Revisiting GPC and AND Connector in Real-Time Calculus

Yue Tang[1], Nan Guan[1], Weichen Liu[2], Linh Thi Xuan Phan[3], Wang Yi[4]

[1]The Hong Kong Polytechnic University, Hong Kong
[2]Chongqing University, China
[3]University of Pennsylvania, US
[4]Uppsala University, Sweden

***Abstract*—Real-Time Calculus (RTC) is a powerful framework for modeling and worst-case performance analysis of networked systems. GPC and AND are two fundamental components in RTC, which model priority-based resource arbitration and synchronization operations, respectively. In this paper, we revisit GPC and AND. For GPC, we develop tighter output arrival curves to more precisely characterize the output event streams. For AND, we first identify a problem in the existing analysis method that may lead to negative values in the output curves, and present corrections to the problem. Then we generalize AND to synchronize more than two input event streams. We implement our new theoretical results and conduct experiments to evaluate their performance. Experiment results show significant improvement of our new methods in analysis precision and efficiency.**

I. Introduction

Real-Time Calculus (RTC) [1], [2] is a theoretical framework for performance analysis of networked embedded systems, which is rooted in the Network Calculus theory [3]. RTC uses variability characterization curves (called *curves* for short) [4] to model workload and resource, and analyzes workload flows through a network of processing and communication resources to bound the worst-case performance of the system. Comparing to the traditional real-time scheduling theory, RTC uses much more general workload and resource models, and thus can model a much wider range of realistic systems. At the same time, its closed-form analytical bounds also provide much higher analysis efficiency compared to state-based modeling and analysis techniques such as model checking [5], [6]. Due to these advantages, RTC has drawn considerable attention from both academia and industry in recent years, and has been successfully applied to solve many realistic problems.

RTC uses different abstract components to model different resource arbitration schemes or operational semantics. Two fundamental abstract components in RTC are GPC [2] and AND connector [4]. GPC (Greedy Processing Component) essentially models priority-based resource arbitration among multiple workload streams sharing the same hardware platform. GPC is the most widely used building block in RTC due to the universal use of fixed-priority scheduling in practice. AND connector is another important component in RTC, which models synchronization of events from two streams. AND is widely used in many application domains such as sensor networks and IoT, where the states of different parts of the system with the same time stamp should be fused to derive useful information about the system.

This paper makes a revisit to GPC and AND in RTC. We present new results to fix problems in existing RTC theory, improve the analysis precision as well as make it more general. Due to the wide usage of GPC and AND, our new results are significant to the whole RTC framework. More specifically, we make the following contributions in the paper.

For GPC, we derive tighter output arrival curves to more precisely bound the timing behavior of output event streams. The original output arrival curves of GPC were developed about 15 years ago [1], [2], as a major foundational result of the RTC framework. Since then, no improvement has ever been made, although it is widely known that these bounds are not tight. This paper for the first time makes these bounds tighter. The key idea is to utilize the remaining service curve to refine the information about how much resource can be actually consumed, and exclude the unused resource when computing the output arrival curves.

For AND connector, we make contributions in two aspects:

- We identify and fix a problem in the existing analysis method for AND in [4] that may lead to negative values in the lower output curves. We present new lower output curves to fix the problem.
- We generalize AND to support synchronization of more than two input event streams. The original AND only has two input ports. A straightforward way for the generalization is to model a multi-input AND as several dual-input AND connectors cascaded together. However, this straightforward generalization is both imprecise and inefficient. We present a more elegant way to generalize AND to multiple inputs, which outperforms the straightforward generalization approach in terms of both precision and efficiency.

Finally, we conduct experiments to evaluate our new results in different aspects, with randomly generated system models under different configurations. Experiment results show significant improvement of our new methods in analysis precision and efficiency.

II. Related Work

There has been previous efforts to improve the analysis accuracy of the RTC theory. The first direction models the system using a state-based model, such as timed automata [7] or event count automata [5], which can then be analyzed using standard verification techniques. These approaches can provide tight bounds; however, as is with verification, they suffer from the state explosion problem. To solve this efficiency issue, prior work has developed interfacing techniques [8], [9] that combine RTC with state-based approaches. Our improved RTC analysis can be transparently integrated into these interfacing techniques to analyze the RTC components, thus improving their overall analysis results.

Beyond state-based approaches, the trajectorial approach has been developed [10] to bound the end-to-end analysis of distributed systems. This approach gives better accuracy than RTC does, but its fixed point computation is also much more expensive than RTC.

Within the RTC context, Bouillard [11] has developed a tighter analysis for blind multiplexing and FIFO networks using linear programming. Our work focuses on improving the output bound of a single component (GPC and multi-input AND), and thus it is orthogonal to the work in [11].

In [12], the analysis of AND was studied with the so-called standard event models, which are special cases of the curves in RTC. The analysis techniques in [12] cannot handle AND with general curves as inputs. A hybrid framework was proposed in [13] for analyzing real- time embedded systems combining RTC and Timed Automata. In principle, the timed automata component can be used to model synchronization operations. However, this approach is still limited by the low efficiency for model-checking the timed automata, although the state space has already been significantly reduced comparing with modeling the whole system with a timed automata network.

III. RTC Basics

RTC models event stream flows through a network of processing and communication resources by components connected with their input and output ports. The workload and resource are modeled by variability characterization curves (*curves* for short) [4]. Each type of component models a particular resource arbitrary policy or operational semantic, supported by closed-form formula to transform the input curves into the output curves, and the worst-case delay and backlog bounds.

A. Arrival and Service Curves

RTC uses *variability characterization curves* (*curves* for short) to describe timing properties of event streams and available resource:

Definition 1 (Arrival Curve): Let $R[s,\, t)$ denote the total amount of requested capacity to process in time interval $[s,\, t)$. Then, the corresponding upper and lower arrival curves are denoted as α^u and α^l, respectively, and satisfy:

$$\forall s < t, \quad \alpha^l(t-s) \leq R[s,\, t) \leq \alpha^u(t-s) \tag{1}$$

where $\alpha^u(0) = \alpha^l(0) = 0$.

Definition 2 (Service Curve): Let $C[s,\, t)$ denote the amount of events that the resource can process in time interval $[s,\, t)$. Then, the corresponding upper and lower service curves are denoted as β^u and β^l, respectively, and satisfy:

$$\forall s < t, \quad \beta^l(t-s) \leq C[s,\, t) \leq \beta^u(t-s) \tag{2}$$

where $\beta^u(0) = \beta^l(0) = 0$.

For simplicity of presentation, we use a curve pair $f = (f^u, f^l)$ to represent both the upper and lower curves.

B. GPC: Greedy Processing Component

A GPC processes events from input (described by arrival curves α^u and α^l) in a greedy fashion, as long as it complies with the availability of resources (described by service curves β^u and β^l). Input events are backlogged in a FIFO buffer if currently no resource is available. When the available resource arrives, the first event in the buffer is processed and the corresponding output event is generated (described by output arrival curves α'^u and α'^l). If the buffer is empty, the resource will be passed to the subsequent component (described by remaining service curves β'^u and β'^l):

$$\alpha^{u'} = \min((\alpha^u \otimes \beta^u) \oslash \beta^l, \beta^u) \tag{3}$$

$$\alpha^{l'} = \min((\alpha^l \oslash \beta^u) \otimes \beta^l, \beta^l) \tag{4}$$

$$\beta^{u'} = (\beta^u - \alpha^l) \overline{\oslash} 0 \tag{5}$$

$$\beta^{l'} = (\beta^l - \alpha^u) \overline{\otimes} 0 \tag{6}$$

where

$$(f \otimes g)(\Delta) \triangleq \inf_{0 \leq \lambda \leq \Delta} \{f(\Delta - \lambda) + g(\lambda)\}$$

$$(f \overline{\otimes} g)(\Delta) \triangleq \sup_{0 \leq \lambda \leq \Delta} \{f(\Delta - \lambda) + g(\lambda)\}$$

$$(f \oslash g)(\Delta) \triangleq \sup_{\lambda \geq 0} \{f(\Delta + \lambda) - g(\lambda)\}$$

$$(f \overline{\oslash} g)(\Delta) \triangleq \inf_{\lambda \geq 0} \{f(\Delta + \lambda) - g(\lambda)\}$$

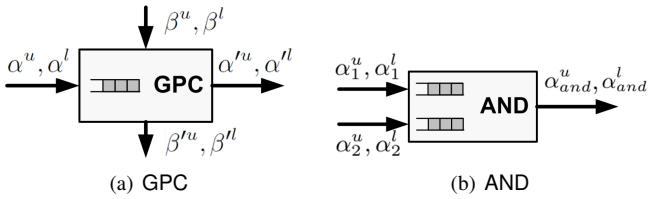
The amount of events in the input buffer, i.e., the backlog, can be bounded by $\max(0, V(\alpha^u, \beta^l))$, where $V(f, g)$ gives the maximal vertical distance from curve f to curve g:

$$V(f, g) \triangleq \max(0, \sup_{\lambda \geq 0} \{f(\lambda) - g(\lambda)\})$$

The delay of events can be bounded from above by $H(\alpha^u, \beta^l)$, where $H(f, g)$ gives the maximal horizontal distance from curve f to curve g

$$H(f, g) \triangleq \sup_{\lambda \geq 0} \{\inf\{\varepsilon \geq 0 : f(\lambda) \leq g(\lambda + \varepsilon)\}\}$$

Multiple GPCs are connected into a network. The output arrival and resource curves of one GPC are used as the input for the analysis of the downstream nodes along the event and resource flow.



(a) GPC (b) AND

Fig. 1. Schematic of GPC and AND.

C. AND *Connector*

Another useful abstract component is the AND connector [4], which combines two input event streams into a single combined stream. Data arriving on one input stream must be buffered until partner events arrive on the other input stream. Partnering events join together and immediately pass the AND connector, and consequently either of the internal buffers is empty at any point of time. The two input event streams are characterized by arrival curves α_1^u, α_1^l and α_2^u, α_2^l. In [4], the following output curves $\alpha_{1,2}^u$ and $\alpha_{1,2}^l$ are used to upper and lower bound the combined event streams at the output:

$$\begin{aligned}\alpha_{1,2}^u = &\max(\min(\alpha_1^u \oslash \alpha_2^l + B_1 - B_2, \alpha_2^u)),\\ &\min(\alpha_2^u \oslash \alpha_1^l + B_2 - B_1, \alpha_1^u))\end{aligned} \tag{7}$$

$$\begin{aligned}\alpha_{1,2}^l = &\max(\min(\alpha_1^l \overline{\oslash} \alpha_2^u + B_1 - B_2, \alpha_2^l),\\ &\min(\alpha_2^l \overline{\oslash} \alpha_1^u + B_2 - B_1, \alpha_1^l))\end{aligned} \tag{8}$$

where B_1 and B_2 denote the initial buffer levels of the two input streams. Later in Section V, we will show that there is a problem with the *lower* output bound in above, and also present a correction for it.

The delay of events at two inputs is bounded by

$$\begin{aligned}d_1 &\le H(\alpha_1^u + B_1, \alpha_2^l + B_2)\\ d_2 &\le H(\alpha_2^u + B_2, \alpha_1^l + B_1)\end{aligned}$$

and the backlog at the two input buffers are bounded by

$$\begin{aligned}b_1 &\le \max(0, V(\alpha_1^u + B_1, \alpha_2^l + B_2))\\ b_2 &\le \max(0, V(\alpha_2^u + B_2, \alpha_1^l + B_1))\end{aligned}$$

IV. Improving GPC

Our tighter output arrival curves are obtained by modeling the operational semantics of GPC in a more subtle way. In GPC, when the buffer is empty, the available resource will not be consumed. Instead, it will be directly sent to the output resource interface. In other words, only a portion of the input resource is actually consumed to process the events. If we classify the input resources into two types, (i) the input resources that are actually consumed to process events and (ii) the input resources that are not consumed, then eliminating all the type (i) resources from the input resource stream will not affect the timing behavior of output event stream.

Based on the above observation, we develop a new way to model the semantics of GPC, as shown in Figure 2. Inside the GPC, a **controller** classifies the input resources into the above mentioned two types by checking whether the input

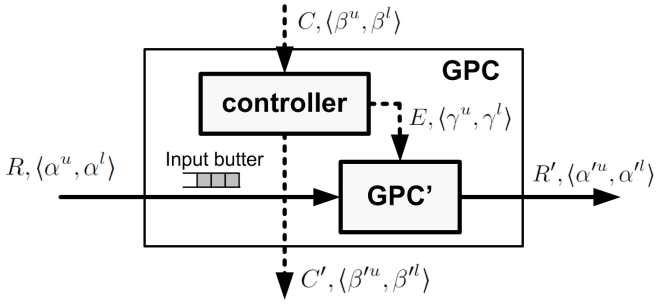


Fig. 2. A new model of the GPC semantics.

buffer is empty. If yes, the controller will send the resource to output resource interface. Otherwise, the resource will be sent to GPC'and perform the actual event processing functionality of GPC, which is modeled by *effective service curve*:

Definition 3 (Effective Service Curve): Let $E[s,t)$ denote the amount of resource actually sent to GPC' to process the events in time interval $[s,t)$, then the corresponding upper and lower effective service curves are denoted as γ^u and γ^l, respectively, and satisfy:

$$\forall s < t, \quad \gamma^l(t-s) \le E[s,\, t) \le \gamma^u(t-s) \tag{9}$$

where $\gamma^u(0) = \gamma^l(0) = 0$.

It is easy to know the following lemma:

Lemma 1: Let $C[s,t)$ denote the amount of available resource in time interval $[s,t)$, $C'[s,t)$ denote the amount of resource not used to process events in $[s,t)$. Let $E[s,t)$ denote the amount of resource actually sent to GPC' to process the events in $[s,t)$. For any time interval $[s,t)$ it holds:

$$C[s,t) = E[s,t) + C'[s,t)$$

Proof: Any available resource available in $[s,t)$ is either used to process the events (included in $E[s,t)$) or not (included in $C'[s,t)$). ■

By transferring this relation to the time interval domain, we can compute γ^u by β and β' as follows:

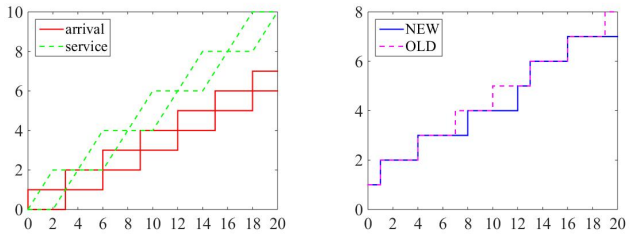
Lemma 2: For any time interval $[s,t)$ with $t - s = \Delta$, $E[s,t)$ is upper and lower bounded by

$$\gamma^u(\Delta) = \inf_{\lambda \ge \Delta} \{\beta^u(\lambda) - \beta'^l(\lambda)\} \tag{10}$$

$$\gamma^l(\Delta) = \sup_{0 \le \lambda \le \Delta} \{\beta^l(\lambda) - \beta'^u(\lambda)\} \tag{11}$$

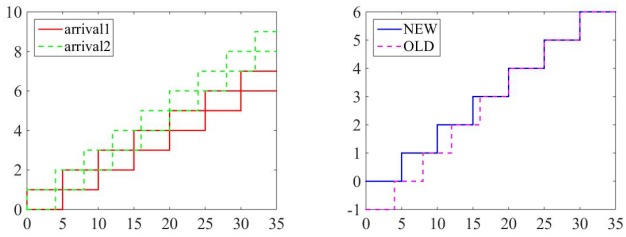
Proof: We first prove (10). For any time interval $[s - \varepsilon_1, t + \varepsilon_2)$ with $\varepsilon_1 \ge 0$ and $\varepsilon_2 \ge 0$, it holds $E[s,t) \le E[s - \varepsilon_1, t + \varepsilon_2)$, so we have

$$\begin{aligned}E[s,t) &= \inf_{\varepsilon_1 \ge 0 \wedge \varepsilon_2 \ge 0} \{E[s - \varepsilon_1, t + \varepsilon_2)\}\\ &= \inf_{\varepsilon_1 \ge 0 \wedge \varepsilon_2 \ge 0} \{C[s - \varepsilon_1, t + \varepsilon_2) - C'[s - \varepsilon_1, t + \varepsilon_2)\}\\ &\le \inf_{\varepsilon_1 \ge 0 \wedge \varepsilon_2 \ge 0} \{\beta^u(\Delta + \varepsilon_1 + \varepsilon_2) - \beta'^l(\Delta + \varepsilon_1 + \varepsilon_2)\}\\ &= \inf_{\varepsilon \ge 0} \{\beta^u(\Delta + \varepsilon) - \beta'^l(\Delta + \varepsilon)\}\\ &= \inf_{\lambda \ge \Delta} \{\beta^u(\lambda) - \beta'^l(\lambda)\}\end{aligned}$$



(a) Input arrival curves. (b) Upper output arrival curves.

Fig. 3. An example comparing the original and new upper output arrival curves for GPC.



(a) Input arrival curves. (b) Lower output arrival curves.

Fig. 4. An example illustrating the negative value problem in original output curves of AND.

Then we prove (11). For any time interval $[s+\varepsilon_1, t-\varepsilon_2)$ with $\varepsilon_1 \geq 0$, $\varepsilon_2 \geq 0$ and $\varepsilon_1 + \varepsilon_2 \leq t-s$, it holds $E[s,t) \geq E[s+\varepsilon_1, t-\varepsilon_2)$, so we have

$$\begin{aligned}
&E[s,t)\\
&= \sup_{\varepsilon_1\geq 0\wedge\varepsilon_2\geq 0\wedge\varepsilon_1+\varepsilon_2\leq t-s} \{E[s+\varepsilon_1, t-\varepsilon_2)\}\\
&= \sup_{\varepsilon_1\geq 0\wedge\varepsilon_2\geq 0\wedge\varepsilon_1+\varepsilon_2\leq t-s} \{C[s+\varepsilon_1, t-\varepsilon_2) - C'[s+\varepsilon_1, t-\varepsilon_2)\}\\
&\geq \sup_{\varepsilon_1\geq 0\wedge\varepsilon_2\geq 0\wedge\varepsilon_1+\varepsilon_2\leq \Delta} \{\beta^l(\Delta-\varepsilon_1-\varepsilon_2) - \beta'^u(\Delta-\varepsilon_1-\varepsilon_2)\}\\
&= \sup_{0\leq\varepsilon\leq\Delta} \{\beta^l(\Delta-\varepsilon) - \beta'^u(\Delta-\varepsilon)\}\\
&= \sup_{0\leq\lambda\leq\Delta} \{\beta^l(\lambda) - \beta'^u(\lambda)\}
\end{aligned}$$

■

With the γ^u and γ^l obtained from the above lemma, now we can compute the output arrival curves of GPC'. Note that the operational semantics of GPC' is exactly the same as the original greedy processing component GPC, so we have the following theorem:

Lemma 3: Given a GPC with input arrival curves α^u, α^l and service curves β^u, β^l, and γ^u, γ^l computed by Lemma 2, the output events can be upper bounded by:

$$\alpha^{u'} = [(\alpha^u \otimes \gamma^u) \oslash \gamma^l] \wedge \gamma^u \tag{12}$$

The new upper output arrival curve in the above lemma is generally incomparable with the original one: sometimes our new curve is tighter, sometimes the original one is tighter, and sometimes the new curve and the original curve do not dominate each other (one curve is tighter for some parts and the other curve is tighter for some other parts). Therefore, combining our new curve with the original one gives the best result as stated in the following theorem:

Theorem 1: Given a GPC with input arrival curves α^u, α^l and service curves β^u, β^l, and γ^u, γ^l computed by Lemma 2, the output events can be upper bounded by:

$$\alpha^{u'} = [(\alpha^u \otimes \gamma^u) \oslash \gamma^l] \wedge [(\alpha^u \otimes \beta^u) \oslash \beta^l] \wedge \gamma^u \tag{13}$$

Similarly, by using γ^u and γ^l to replace β^u and β^l, we can also get a new *lower* output arrival curve. However, this yields a looser lower output arrival curve than the original one. Therefore, for computing lower output arrival curve, we should still use (4).

An Example. We use a simple example to demonstrate our new output arrival curve bounds. Suppose we have a strictly periodic event arriving pattern with a period of 3 time units (the processing of each event takes one time unit) and a TDMA resource provides two time units of resource with a period of 4 time units, the corresponding arrival and service curves of which are shown in Figure 3-(a). The original and our new output upper arrival curves, denoted by OLD and NEW respectively, are shown in Figure 3-(b), where we can see our new curve is tighter than the original one.

V. Revising AND

We first identify the problem in existing analysis methods of AND, and then present solutions to fix the problem.

A. Problem

Recall the original lower output curve in [4]:

$$\begin{aligned}
\alpha^l_{1,2} = &\max(\min(\alpha^l_1 \overline{\oslash} \alpha^u_2 + B_1 - B_2, \alpha^l_2),\\
&\min(\alpha^l_2 \overline{\oslash} \alpha^u_1 + B_2 - B_1, \alpha^l_1))
\end{aligned}$$

We use the following example to illustrate its problem. Suppose we have two strictly periodic event streams with periods $P_1 = 5$ and $P_2 = 4$, and initial buffers $B_1 = B_2 = 0$. The input curves are shown in Figure 4-(a). Then their upper and lower arriving curves are

$$\alpha^u_1(\Delta) = \lceil \Delta/5 \rceil, \quad \alpha^l_1(\Delta) = \lfloor \Delta/5 \rfloor$$
$$\alpha^u_2(\Delta) = \lceil \Delta/4 \rceil, \quad \alpha^l_2(\Delta) = \lfloor \Delta/4 \rfloor$$

Let $\Delta = 1$, then following the definition of $\overline{\oslash}$ we have

$$\alpha^l_{1,2}(1) = \max\left(\min\left(X_1, 0\right), \min\left(X_2, 0\right)\right)$$

where

$$\begin{aligned}
X_1 &= \inf_{\lambda\geq 0}\{\lfloor (1+\lambda)/5 \rfloor - \lceil \lambda/4 \rceil\}\\
&\leq \lfloor (1+0.1)/5 \rfloor - \lceil 0.1/4 \rceil = -1\\
X_2 &= \inf_{\lambda\geq 0}\{\lfloor (1+\lambda)/4 \rfloor - \lceil \lambda/5 \rceil\}\\
&\leq \lfloor (1+0.1)/4 \rfloor - \lceil 0.1/5 \rceil = -1
\end{aligned}$$

So both X_1 and X_2 are negative, and consequently $\alpha^l_{1,2}(1)$ is also negative (the resulting output curve is shown as the dash line in Figure 4). This violates the basic assumption for all the computation rules in RTC that all curves are non-negative.

By having a closer look into the above example, we will see that these negative values are actually a *precision* problem rather than a *correctness* problem. α^l is a *lower* bound for the number output events, and a negative number is indeed a correct lower bound. Therefore, for a single AND connector, the original lower output curve is still correct, but just too imprecise (even more imprecise than the naive lower bound 0 in some cases). However, when the AND connectors are put into a RTC network, the effect of these negative curves will propagate to other components, and eventually may cause inconsistency to the operational semantics of the system model, as all the computation rules in RTC are based on the implicit assumptions that all curves are positive.

B. Solution

The problem mentioned above can be easily fixed by changing all the negative values to 0. However, this quick fix only superficially solves the negative value problem, but does not really address the real source of pessimism behind the problematic original lower output curve.

In the following, we present a new lower output curve for AND. Our new result is tighter than the original one and systematically solves the negative value problem.

Let $R_i[s,t)$ denote the total number of events arrived in time interval $[s,t)$, and use $R_i(t)$ to represent $R_i[0,t)$ for short. Moreover, we use $[x]^0$ to denote $\max(x,0)$ for simplicity.

We first quote a known result from [4]:

Lemma 4: $R_i(t) - R_j(s)$ is upper and lower bounded by:

$$\alpha_i^l \overline{\oslash} \alpha_j^u(t-s) \le R_i(t) - R_j(s) \le \alpha_i^u \oslash \alpha_j^l(t-s)$$

Theorem 2: The output event stream of an AND connector with two input event streams characterized by arrival curves α_1 and α_2 is lower bounded by the curve:

$$\begin{aligned}\alpha_{1,2}^l = &\min(\max(\alpha_1^l \overline{\oslash} \alpha_2^u + B_1 - B_2, \alpha_1^l),\\ &\max(\alpha_2^l \overline{\oslash} \alpha_1^u + B_2 - B_1, \alpha_2^l))\end{aligned} \tag{14}$$

Proof: The backlogs of the two streams at time t are

$$\begin{aligned}b_1(t) &= [R_1(t) + B_1 - (R_2(t) + B_2)]^0\\ b_2(t) &= [R_2(t) + B_2 - (R_1(t) + B_1)]^0\end{aligned}$$

Let $R_{1,2}[s,t)$ denote the number of output events in time interval $[s,t)$, which is the minimum between the events of the two streams in this interval:

$$\begin{aligned}R_{1,2}[s,t) &= \min(R_1[s,t) + b_1(s), R_2[s,t) + b_2(s))\\ &= \min(R_1[s,t) + [R_1(s) - R_2(s) + B_1 - B_2]^0,\\ &\qquad R_2[s,t) + [R_2(s) - R_1(s) + B_2 - B_1]^0)\\ &= \min(\max\{R_1[s,t), R_1(t) - R_2(s) + B_1 - B_2\},\\ &\qquad \{R_2[s,t), R_2(t) - R_1(s) + B_2 - B_1\})\end{aligned}$$

and by applying Lemma 4 we finally get

$$\begin{aligned}R_{1,2}[s,t) \ge &\min(\max(\alpha_1^l \overline{\oslash} \alpha_2^u + B_1 - B_2, \alpha_1^l),\\ &\max(\alpha_2^l \overline{\oslash} \alpha_1^u + B_2 - B_1, \alpha_2^l))\end{aligned}$$

by which the theorem is proved. ■

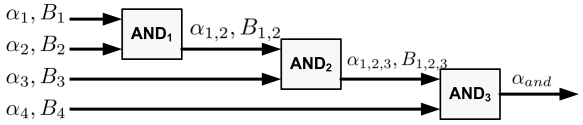


Fig. 5. Modeling a multi-input AND connector as cascaded dual-input AND connectors.

The solid line in Figure 3-(b) is the lower output curve generated by out new method. We can see that it does not only solve the negative value problem, but also in general more precise than the original curve even if the negative values are changed to zero.

VI. Generalizing AND

Many realistic systems need to synchronize events from *more than two* streams. In the following we generalize the original dual-input AND connector to the multi-input setting.

A. The Cascaded Approach

A straightforward approach to analyze AND connectors with multiple inputs is to model a multi-input AND connector as several cascaded dual-input AND connectors. Figure 5 shows the cascaded modeling of a multi-input AND connector with four input streams.

We use

$$(\alpha_{i,j}, B_{i,j}) = (\alpha_i, B_i) \circledast (\alpha_j, B_j)$$

to represent the computation of output curves for an AND connector with input curves α_i and α_j (using (7) to compute the upper curves and using our new method (14) to compute the lower curves) as well as the initial buffer level $B_{i,j}$ for this event stream that is useful when it is further combined with other streams:

$$B_{i,j} = \min(B_i, B_j)$$

In general, for n-input streams cascaded by dual-input AND connectors in a particular order $\pi = \{\alpha_1, \alpha_2, \cdots, \alpha_n\}$ the final output curve is computed by

$$(\alpha_{and,\pi}, B_{and}) = (\alpha_1, B_1) \circledast \cdots \circledast (\alpha_n, B_n) \tag{15}$$

The result of this approach is sensitive to the order to cascade the inputs, and it is generally unknown which order gives the best result. On the other hand, the output bounds obtained with any of the pairing orders are valid. Therefore, we can join the results with all the possible orders to get tighter bounds:

Theorem 3: For multi-input AND connector, let Π be the full permutation of $\{\alpha_1, \cdots, \alpha_n\}$, i.e., the set of all possible cascading orders of the input streams, and $\alpha_{and,\pi}^u$ and $\alpha_{and,\pi}^l$ are the upper and lower output curves for a particular cascading order $\pi \in \Pi$, then output curves of the multi-input AND connector is upper and lower bounded by:

$$\alpha_{and}^u = \min_{\pi \in \Pi}\{\alpha_{and,\pi}^u\}, \qquad \alpha_{and}^l = \max_{\pi \in \Pi}\{\alpha_{and,\pi}^l\}$$

The proof the theorem is straightforward and thus omitted.

The computation of the maximum delay and backlog of each stream also depends on the cascading order. For example, in Figure 5, the maximal delay (backlog) of events in stream α_1 should be counted as the sum of the delay (backlog) incurred at AND_1, AND_2 and AND_3, while for an event in stream α_4 only the delay (backlog) at AND_3 is counted. Obviously, to compute a tight delay (backlog) bound for a stream α_i, we should use a cascading order in which α_i only connects to the last dual-input AND connector:

Theorem 4: A multi-input AND connector has n inputs characterized by $\{\alpha_1, \alpha_2, \cdots, \alpha_n\}$. The maximal delay and backlog of events in a stream α_i is upper bounded by

$$d_i \leq H\left(\alpha_i^u + B_i, \alpha^{l*} + \min_{i\neq j}\{B_j\}\right)$$
$$b_i \leq \max\left(0, V\left(\alpha_i^u + B_i, \alpha^{l*} + \min_{i\neq j}\{B_j\}\right)\right)$$

where α^{l*} is the output curve for joining the other $n-1$ input streams using Theorem 3.

The proof the theorem is straightforward and thus omitted.

B. The Holistic Approach

In the following we present a new approach to compute the output curve and delay/backlog bounds for multi-input AND which is both more precise and more efficient than the above straightforward approach. We call this new approach the *holistic* approach, as it computes the desired results with all the inputs curves at the same time (rather than computing them step by step with two input curves at each step in the cascaded approach).

Theorem 5: Given an AND connector with n inputs, which are characterized by the input upper and lower curves (α_1^u, α_1^l), (α_2^u, α_2^l), $\cdots$, (α_n^u, α_n^l), and the initial buffer levels $B_1, \cdots, B_n$, the output arrival curves are computed by:

$$\alpha_{and}^u = \max_{\text{all } k}\left\{\min\left(\min_{i\neq k}\left\{\alpha_i^u \oslash \alpha_k^l + B_i - B_k\right\}, \alpha_k^u\right)\right\} \quad (16)$$

$$\alpha_{and}^l = \min_{\text{all } k}\left\{\max\left(\max_{i\neq k}\left\{\alpha_k^l \overline{\oslash} \alpha_i^u + B_k - B_i\right\}, \alpha_k^l\right)\right\} \quad (17)$$

The maximal delay and backlog of input i are bounded by

$$d_i = H\left(\alpha_i^u + B_i, \min_{j\neq i}\left\{\alpha_j^l + B_j\right\}\right) \quad (18)$$

$$b_i = \max\left(V\left(\alpha_i^u + B_i, \min_{j\neq i}\left\{\alpha_j^l + B_j\right\}\right), 0\right) \quad (19)$$

Proof: We first prove (18). Let $R_i(t)$ denote the accumulated amount of arrived events of input i from time 0 to t. The total amount of available events of input i until time t is $R_i(t) + B_i$. Then $\min_{i\neq j}\{R_j(t) + B_j\}$ is the minimum of the available events among all the other inputs. So the maximal delay of the event backlogged at input i at time t is

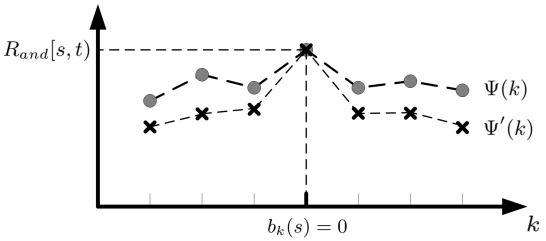


Fig. 6. The relation between $\Psi(k)$ and $\Psi'(k)$.

$$\begin{aligned} d_i(t) =& \inf\{\tau \geq 0 : R_i(t) + B_i \leq \min_{i\neq j}\{R_j(t+\tau) + B_j\}\} \\ \leq& \sup_{\lambda\leq 0}\{\inf\{\tau \geq 0 : R_i(\lambda) + B_i \leq \min_{i\neq j}\{R_j(\lambda+\tau) + B_j\}\} \\ \leq& \sup_{\lambda\leq 0}\{\inf\{\tau \geq 0 : \alpha_i^u(\lambda) + B_i \leq \min_{i\neq j}\{\alpha_j^l(\lambda+\tau) + B_j\}\} \\ =& H(\alpha_i^u + B_i, \min_{i\neq j}\{\alpha_j^l + B_j\}) \end{aligned}$$

Now we prove (19). The total amount of output events by time t is $R'(t) = \min_{\text{all } j}\{R_j(t) + B_j\}$, so the buffer of input i at time t is

$$\begin{aligned} b_i(t) &= [R_i(t) - R'(t)]^0 \\ &= [R_i(t) + B_i - \{\min_{\text{all } j}\{R_j(t) + B_j\}\}]^0 \\ &= [R_i(t) + B_i - \{\min_{i\neq j}\{R_j(t) + B_j\}\}]^0 \end{aligned}$$

Therefore, we have

$$b_i(t) = [\max_{i\neq j}\{R_i(t) - R_j(t) + B_i - B_j\}]^0 \quad (20)$$

By applying Lemma 4 to this, we get

$$\begin{aligned} b_i(t) &\leq [\max_{i\neq j}\{\alpha_i^u \oslash \alpha_j^l(0) + B_i - B_j\}]^0 \\ &= [\max_{i\neq j}\{(\alpha_i^u + B_i) \oslash (\alpha_j^l + B_j)(0)\}]^0 \\ &= \max_{i\neq j}\{V(\alpha_i^u + B_i, \alpha_j^l + B_j)\} \\ &= \max\{V(\alpha_i^u + B_i, \min_{i\neq j}\{\alpha_j^l + B_j\}), 0\} \end{aligned}$$

In the following we prove (16). $R_{and}[s,t)$ denotes the amount of output combined event generated in time interval $[s,t)$, which equals the minimum among all the inputs:

$$R_{and}[s,t) = \min_{\text{all } i}\{R_i[s,t) + b_i(s)\} \quad (21)$$

where $b_i(s)$ is the buffer level of input i at time s.

In the following we prove

$$\min_{\text{all } i}\{R_i[s,t) + b_i(s)\} = \max_{\text{all } k}\{\Psi(k)\} \quad (22)$$

where

$$\Psi(k) = \min\left(\min_{i\neq k}\left\{R_i[s,t) + b_i(s)\right\}, R_k[s,t)\right)$$

By the definition of $\Psi(k)$ we know that for any $k \in [1,n]$

$$\Psi(k) \leq \min_{\text{all } i}\{R_i[s,t) + b_i(s)\} \quad (23)$$

On the other hand, at least one of $b_1(s), b_2(s), \cdots, b_n(s)$ must be 0. Without loss of generality, let $b_x(s) = 0$, then by the definition of Ψ we have

$$\Psi(x) = \min_{\text{all } i}\{R_i[s,t) + b_i(s)\} \tag{24}$$

In summary, the LHS of (22) is no smaller than $\Psi(k)$ for all $k \in [1, n]$, and there exists at least one $\Psi(x)$ that equals to the LHS (22), by which (22) is proved. Combining (21) and (22) yields

$$R_{and}[s,t) = \max_{\text{all } k}\{\Psi(k)\} \tag{25}$$

In the following we will derive an upper bound for $\max_{\text{all } k}\{\Psi(k)\}$. Note that we will derive an upper bound for the entire $\max_{\text{all } k}\{\Psi(k)\}$, rather than upper bounding $\Psi(k)$ for each k and then getting their maximum.

By applying (20) to $\Psi(k)$, we have

$$\Psi(k) = \min\left(\min_{i\neq k}\{R_i[s,t) + [R_i(s) + B_i - \sigma]^0\}, R_k[s,t)\right)$$

where

$$\sigma = \min_{i\neq j}\{R_j(s) + B_j\}$$

We define another function $\Psi'(k)$ respect to k as follows:

$$\Psi'(k) = \min\left(\min_{i\neq k}\{R_i[s,t) + [R_i(s) + B_i - \sigma']^0\}, R_k[s,t)\right)$$

where

$$\sigma' = R_k(s) + B_k$$

Now we discuss the relation between $\Psi'(k)$ and $\Psi(k)$. First, since $\sigma \leq \sigma'$, we know the general relation between $\Psi'(k)$ and $\Psi(k)$:

$$\Psi'(k) \leq \Psi(k)$$

Then we focus on the relation between $\Psi'(k)$ and $\Psi(k)$ with a particular k satisfying $b_k(s) = 0$ In this case, $R_k(s) + B_k$ must be no larger than $R_i(s) + B_i$ for any i, which implies $\sigma = \sigma'$. Therefore, we know $b_k(s) = 0$ implies

$$\Psi(k) = \Psi'(k)$$

Moreover, by the definition of $\Psi(k)$, we know $\Psi(k)$ reaches its maximal value with k if $b_k(s) = 0$, i.e.,

$$\Psi(k) = \min_{\text{all } i}\{R_i[s,t) + b_i(s)\} = \max_{\text{all } k}\{\Psi(k)\} \tag{26}$$

Putting the above discussions together, the relation between $\Psi'(k)$ and $\Psi(k)$ can be summarized as follows:

In general $\Psi'(k) \leq \Psi(k)$, while both of them reach the same maximal value with a particular k satisfying $b_k(s) = 0$. Moreover, there must exist $b_k(s) = 0$ among $\{b_1(s), b_2(s), \cdots, b_n(s)\}$ since at any time point at least one of the stream buffers must be empty. These relations are illustrated in Figure 6. Therefore, we can conclude that

$$\max_{\text{all } k}\{\Psi(k)\} = \max_{\text{all } k}\{\Psi'(k)\} \tag{27}$$

In the following, we compute an upper bound for $\max_{\text{all } k}\{\Psi'(k)\}$:

$$\begin{aligned}&\max_{\text{all } k}\{\Psi'(k)\}\\ =&\max_{\text{all } k}\{\min(\min_{i\neq k}\{R_i[s,t) + [R_i(s) + B_i - \sigma']^0\}, R_k[s,t))\}\end{aligned}$$

$\Psi'(k)$ reaches the maximal value with k satisfying $b_k(s) = 0$. If $b_k(s) = 0$, we know $R_i(s) + B_i - R_k(s) - B_k \geq 0$, i.e., $R_i(s) + B_i - \sigma' \geq 0$, so the above equation is rewritten as

$$\begin{aligned}&\max_{\text{all } k}\{\Psi'(k)\}\\ =&\max_{\text{all } k}\{\min(\min_{i\neq k}\{R_i(t) - R_k(s) + B_i - B_k\}, R_k[s,t))\}\end{aligned}$$

and finally by (25), (27) and Lemma 4 we have

$$\begin{aligned}&R_{and}[s,t)\\ \leq&\max_{\text{all } k}\{\min(\min_{i\neq k}\{\alpha_i^u \oslash \alpha_k^l + B_i - B_k\}, \alpha_k^u)\}(t-s)\end{aligned}$$

By now we have proved (16) for the upper output curve.

In the following we prove (17) for the lower output curve.

The amount of output events in $[s,t)$ is the minimum amount of events among all streams in this time interval:

$$\begin{aligned}&R_{and}[s,t) = \min_{\text{all } k}\{R_k[s,t) + b_k(s)\}\\ =&\min_{\text{all } k}\left\{R_k[s,t) + [\max_{i\neq k}\{R_k(s) - R_i(s) + B_k - B_i)\}]^0\right\} \textit{// by (19)}\\ =&\min_{\text{all } k}\left\{\max(R_k[s,t), \max_{i\neq k}\{R_k(t) - R_i(s) + B_k - B_i\})\right\}\end{aligned}$$

By $R_k[s,t) \geq \alpha_k^l(t-s)$ and Lemma 4, we get

$$R_{and}[s,t) \geq \min_{\text{all } k}\left\{\max\left(\max_{i\neq k}\{\alpha_k^l \overline{\oslash} \alpha_i^u + B_k - B_i\}, \alpha_k^l\right)\right\}$$

■

VII. Experimental Evaluation

We implement our new theoretical results in RTC Toolbox [14] and conduct experiments to evaluate their performance. Experiments are conducted a computer with a 2.50GHZ Intel Core i7 processor and 4.00 GB RAM.

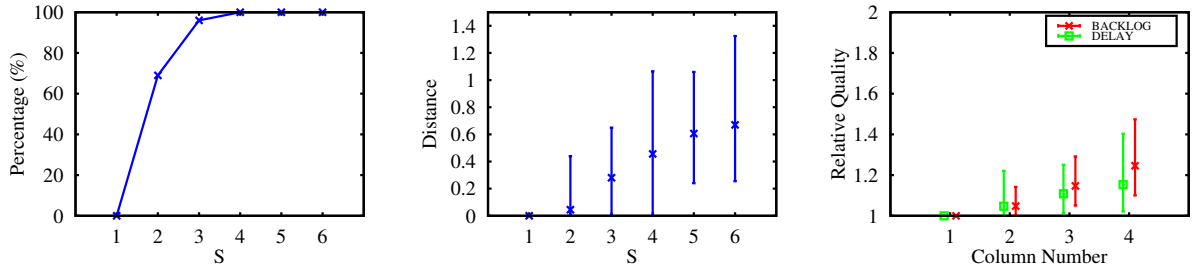
A. Evaluation for GPC

We first evaluate the analysis precision improvement for GPC. We compare two methods to compute output curves:

- GPC-Old: The original methods to compute the upper and lower output arrival curves using (3) and (4).
- GPC-New: Computing the upper output arrival curve using our new method in (13) and the lower output arrival curve using the original method (4).

The input arrival curves are generated following the PJD workload model [15] characterized by (p, j, d):

$$\alpha^u(\Delta) = \min\left(\left\lceil\frac{\Delta + j}{p}\right\rceil, \left\lceil\frac{\Delta}{d}\right\rceil\right), \quad \alpha^l(\Delta) = \left\lfloor\frac{\Delta - j}{p}\right\rfloor$$



(a) The percentage of improved experiments. (b) Distance. (c) Relative Quality.

Fig. 7. Experiment results for GPC.

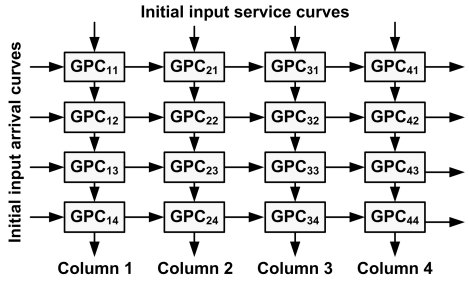


Fig. 8. The 4×4 RTC network.

The service curves are generated following the TDMA model [14] characterized by (s, c, b):

$$\beta^u(\Delta) = \left(\left\lfloor \frac{\Delta}{c} \right\rfloor \cdot s + \min(\Delta \bmod c, s) \right) \cdot b$$
$$\beta^l(\Delta) = \left(\left\lfloor \frac{\Delta'}{c} \right\rfloor \cdot s + \min(\Delta' \bmod c, s) \right) \cdot b$$

where $\Delta' = \max(\Delta - c + s, 0)$.

Figure 7-(a) and (b) show the experiment results with a single GPC. The input arrival curves are randomly generated by selecting the p, j and d values in the following ranges: $p \in [20, 50]$, $i \in [10, 100]$ and $d \in [1, 10]$. The input service curves have a fixed $c = 60$ and $b = 1$, and s varies for different groups of experiments (corresponding to the x-axis). With each s value, we perform 1000 experiments with both methods.

Figure 7-(a) reports the percentage of generated GPC for which the output curve obtained by GPC-New is more precise than GPC-Old. When $s = 1$, the long-term slope of the service curves is smaller than the arrival curves, with which the two methods perform the same. As s increases, the resource becomes more sufficient, and the percentage of experiments in which GPC-New is more precise becomes higher. Eventually, when $s \geq 4$, the long-term slope of resource curves is always larger than the arrival curves, and our new method always yields more precise results than GPC-Old.

Figure 7-(b) reports the *distance* between the upper output arrival curves obtained by the two methods. The distance of two curves f and g is defined as follows:

$$dist(f, g) = \frac{\sum_{\Delta=1}^{n} |(f(\Delta) - g(\Delta))|}{n}$$

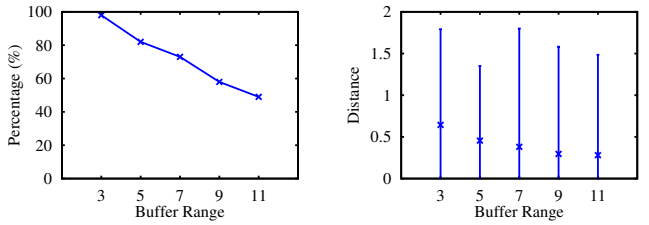
Figure 7-(b) reports the results of $dist(\alpha^u_{old}, \alpha^u_{new})$ with $n = 200$ and different s. For each s, the upper and lower ends of the vertical segment represent the maximal and minimal distance, and the cross symbol in the middle represents the average distance of all the experiments with this s value. In general, the distance between GPC-New and GPC-Old is larger when s increases. In summary, Figure 7-(a) and (b) show that the precision improvement of our new upper output arrival curves is more significant with more sufficient resource.

In RTC, the final goal is to compute the backlog and delay bounds of the event streams. Therefore, the precision improvement in the output curves is meaningful only if it leads to more precise backlog and delay bounds. Therefore, we evaluate the backlog and delay bounds of a 4×4 RTC network, as shown in Figure 8. The initial input arrival curves are randomly generated in the same way as the above experiments, and the initial input service curves are generated with $s = 20$, $c = 60$ and $b = 1$. Figure 7-(c) shows the ratio between the delay (backlog) bounds obtained using GPC-Old and those obtained by GPC-New, namely the *relative quality*. Each result for x-axis value i is the average of the delay (backlog) relative quality of components in the i^{th} column. Similar to Figure 7-(b), the results include the minimal, maximal and average relative quality for each group of experiments. We can see that the precision improvement for delay (backlog) bounds using our new output arrival curves is more significant for the downstream components. This is because the precision gain in output curves by our new method will be accumulated as the curves propagate in the network.

B. Evaluation for Dual-Input AND

Recall that the negative value problem of the original lower output curve in (8) can be fixed by changing the negative values to 0, but the resulting output curves are still pessimistic. Our new method does not only systematically solve the negative value problem, but also can provide tighter results. Next we evaluate the precision improvement of our new lower output curves for dual-input AND connectors in (14). We compare two methods to compute the lower output curves:

- AND-Naive: The lower output curve obtained by fixing the negative value problem in the original lower output curve in (8) by simply changing the negative values to 0.
- AND-New: Our new lower output curve in (14).



(a) Percentage. (b) Distance.

Fig. 9. Experiment results for dual-input AND.

In all the experiments with AND (including the multi-input AND in the next subsection), we use a revised version of the PJD event model to generate input curves. For AND, if the long-term slope of the input curves are different, the events backlogged in the buffer of the one with lower slope will increase infinitely. Therefore, AND is typically used to join curves with the same long-term slope. However, in the original PJD model [15], the long-term slope of a curve only depends on the parameter p, which represents the period of the events. Therefore, with the original PJD model, all the input curves to AND must have the same period, which not only represents a very special case but may lead to biased comparison results. In order to cover more general cases and make our results more convincing, we extend the PJD model from three parameters (p, j, d) to four parameters (p, j, d, r):

$$\alpha^u(\Delta) = \min\left(\left\lceil\frac{(\Delta+j)r}{p}\right\rceil, \left\lceil\frac{\Delta}{d}\right\rceil\right), \quad \alpha^l(\Delta) = \left\lfloor\frac{(\Delta-j)r}{p}\right\rfloor$$

With the extended model, the long-term slope of the curves depends on the value of $\frac{r}{p}$. Therefore, we can generate curves with the same long-term slope but different periods.

The input curves are randomly generated by selecting the p, j and d values in the same ranges as the experiments in Figure 7-(a) and (b). For the two input curves of an AND, their r values are derived so that the long-term slopes of the two curves are the same. The initial buffer level of each input stream is randomly chosen in the range $[1, x]$, where x corresponds to the x-axis in Figure 9. Similar to Figure 7-(a) and (b), Figure 9-(a) reports the percentage of experiments in which AND-New is more precise than AND-Naive, and Figure 9-(b) reports the distance between the lower output curves obtained by AND-New and AND-Naive. The results show that the precision improvement of our new results is more significant when the initial buffer level is smaller.

C. Evaluation for Multi-Input AND

Next we evaluate the generalization of AND to multiple inputs. We first compare the output curves obtained by the cascaded approach and the holistic approach:

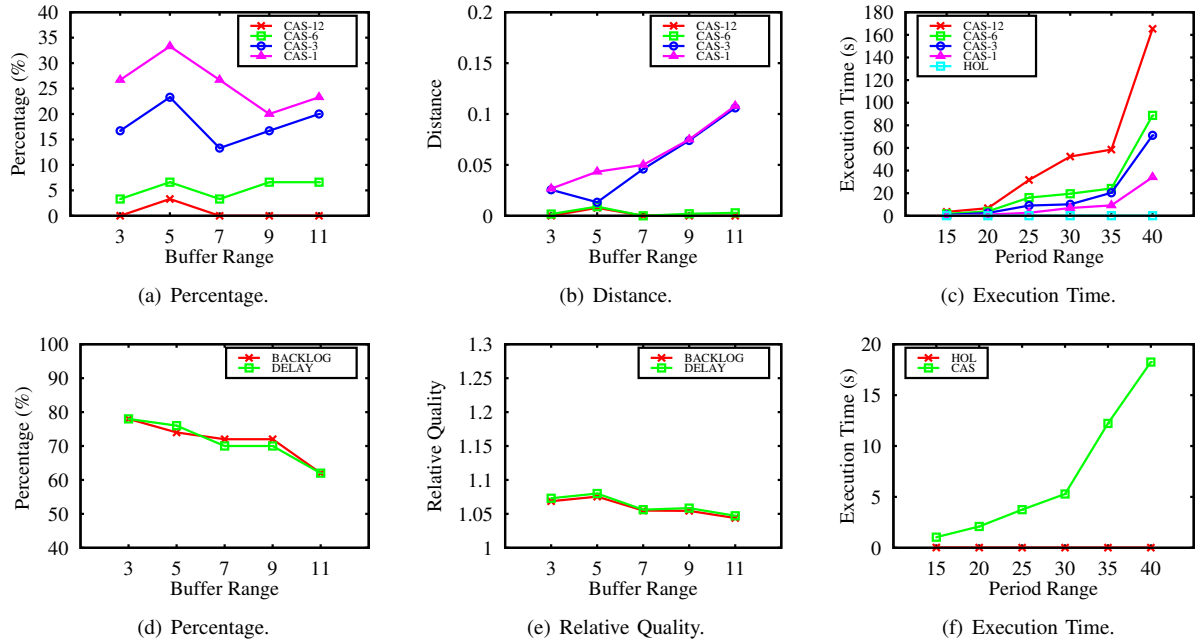
- CAS-x: The cascaded approach to compute the output curves in (15). Recall that the cascaded approach is sensitive to the order to apply dual-input AND to the event streams. The result with any order provides valid upper and lower output curves, while joining the results with different orders in general may improve the precision. CAS-x represents the final results obtained by joining results with x randomly selected orders.
- HOL: The holistic approach to compute the output curves using (16) and (17).

Figure 10 shows experiment results with AND with four input streams. There are in total $4!/2 = 12$ different cascading orders for a four-input AND. Figure 10-(a) reports the percentage of experiments that HOL is more precise than CAS-x, and Figure 10-(b) reports the distance between the output curves obtained by CAS-x and HOL, with x being 1, 3, 6 and 12, respectively. The input curves are generated in the same way as in Section VII-B.

We also compare the time consumed by the analysis of each four-input AND by different methods in Figure 10-(c). The input curves are generated in a similar way with above, and the only difference is that we change the range for selecting the period p: the lower bound is 10, while the upper bound is $15, 20, \cdots, 40$ (the values on the x-axis). The initial buffer level is in the range $[1, 5]$. The experiment results show that the HOL method is consistently efficient: it on average takes less than 0.1 second, and rarely exceeds 1 second. However, the efficiency of the cascaded approach is much lower. Even CAS-1 (only one cascading order is analyzed) is much slower than HOL. The time consumption of CAS-x increases exponentially as the range of periods increases.

The low efficiency of the cascaded approach is because of the "period explosion" problem [16]. In RTC, the curves are conceptually infinite, which are not implementable. Practical implementations of RTC, such as the RTC Toolbox [16], are restricted to a class of regular curves [16], which can be efficiently represented by finite data structures but are still expressive enough to model most realistic problems. A regular curve consists of an aperiodic part, followed by a periodic part. Each part is represented by a concatenation of linear segments. Generally, the computation time and memory requirement of an operation between two curves are proportional to the number of segments contained by the curves. The number of segments of a curve representing a PJD event model is generally proportional to its period. The period of the output curve of a dual-input AND is the hyperperiod of the two input curves. In the cascaded approach, the period of the event stream increases exponentially as it travels through the dual-input AND, which leads to the low efficiency of the cascaded approach. When periods of the input curves are selected from a wider range, the resulting hyperperiod of them is larger, and thus the efficiency of the cascaded approach is worse. By contrast, the holistic approach in (16) and (17) only performs corresponding operations on each pair of the input curves, which avoids the above "period explosion" problem.

Finally, we compare the precision and computation efficiency of the delay (backlog) bounds using the cascaded method (Theorem 4) and holistic method (Theorem 5) for four-input AND. The input curves are generated in the same way as the corresponding experiments in above. Figure 10-(d) and (e) report the percentage of experiments where the holistic approach gives more precise results and ratio between



(a) Percentage. (b) Distance. (c) Execution Time.

(d) Percentage. (e) Relative Quality. (f) Execution Time.

Fig. 10. Experiment results for multi-input AND.

the delay (backlog) bounds by the cascaded approach and by the holistic approach. Figure 10-(f) gives the time consumption of the two approaches, where the holistic approach on average takes less than 0.1 second, while the time consumption of the cascaded approach is much longer and increases exponentially as the period range is larger. Note that to compute the delay and backlog bounds of an event stream in a four-input AND, we only need to compute the output curve joining the remaining three streams, so the time consumption is lower than the experiments in Figure 10-(c) which join all the four streams. In summary, to generalize AND to support multiple inputs, our new holistic approach is not only more precise but also significantly more efficient than the naive approach by cascading dual-input AND.

VIII. Conclusion

In this paper, we improve the two widely used components, GPC and AND, in real-time calculus. First, we develop a more precise method to compute the upper output arrival curve of GPC. The key idea of our new method is to use the remaining service curves to refine the information about how much resource is actually consumed to process the input events. Second, we identify and fix a problem in the existing analysis method of AND. Third, we study the analysis of AND connectors with more than two inputs (called multi-input AND). We first present a straightforward generalization approach by modeling a multi-input AND as several cascaded dual-input AND, then present a holistic approach for the generalization which is more precise and efficient.

References

[1] L. Thiele, S. Chakraborty, and M. Naedele., "Real-time calculus for scheduling hard real-time systems." in *Proc. Inti. Symposium on Circuits and Systems*, 2000.

[2] S. Chakraborty, S. Knzli, and L. Thiele, "A general framework for analysing system properties in platform-based embedded system designs," in *DATE*. Springer Verlag, 2003.

[3] J. L. Boudec and P. Thiran, "Network calculus - a theory of deterministic queuing systems for the internet," in *LNCS 2050*. Springer Verlag, 2001.

[4] E. Wandeler, "Modular performance analysis and interface-based design for embedded real-time systems," in *PhD thesis, ETHZ*, 2006.

[5] S. Chakraborty, L. T. X. Phan, and P. S. Thiagarajan, "Event count automata: a state-based model for stream processing systems," in *RTSS*, 2005.

[6] E. Fersman, L. Mokrushin, P. Pettersson, and W. Yi, "Schedulability analysis of fixed-priority systems using timed automata," in *Theor. Comput. Sci.*, 2006.

[7] Y. Abdeddaim, A. Kerbaa, and O. Maler, "Task graph scheduling using timed automata," in *Proceedings International Parallel and Distributed Processing Symposium*, 2003.

[8] L. T. X. Phan, S. Chakraborty, P. S. Thiagarajan, and L. Thiele, "Composing functional and state-based performance models for analyzing heterogeneous real-time systems," in *RTSS*, 2007.

[9] K. Lampka, S. Perathoner, and L. Thiele, "Analytic real-time analysis and timed automata: a hybrid methodology for the performance analysis of embedded real-time systems," *Design Automation for Embedded Systems*, vol. 14, no. 3, pp. 193–227, 2010.

[10] H. Bauer, J.-L. Scharbarg, and C. Fraboul, "Applying and optimizing trajectory approach for performance evaluation of afdx avionics network," in *ETFA*, 2009.

[11] A. Bouillard, "Algorithms and efficiency of network calculus," Ph.D. dissertation, Ecole Normale Supérieure (Paris), 2014.

[12] R. Henia, A. Hamann, M. Jersak, R. Racu, K. Richter, and R. Ernst, "System level performance analysis - the symta/s approach," *IEE proceedings-Computers and Digital Techniques*, vol. 152, no. 2, pp. 148–166, 2005.

[13] K. Lampka, S. Perathoner, and L. Thiele, "Analytic real-time analysis and timed automata: a hybrid method for analyzing embedded real-time systems," in *EMSOFT*, 2009.

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

[15] K. Richter, "Compositional scheduling analysis using standard event models," in *In Ph.D. Thesis, Technical University Carolo-Wilhelmina of Braunschweig*. Springer Verlag, 2005.

[16] N. Guan and W. Yi, "Finitary real-time calculus: Efficient performance analysis of distributed embedded systems," in *RTSS*, 2013.