-1} \leq X \leq [C^l]^{-1} + 1$ *and*

$$X \oslash X \leq [D^l]^{-1} + 1 \text{ and } X \overline{\oslash} X \geq [D^u]^{-1}$$

Proof: In case of the clock bounds, this is a straightforward application of Lemma 4. For the drift bounds, one also needs to apply that $[f \oslash g]^{-1} \geq g^{-1} \overline{\oslash} f^{-1}$ and $[f \overline{\oslash} g]^{-1} \leq g^{-1} \oslash f^{-1}$. ∎

B. Transitivity of Bounds

The next mapping derives directly from the transitivity of relative counter functions. Transitivity propagates bounds through the system.

Theorem 2 (Transitivity of Bounds)**.** *Let* $(\mathcal{C}, \mathcal{D})$ *be a pair of bound matrices then* $\boldsymbol{\chi} \models (\mathcal{C}, \mathcal{D})$ *implies*

$$C^l \circ C^l \leq X \leq C^u \circ (C^u + 1)$$
$$X \oslash X \leq D^u \circ (D^u + 1) \text{ and } X \overline{\oslash} X \geq D^l \circ (D^l - 1)$$

Where $\circ$ *is a matrix multiplication with the point-wise minimum as addition for the upper bounds and point-wise maximum for the lower bounds.*

Proof: Let K be the set of clocks, then application of 5 and subsequent substitution of X by C^u yields $X_{i/j} \leq X_{i/k} \circ (X_{k/j} + 1) \leq C^u_{i/k} \circ (C^u_{k/j} + 1)$ for all $k \in K$ and $\boldsymbol{\chi} \models (\mathcal{C}, \mathcal{D})$. Transitivity of the drift bound follows from the semi-distributivity of function composition over the deconvolution ($f \circ (g \oslash h) \geq (f \circ g) \oslash (f \circ h)$) and linear w.r.t. addition of a constant ($(f + K) \oslash g = (f \oslash g) + K$), viz. $X_{i/j} \oslash X_{i/j} \leq X_{i/k} \circ (X_{k/j} + 1) \oslash X_{i/j} \circ X_{k/j} \leq X_{i/k} \circ ((X + 1) \oslash X)) = X \circ ((X \oslash X) + 1) \leq D^u \circ (D^u + 1)$ for all $\boldsymbol{\chi} \models (\mathcal{C}, \mathcal{D})$. Transitivity for the lower bounds is proven similarly. ∎

C. Drift Bounds Strengthen Clock Bound

The drift bounds limit the incline of any relative counter function, therefore it also has an effect on the clock bound.

Theorem 3 (Drift bound on clock bound)**.** *Let* $(\mathcal{C}, \mathcal{D})$ *be a pair of bound matrices then* $\boldsymbol{\chi} \models (\mathcal{C}, \mathcal{D})$ *implies*

$$D^l \otimes C^l \leq X \leq C^u \otimes D^u$$

Proof: This is a consequence of the satisfaction criteria, namely that $\boldsymbol{\chi} \models (\mathcal{C}, \mathcal{D})$ implies $X \leq X \otimes D^u$ and $X \leq C^u$. Thus, substituting X for $X \otimes D^u$ and subsequently C^u, we have $X \leq X \otimes D^u \leq C^u \otimes D^u$. Idem for the lower bound. ∎

D. Clock Bounds strengthen Drift Bound

A drift bound is limited by the maximum incline that is possible for any relative clock that satisfies the clock bounds. In particular, if the upper and lower clock bounds are very close to one another, there is little room for bursts so the drift bound will resemble a straight line.

Theorem 4 (Clock bound on drift bound)**.** *Let* $(\mathcal{C}, \mathcal{D})$ *be a pair of bound matrices then* $\boldsymbol{\chi} \models (\mathcal{C}, \mathcal{D})$ *implies*

$$X \oslash X \leq C^u \oslash C^l \text{ and } X \overline{\oslash} X \geq C^l \overline{\oslash} C^u$$

Proof: This amounts to a simple substitution of X its bounds $C^l \leq X \leq C^u$, as $\boldsymbol{\chi} \models (\mathcal{C}, \mathcal{D})$ implies $X \leq C^u$ and, by Lemma 4, $X \geq C^l$. Note that the deconvolution operator is monotone for its left operant, but antitone for its right operant, therefore we substitute the left operant by the upper bound and the right operant by the upper bound, which yields $X \oslash X \leq C^u \oslash C^l$. The lower bound is proven analogously. ∎

VI. A Fixpoint Algorithm

The theorems and lemmas derived in the previous two sections all share a common shape: given some behavior delineated by bounds, they compute new bounds for the relative staircase functions and their drift. This section constructs a fixpoint algorithm based on those mappings, that allows us to compute ever tighter bounds and corresponding backlog bounds, for data-flow systems consisting of the introduced components.

A. A lattice of bounds

Informally, bounds $(\mathcal{C}', \mathcal{D}')$ are tighter than bounds $(\mathcal{C}, \mathcal{D})$ if the bounds $(\mathcal{C}, \mathcal{D})$ envelop the bounds $(\mathcal{C}', \mathcal{D}')$. The following bound lattice formalizes the tightness relation as a partial order on bounds.

Lemma 10 (Complete bound lattice)**.** *The partial order of bounds $\mathcal{F}$ ordered by the point-wise comparison $\leq$ is a complete lattice where the join and meet are the point-wise maximum $\wedge$ and minimum $\vee$ respectively. The top $\top$ and bottom $\bot$ are then defined such that $\top(0) = 0$, $\top(n) = \infty$ for $n > 0$, and $\bot(n) = 0$ for all $n \in \mathbb{N} \cup \{\infty\}$.*

The partial order of bounds is extended first to matrices and then to bounds pairs $(\mathcal{C}, \mathcal{D})$ by point-wise applications of the comparison. By [6] this then yields another complete lattice ordered by $\sqsubseteq$ and joined by $\sqcup$. The bottom element of the bound matrix lattice has only upper bounds equal to $\bot$ and lower bounds equal to $\top$, and the reverse for the top element.

Thus, $(\mathcal{C}, \mathcal{D})$ is tighter than $(\mathcal{C}', \mathcal{D}')$ if, and only if $(\mathcal{C}, \mathcal{D}) \sqsubseteq (\mathcal{C}', \mathcal{D}')$. Tightness coincides with the subset relation over behavior: if $(\mathcal{C}', \mathcal{D}')$ is tighter, the set of clock vectors that satisfies $(\mathcal{C}, \mathcal{D})$ contains the set of clock vectors that satisfy $(\mathcal{C}', \mathcal{D}')$. Since the lattice of clock bound matrices is complete, any set of clock vectors has unique tightest bounds.

B. Fixpoint of monotonic mappings

Because all mappings are constructed directly from the inequalities in the theorems, they yield the largest bounds that satisfy the inequalities. More precisely, for each equality we define a mapping M over bounds $(\mathcal{C}, \mathcal{D})$ and rewrite the inequality such that $C'^l \leq X \leq C'^u$, $X \oslash X \leq U'^u$ and $X \overline{\oslash} X \leq U'^l$ for all $\boldsymbol{\chi} \models (\mathcal{C}, \mathcal{D})$ where $(\mathcal{C}', \mathcal{D}') = M(\mathcal{C}, \mathcal{D})$. For example, the mapping for a delay $delay\langle i, o, d, r\rangle(\mathcal{C}, \mathcal{D}) = (\mathcal{C}', \mathcal{D}')$ is defined as

$$\begin{cases} C'^u_{o/r}(n) & = C^u_{i/r}(n-d) \\ C'^l_{o/r}(n) & = C^l_{i/r}(n-d) \end{cases}$$

and $(\mathcal{C}', \mathcal{D}')_{k/j} = (\mathcal{C}, \mathcal{D})_{k/j}$ for all clocks $k \neq o$ or $j \neq r$. This guarantees that $(\mathcal{C}', \mathcal{D}')$ are the greatest bounds for which the inequalities of Lemma 9 hold and consequently, that all clock vectors that satisfy the delay predicate and bounds $(\mathcal{C}, \mathcal{D})$ satisfy the bounds $(\mathcal{C}', \mathcal{D}')$.

The BD, PJD and TDMA components are constant mappings. For example a $\text{BD}(r, d, B, k)$ component corresponds to a mapping $bd\langle r, d, B, k\rangle(\mathcal{C}, \mathcal{D}) = (\mathcal{C}', \mathcal{D}')$ where

$$\begin{cases} C'^u_{r/k}(n) & = \lceil Bn \rceil \\ C'^l_{r/k}(n) & = \lceil Bn \rceil + d \end{cases}$$

and $(\mathcal{C}', \mathcal{D}')_{i/j} = (\mathcal{C}, \mathcal{D})_{i/j}$ for all clocks $i \neq r$ or $j \neq r$.

The mappings are all order- and limit-preserving for they are compositions of monotonic operators. Proofs of monotonicity and continuity of the involved operators have been omitted but are similar to the proofs in [12]. The consequence is (see [6]) that the combined mapping $\Pi(\mathcal{C}, \mathcal{D}) = (\mathcal{C}, \mathcal{D}) \sqcup M_1(\mathcal{C}, \mathcal{D}) \sqcup M_2(\mathcal{C}, \mathcal{D}) \sqcup \ldots$ has a unique greatest fix-point where all the inequalities hold. The algorithm used for experimentation (see the next section), iteratively applies Π starting with the top drift bounds.

VII. Experimental Results

We have implemented the mappings and resulting fix-point algorithm in a prototype programmed in Python. At the core, is a suite of operators on functions in $\mathcal{F}$. More precisely, operators on eventually periodic functions that are represented by a finite transient part (the aperiodic introduction) and a periodic part with a finite period. The program consists of a few hundred lines of code, of which more than half is concerned with the operators. The implementation of the operators is based on [2] and [18].

In order to compare to conventional network calculus' analysis both systems were modelled with an additional clock that represented real-time. So all bounds on input streams and availability of resources were modelled with respect to the added real-time clock k.

The multimedia decoder of Figure 1 is described and analysed by the following code. The system consists of a list of mappings over bound matrices that correspond to the predicates introduced for the components in Section IV.

```
[k,i,x,y,o,r,s] = range(7) # identifiers

# declare system components
system = [
    pjd(i, 4, 24, 1, k),
    bd(r, 0.3, 3, k),
    bd(s, 0.3, 3, k),
    gpc(i, x, r),
    gpc(i, y, s),
    and_join(x, y, o),
]

bounds = solve(system) # fixpoint calculation

print backlog_bound(bounds, x, o)
```

Not visible in the above snippet, are the mappings described by the theorems in Section V, that handle the interaction between the bounds; the interaction between lower on upper bounds, transitivity, and the interaction of drift bounds and clock bound are applied by the solver implicitly because they are independent of the system's components.

The multimedia decoder is modelled with an input stream generated by a PJD component with period of 4, jitter of 24 and a minimum inter-arrival distance of 1. The two decoders had resources, generated by the BD components, with bandwidth of 0.3 units per time-unit and a drift of 3.

We calculate the buffer size using conventional network calculus using the curves for input stream i and resources r, s defined by the PJD and BD components with respect to the real-time clock k, e.g., for stream i the upper arrival input curve is $\alpha^u_i = D^u_{i/k}$. Then we derive a bound $\max\{\alpha^u_x(n) - \alpha^l_y(n) \mid n \in \mathbb{N}\}$ for the buffer size of stream x, where $\alpha^u_x = (\alpha^u_i \otimes \beta^u_r) \oslash \beta^l_r \wedge \beta^u_r$ and $\alpha^l_y = (\alpha^l_i \otimes \beta^l_s)$.

In this particular configuration our analysis calculated, in twelve iterations, an upper bound of 8 for the backlog of

streams x and y, whereas the conventional method yields a bound of 18. We also confirmed that increasing the jitter of the input stream i has little effect on the estimated buffer size, while the conventional method's estimate grows quickly with jitter.

Experiments on the second example, shown below, where a server responds to requests over a bus, showed that the analysis effectively handles the regular nature of requests received over the TDMA bus.

```
[k,i,x,y,o,r,t,s] = range(8) # identifiers

# declare system components
system = [
    pjd(i, 9, 23, 1, r),
    bd(r, 0.33, 13, k),
    tdma(r, t, u, 9),
    gpc(i, x, t),
    delay(9, x, y, r),
    gpc(y, o, u),
]

bounds = solve(system) # fixpoint calculation

print backlog_bound(bounds, y, o)
```

Conventional analysis has no straightforward way to model this second example. The closest analysis we could make, again taking the drift bounds for input stream i and resource r, uses $\beta_t^u = \beta_s^u = \beta_r^u \otimes tdma^u \otimes id$ where $tdma^u(n) = 9\lceil n/16 \rceil$ and $\beta_t^l = \beta_s^l = \beta_r^l \otimes tdma^l \otimes id$ where $tdma^l(n) = 9\lfloor (n+7)/16 \rfloor$ for the upper (resp. lower) bounds for the shared bus resources. The delay is modeled according to [12] such that $\alpha_y^u = \alpha_x^u \otimes delay$ where $delay(n) = 0$ if $n \leq 9$ else ∞ and $\alpha_x^u = (\alpha_i^u \otimes \beta_t^u) \oslash \beta_t^l \wedge \beta_t^u$. Then we obtain a buffer size bound $\max\{\alpha_y^u(n) - \beta_s^l(n) \mid n \in \mathbb{N}\}$.

With a TDMA period of 9 and matching delay our analysis consistently showed, after 14 iterations, a backlog bound of 2 between streams y and o, whereas the backlog bound calculated by the conventional method is 7. Here the improved precision of our approach is due to the combined use of clock and drift bounds.

Clearly conventional analysis has a much lower complexity because we model n streams with n^2 bounds, whereas conventional analysis needs only n bounds. There is room for improvement because the effect of most bounds is local, there is no need to track all n^2 relations; the implementation could exploit system topology to reduce the number of bounds involved in the calculation. It should also be noted that conventional analysis does not need a fix-point computation if there are no cyclic dependencies, as is the case for the examples.

Experience suggests that the computational cost is most affected by the choice of curves. Combining curves with many different prime factors can lead to very long periods, which affects all further operations on the curve.

VIII. Discussion

We have presented a calculus for stream processing systems that preserves correlations between streams. The joint use of clock bounds and drift bounds allows for an expressive modeling of timing constraints, such as bursty behavior and resource scheduling. We have implemented a prototype tool and shown that for some frequently encountered system topologies the proposed calculus yields significantly better estimates than conventional network calculus.

Future work will employ the presented calculus to distribute synchronous (deterministic) data-flow systems over loosely time-triggered architectures [1], so as to guarantee quality of service. We intend to use drift bounds to capture the bounded nondeterminism of communication between independently clocked processors.

References

[1] Albert Benveniste, Paul Caspi, Marco Di Natale, C. Pinello, Alberto L. Sangiovanni-Vincentelli, and Stavros Tripakis. Loosely time-triggered architectures based on communication-by-sampling. In *ICES*, pages 231–239. ACM, 2007.

[2] Anne Bouillard and Éric Thierry. An Algorithmic Toolbox for Network Calculus. *DEDS*, 18(1):3–49, October 2007.

[3] Samarjit Chakraborty, S Kunzli, and Lothar Thiele. A general framework for analysing system properties in platform-based embedded system designs. *DATE*, pages 190–195, 2003.

[4] A. Cohen, L. Mandel, F. Plateau, and M. Pouzet. Abstraction of clocks in synchronous data-flow systems. *PLS*, pages 237–254, 2008.

[5] Albert Cohen, M. Duranton, C. Eisenbeis, C. Pagetti, F. Plateau, and M. Pouzet. N-synchronous Kahn networks: a relaxed model of synchrony for real-time systems. In *ACM SIGPLAN Notices*, volume 41, pages 180–193. ACM, 2006.

[6] B. A. Davey and H. A. Priestley. *Introduction to Lattices and Order*. Cambrige University Press, 2002.

[7] W. Haid and Lothar Thiele. Complex task activation schemes in system level performance analysis. In *CODES*, pages 173–178. ACM, 2007.

[8] M.G. Harbour, M.H. Klein, R. Obenza, B. Pollak, and T. Ralya. *A Practitioner's Handbook for Real-Time Analysis*. Kluwer, 1993.

[9] Martijn Hendriks and Marcel Verhoef. Timed automata based analysis of embedded system architectures. In *IPDPS*. IEEE, 2006.

[10] R Henia, A Hamann, M Jersak, R Racu, K Richter, and R Ernst. System level performance analysis the SymTA/S approach. *CDT*, 152(2):148–166, 2005.

[11] Kai Huang and Lothar Thiele. Performance analysis of multimedia applications using correlated streams. In *DATE*, pages 912–917. EDA Consortium, 2007.

[12] Jean-Yves Le Boudec and Patrick Thiran. *Network calculus: a theory of deterministic queuing systems for the internet*. Springer-Verlag, 2001.

[13] Simon Perathoner, Tobias Rein, Lothar Thiele, Kai Lampka, and Jonas Rox. Modeling structured event streams in system level performance analysis. *LCTES*, 45(4):37, April 2010.

[14] C. A. Petri. *Interpretations of net theory*. Gesellschaft für Mathematik und Datenverarbeitung, mbH Bonn, 1976.

[15] Jonas Rox and Rolf Ernst. Modeling Event Stream Hierarchies with Hierarchical Event Models. In *DATE*, pages 492–497. Ieee, March 2008.

[16] Irina M. Smarandache, Thierry Gautier, and Paul Le Guernic. Validation of mixed SIGNAL-ALPHA real-time systems through affine calculus on clock synchronisation constraints. In *FM*, pages 1364–1383, 1999.

[17] Lothar Thiele, S. Chakraborty, and M. Naedele. Real-time calculus for scheduling hard real-time systems. In *ISCAS*, volume 4, pages 101–104. IEEE, 2000.

[18] Ernesto Wandeler. *Modular performance analysis and interface-based design for embedded real-time systems*. PhD thesis, ETH Zürich, 2006.

[19] Ernesto Wandeler and Lothar Thiele. Characterizing workload correlations in multi processor hard real-time systems. In *RTAS*, 2005.