Upper Bound for Delay Densities

Florian Hock
Ulm University
Ulm
florian.hock@uni-ulm.de

Chijun Shen
Ulm University
Ulm
chijun.shen@uni-ulm.de

Victor Pollex
Ulm University
Ulm
victor.pollex@uni-ulm.de

Tobias Bund
Ulm University
Ulm
tobias.bund@uni-ulm.de

Frank Slomka
Ulm University
Ulm
frank.slomka@uni-ulm.de

Abstract.

In networked control systems, sensors and actuators are connected via networks to the controller platform. Contrary to direct links, delays vary in a more or less wide range in network links. Hence, delays are one of the key issues in the design process of a networked control system. Therefore, obtaining safe upper bounds is essential to guarantee the desired behavior of the system. On the other hand, the bounds should be as tight as possible. Large overestimations result in over-dimensioned platforms and networks. A common approach is to assume the occurrence of the worst case response time (WCRT) for all transmissions. This simplifies the design process as the WCRT can be derived directly from the platform model. Though, it contains inherently large overestimations.

A more accurate approach is featured by delay densities. By not treating each event individually, delay densities give a description of the timing behavior within intervals. Additionally, the method is not limited to strictly periodic events.

In this paper, we propose a method for deriving an upper bound from a platform description in Real-Time-Calculus.

1. Introduction

The behavior of networked control systems is dependent on the physical context. It is necessary that the systems comply with the characteristics derived from that context to guarantee a desired behavior. But, on the other hand, the hardware platform should be designed as slim as possible to limit costs or energy consumption. Both objectives compete with each other in the design process. It has to be guaranteed that the task set which is executed on the computation platform is schedulable under all circumstances. Also, the controllers' state has to stay within its specifications. For both objectives, the timing behavior of the system turns out as the key figure. However, the correlation

between the timings of the physical process, which shows a continuous behavior, and the discrete computation process is not fully understood yet. This means, simplifications have to be made to bound execution times and network latencies, which we term as *delays* in the following. The delay bound has to be a safe upper bound which should be as tight as possible to avoid poor utilization.
The maximum delay is one option for the upper bound. To calculate the maximum delay, several methods exist in real-time analysis techniques. However, accounting the maximum delay for all instances leads to an overestimation of the actual timing behavior. This is due to the issue, that the maximum delay occurs rarely during the runtime of the control system. Besides, sporadic longer delays may not result in violation of the performance specifications.
In [3], Bund et al. proposed the delay density model which is more realistic and offers tighter bounds than the worst-case response time (WCRT). Bund et al. show how delay densities can be build from sequences of delays in the time domain. From those sequences, minimum and maximum cumulative delay functions are derived in the interval domain. The model is included in [2] in a complete set of densities, in particular the signal and disturbance density and the density of dropped samples. Those densities allow to bound the influence of disturbances, delays and dropped samples on the control quality. However, there is no method given to calculate safe upper bounds for delay densities directly from a system's specification, given by a task set and scheduling. In this paper we propose a safe upper bound (as described in: [11]), which can be calculated by utilizing the principles of Real-Time-Calculus (RTC).
The paper is structured into four sections. We give a brief overview about the related work in section 2. This is followed by the definition of the delay density function as described in [3] and a short introduction to the Real-Time Calculus [13] in subsection 3.2. In section 4 we describe our algorithms for the calculation of upper bound for delay densities.

2. Related Work

Several approaches have been published in recent years to address the problem of describing the timing behavior more realistic than the WCRT does. In [9] the delay analysis is based on knowledge of the probability distribution of delays. Lincoln et al. [7] propose to use a Markov Jump Linear System to model a closed-loop digital control system. Both are based on stochastic models and do not provide a guaranteed bound for the delay. However, safe bounds are mandatory when designing safety critical systems [4]. A method to find the demand bound function for a schedulability analysis for self triggered controllers is proposed by Aminifar et al. [1]. This is done by partitioning the state space into subregions. For each subregion the maximum time the system could run in open-loop until it becomes unstable is determined. Then a transition graph is created to model the possibility of transitions from one subregion to another. The transition graph is finally used to calculate the demand bound function by finding the shortest interval with a minimum number of events. Xu et al. propose methods for scheduling and control co-design in [14] and [15]. The methods are based on minimizing costs for LQ-controllers by the determination of optimal periods. Both publications are focused on the issue of event generation and delays are modeled either as upper bounded or averaged values. In [5] a maximum rate on stable and unstable samples is derived from a control system. A delay threshold is set to classify if a sample is stable or not. [8] proposes a method to model the time delay as a constraint of the estimator. Eliminating the delay in the system, the controller can be parameterized using methods for delay-free systems. The delay itself is a constant delay. In [12], Tolic and Hirche examine the gain in stability by using an estimator. Delays are

treated separately for both directions, from plant to estimator and back. Stability is proven by the Ljapunov method, which implies using constant upper bounds.

3. System Model

In this paper we consider an event triggered system consisting of $n \in \mathbb{N}$ tasks. For each task the events that trigger it are enumerated chronologically. Each event causes the creation of an instance or job of that task. This instance is then processed according to the scheduler that is used. Since the event that creates an instance of a task and that instance itself can be mapped bijectively, we will be using the indexed event and its corresponding instance synonymously.

3.1. Delay Density

Given these assumptions a delay function can be defined for each task which maps each of these events to the delay of their corresponding instance.

Definition 1 *(Delay Function). Let k be a natural number ($k \in \mathbb{N}$) and let a_k be a non-negative real number ($a_k \in \mathbb{R}_0^+$) that denotes the point in time at which the k-th event occurs. Similarly let e_k be a non-negative real number ($e_k \in \mathbb{R}_0^+$) that denotes the point in time at which the instance that was generated by the k-th event has been completely processed. The delay function $r\colon \mathbb{N} \to \mathbb{R}_0^+$ maps the k-th event to its delay and is defined as*

$$r(k) := e_k - a_k. \tag{1}$$

With the delay function a cumulative delay function δ can now be defined which represents the sum of the delays of the first k events.

Definition 2 *(Cumulative Delay Function). Let $r\colon \mathbb{N} \to \mathbb{R}_0^+$ denote a delay function (1), then the cumulative delay function $\delta\colon \mathbb{N}_0 \to \mathbb{R}_0^+$ is defined as*

$$\delta(k) := \sum_{j=1}^{k} r(j). \tag{2}$$

Note that for $k = 0$ we have the empty sum, hence $\delta(0) = 0$.

Based on the cumulative delay function the delay density is described.

Definition 3 *(Delay Density). Let $\delta\colon \mathbb{N}_0 \to \mathbb{R}_0^+$ denote a cumulative delay function (2), then the delay density $dR^+\colon \mathbb{N}_0 \to \mathbb{R}_0^+$ satisfies for all $k, \Delta \in \mathbb{N}_0$*

$$\delta(k + \Delta) - \delta(k) \leq dR^+(\Delta). \tag{3}$$

For a fixed non-negative integer Δ the delay density $dR^+(\Delta)$ denotes an upper bound of any sum of delays of Δ consecutive instances.

Provided that the cumulative delay function is known, a valid delay density is

$$dR^+(\Delta) = \sup_{k \in \mathbb{N}_0} \{\delta(k + \Delta) - \delta(k)\} = (\delta \oslash \delta)(\Delta). \tag{4}$$

The operator $\oslash$ is defined as:

Definition 4 *((∧, +)-deconvolution). Let $f, g: \mathbb{R}_0^+ \to \mathbb{R}_0^+$ be two increasing functions, then the (∧, +)-deconvolution is defined as*

$$(f \oslash g)(x) := \sup_{\lambda \in \mathbb{R}_0^+} \{f(x + \lambda) - g(\lambda)\} \tag{5}$$

3.2. Real-Time Calculus

Wandeler [13] has proposed the real-time calculus, a modular framework to analyze real-time systems. Its core element is the greedy processing component (GPC) as shown in Figure 1. A GPC

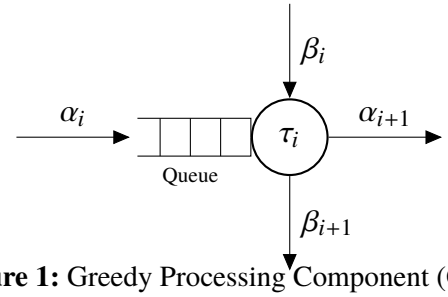

Figure 1: Greedy Processing Component (GPC)

is used to model an entity that requires resources, e.g. a message on a bus or a task that is executed on a processor. Either way, a GPC τ_i has as input the arrival curves α_i and the service curves β_i. These curves come in pairs, one describing a lower bound and the other describing an upper bound.

The arrival curves represent bounds on the frequency with which the GPC is being activated.

Definition 5 *(Arrival Curves, cf. Definition 1 in [13]). Let t and Δ be non-negative real numbers ($t, \Delta \in \mathbb{R}_0^+$) and let $R[t, t + \Delta)$ denote the amount of events that occur in the interval $[t, t + \Delta)$. Then the lower bound α^l and the upper bound α^u of the arrival curve satisfies for all $t, \Delta \in \mathbb{R}_0^+$ the condition*

$$\alpha^l(\Delta) \leq R[t, t + \Delta) \leq \alpha^u(\Delta). \tag{6}$$

Similarly, the service curves represent bounds on the amount of resources that a GPC is being assigned to according to the used arbitration method or scheduler.

Definition 6 *(Service Curves, cf. Definition 2 in [13]). Let t and Δ be non-negative real numbers ($t, \Delta \in \mathbb{R}_0^+$) and let $C[t, t + \Delta)$ denote the amount of resources that are available in the interval $[t, t + \Delta)$. Then the lower bound β^l and the upper bound β^u of the service curve satisfies for all $t, \Delta \in \mathbb{R}_0^+$ the condition*

$$\beta^l(\Delta) \leq C[t, t + \Delta) \leq \beta^u(\Delta). \tag{7}$$

Usually service curves are provided in resource units, e.g. $\beta^l(\Delta)$ describes how many resources have been assigned at least to the GPC in any interval of length Δ. However in this work we assume that the service curves are provided in event units, e.g. $\beta^l(\Delta)$ describes the amount of events that can be processed at least in any interval of length Δ. If a service curve is provided in resource units $\overline{\beta}$ and the GPC has a worst-case execution time of c^+ and a best-case execution time of c^-, then the service curve in event units can be obtained through a division, i.e. $\beta^l = \frac{\overline{\beta}^-}{c^+}$ and $\beta^u = \frac{\overline{\beta}^+}{c^-}$, cf. Equation (4.4) in [13].

Besides its input, a GPC has an output for the outgoing arrival curves α_{i+1} as well as the remaining service curves β_{i+1}. In this work we only use the lower bound of the remaining service curve. The equations for the other bounds can be found in [13].

Proposition 1 *(Remaining Service Curve, cf. Equation (2.10) in [13]). Let α_i^u be the upper bound of the arrival curve and β_i^l the lower bound of the service curve, then the lower bound of the remaining service curve β_{i+1}^l is*

$$\beta_{i+1}^l(\Delta) = \sup_{0 \leq \lambda \leq \Delta} \left\{ \beta_i^l(\lambda) - \alpha_i^u(\lambda) \right\}. \tag{8}$$

With the arrival and the service curves, an upper bound of the delay of a GPC can be expressed by means of the greatest horizontal distance.

Definition 7 *(Greatest Horizontal Distance, cf. Equation (2.11) in [13]). Let $f, g : \mathbb{R}_0^+ \to \mathbb{R}_0^+$ be two increasing functions, then the greatest horizontal distance between them is defined as*

$$(f \leftrightarrow g) := \sup_{\lambda \in \mathbb{R}_0^+} \left\{ \inf_{\mu \in \mathbb{R}_0^+} \{ \mu : f(\lambda) \leq g(\lambda + \mu) \} \right\}. \tag{9}$$

Proposition 2 *(Delay Bound, cf. Equation (2.11) in [13]). Let α^u be an upper bound of the arrival curve and β^l the lower bound of the service bound, then the delay of a single instance is bounded by*

$$r^+ := (\alpha^u \leftrightarrow \beta^l). \tag{10}$$

Note that any delay bound r^+ of a GPC τ satisfies $r(k) \leq r^+$ for every $k \in \mathbb{N}_0$.

4. Deriving Delay Densities

Let τ be a greedy processing component for which its arrival curves and services curves are known. Then we first derive a delay density that is based on a delay bound. Afterwards we improve the delay density by using additional information that is contained in the arrival and service curves which was previously ignored. We want to show this by a running example:

Example 1 *The activation of the GPC is modeled by the model denoted as periodic events with burst (Section 4.3 of [10]) with the parameters period $T = 150$, jitter $J = 450$, and minimum event distance $d = 15$. The GPC requires at most $c^+ = 20$ resource units to process an instance and is provided resources through a time division multiple access (TDMA) scheme where the cycle length is $c = 10$ time units long, the assigned slot for this GPC is $s = 6$ time units long, and per time unit $b = 1$ resource unit is provided. This example describes essentially task* T1 *of the distributed example presented in [6]. Only the parameter minimum event distance was modified to have a value of $d = 15$ instead of the original value $d = 0$ in order to visually distinguish the different events more easily in the figure. With the provided parameters of the activation model the upper arrival curve (Equation 4.14 in [10]) is*

$$\alpha^u(\Delta) = \min \left\{ \left\lceil \frac{\Delta + J}{T} \right\rceil, \left\lceil \frac{\Delta}{d} \right\rceil \right\}$$

and the lower service curve is

$$\beta^l(\Delta) = \left(\left\lfloor \frac{\Delta}{c} \right\rfloor \cdot s + \max \left\{ \Delta - \left\lfloor \frac{\Delta}{c} \right\rfloor \cdot c - (c - s), 0 \right\} \right) \frac{b}{c^+}.$$

The curves of this example are shown in Figure 2.

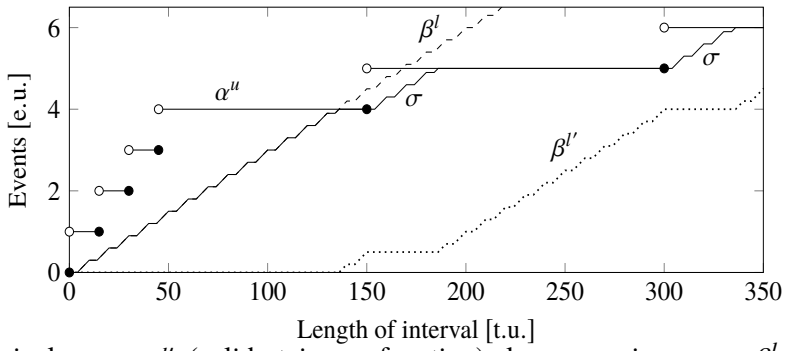

Figure 2: Upper arrival curve α^u (solid staircase function), lower service curve β^l (dashed continuous function), the service spent for processing the events σ (solid continuous function), and the remaining lower service curve $\beta^{l'}$ (dotted continuous function).

4.1. Delay Density Based on Delay Bound

A simple bound for the delay density of a GPC is based on a delay bound r^+, which we refer to as delay bound based delay density (DB-DD). This delay bound can be obtained for example through (10).

Theorem 1 *Let τ be a GPC and let r^+ be a delay bound for it, then a delay density for the GPC τ is*

$$dR^+(\Delta) = \Delta \cdot r^+. \tag{11}$$

The DB-DD assumes that every instance requires the same amount of time to process as the delay bound. Provided that the delay bound is the smallest upper bound of the delay function, i.e. $r^+ = \sup_{k\in\mathbb{N}}\{r(k)\}$ then it is possible that this assumption is true. However it appears to be improbable that this assumption is accurate for most applications. Hence we expect that the delay density as presented in (11) is severely over-approximated in most cases.

4.2. Delay Density Based on Delay Function

The delay density presented in 4.1 is based on a delay bound which can be determined by the greatest horizontal distance between the upper arrival curve and the lower service curve. However, within these two curves additional information is provided than merely a delay bound, namely a delay function can be derived. We use this to improve the delay density of a GPC, which we refer to as delay function based delay density (DF-DD).

The arrival curve represents the maximum number of events and the service curve represents the minimal computing power which is available within any time interval. Though, the actual time interval when the particular worst case occurs is not correlated. Nevertheless, or just therefore, we can assume the joint treatment to constitute an upper bound for the timing behaviour. Not only for the worst case response time r^+, but also for the response times of all other instances of a task within a time interval.

The time interval from zero up to the intersection point is called the busy-period.

However this is only valid for the delays before both curves intersect, e.g. before $\Delta = 136$ time units in Figure 2. From there on the horizontal distance between these curves no longer holds any useful information for our goal, e.g. in Figure 2 the fifth event occurs at the earliest within $\Delta = 150$ time units and the resources to process five events would be available at the latest within an interval

of $\Delta = 166$ time units. So the horizontal distance would be 16 time units. However, to process a single event with the given parameters for the TDMA scheme would require at least 32 time units. This discrepancy is due to the resources which have been assigned to this GPC in the time between $\Delta = 136$ time units and $\Delta = 150$ time units. These resources were lost since no events were pending to be processed. However, these resources were then assumed to have been used in processing the fifth event even though the resources were lost before the fifth event even occurred.

We need to determine the delays of the instances after the curves have intersected. With the available curves we can compute the remaining service curves, in particular the lower bound of the remaining service curve (1). We now have on the one hand the lower bound of the available service curve β^l denoting the amount of resources that were assigned at least to the GPC in any interval of length Δ. On the other hand we have the lower bound of the remaining service curve $\beta^{l'}$ denoting the amount of resources that remains at least in any interval of length Δ, or in other words the amount of resources that were not used in processing the events of the GPC. This allows us to determine the amount of consumed resources

Definition 8 *(Consumed Resources). The consumed resources σ by a GPC is specified by the difference of the lower service curve and the remaining lower service curve*

$$\sigma = \beta^l - \beta^{l'}$$

These curves are shown in Figure 2, where the lower service curve β^l is the dashed continuous function, the remaining lower service curve $\beta^{l'}$ is the dotted continuous function, and the consumed resources σ is the solid continuous function. From the consumed resources σ can be seen that at $\Delta = 186$ time units the resources were spent to have processed the fifth event. Therefore the delay for that instance is $186 - 150 = 36$ time units. So the horizontal distances between the upper arrival curve α^u and the consumed resources σ represents the delays of each of the instances even for those instances beyond the point where the curves originally intersected. Note that $\beta^l(\Delta)$ is identical to $\sigma(\Delta)$ for $\Delta \in [0, 136]$. Since the greatest horizontal distance between α^u and β^l occurs before they intersect, the greatest horizontal distance between α^u and β^l is the same as between α^u and σ, i.e. $(\alpha^u \leftrightarrow \beta^l) = (\alpha^u \leftrightarrow \sigma)$.

Instead of expressing the delays of the instance as horizontal distance between the two curves, we can also express it as the vertical distance of their pseudo inverse (Definition 9). So, the delays of the instances $\overline{r}(k)$ can be expressed as

$$\overline{r}(k) = (\sigma^{-1} - \alpha_i^{u-1})(k).$$

The functions used in real-time calculus have as requirement that they are increasing but not necessarily strictly monotonically increasing. Therefore these function might not be bijective, hence a corresponding inverse function might not exist. However a pseudo-inverse function can be defined instead.

Definition 9 *(Pseudo-Inverse). Let $f : \mathbb{R}_0^+ \to \mathbb{R}_0^+$ be a function, then its pseudo-inverse is defined as*

$$f^{-1}(y) := \inf_{x \in \mathbb{R}_0^+} \{x : y \leq f(x)\} \tag{12}$$

Note that $\overline{r}$ constitutes a delay function, hence we can derive a delay density as proposed in subsection 3.1. So we first obtain the cumulative delay function $\overline{\delta}(k) = \sum_{j=1}^{k} \overline{r}(j)$ (Definition 2) and then we obtain a delay density $\overline{dR^+} = (\overline{\delta} \oslash \overline{\delta})$, see (4).

Since the delay density $\overline{dR^+}$ is the self-deconvolution of the cumulative delay function $\overline{\delta}$, the delay density $\overline{dR^+}$ is sub-additive and $\overline{dR^+}(0) = 0$ holds. So, according to Proposition 3 any non-negative whole number $\Delta \in \mathbb{N}_0$ satisfies

$$\overline{dR^+}(\Delta) \leq \Delta \cdot \overline{dR^+}(1).$$

Proposition 3 *Let $f : \mathbb{R}_0^+ \to \mathbb{R}_0^+$ be an increasing function, then any non-negative integer $n \in \mathbb{N}_0$ satisfies*

$$(f \oslash f)(n) \leq n \cdot (f \oslash f)(1) \tag{13}$$

Furthermore the greatest horizontal distance between α^u and β^l is the same as between α^u and σ, as was noted earlier. Therefore it follows that $\overline{dR^+}(1) = (\alpha^u \leftrightarrow \beta^l) = r^+$. So overall we have

$$\overline{dR^+}(\Delta) \leq \Delta \cdot \overline{dR^+}(1) = \Delta \cdot r^+.$$

Above all, we have to take situations into account that the duration of the controller's runtime can be limited. Furthermore, the controller could be launched repeatedly after any unknown runtime. To take this into account, it is necessary to calculate $\overline{dR^+}$ for all possible runtimes and to compose the supremum from all $\overline{dR_i^+}$, where i denotes the number of instances of the control task during a particular runtime. This is sufficient, as the Real-Time Calculus considers the worst case of the interference of all tasks when the interval $\Delta = 1$. If the worst case response time r^+ will appear for $\Delta = 1$, the DF-DD will be equal to the DB-DD.
Hence this DF-DD is not greater than the DB-DD presented in the previous subsection 4.1. Moreover, we expect that the DF-DD noticeably improves the DB-DD for task sets where r^+ appears for one $\Delta > 1$. If a minimum runtime before re-launch can be assumed, DF-DD will also improve the DB-DD. To take this information into account, the algorithm has to ignore the dR_i^+ for time intervals shorter than the minimum runtime in line 14.

Given a trace containing n events, Algorithm 1 can be used to compute the DF-DD. For complete coverage of all possible situations, n should cover the hyper period of the task set.

4.3. Comparison

We now compare the DB-DD with the DF-DD by means of the example shown in Figure 2. According to the parameter from which the upper arrival curve is derived, the events occur at $\Delta \in \{0, 15, 30, 45, 150 \cdot i\}$ with $i \in \mathbb{N}$. The length of the interval at which k consecutive events are processed at the latest can be determined with $\beta^{l^{-1}}(k) = \left\lceil \frac{k \cdot c^+}{s} \right\rceil \cdot (c - s) + k \cdot c^+$ (cf. Equation (2.9) in [10]). Therefore the first few events are processed within an interval of length $(36, 68, 100, 136, \ldots)$. Both curves, the upper arrival curve α^u and the lower service β^l, intersect at $\Delta = 136$ time units. The greatest horizontal distance between these curves occurs at $\Delta = 45$ time units for the upper arrival curve and at $\Delta = 136$ time units for the lower service curve, resulting in a distance of $r^+ = 136 - 45 = 91$ time units. The resulting delay density is shown in Figure 3.

5. Conclusion and Future Work

Estimating the timing behavior is a key issue in the design process of networked control systems. For that, safe upper bounds for response times and latencies are essential to ensure the desired

Algorithm 1 Upper bound for the delay function based delay density

$\sigma \leftarrow (\beta^l - \beta^{l'})$ ▷ consumed resources
$\overline{r}(k) \leftarrow (\sigma^{-1} - \alpha_i^{u-1})(k)$ ▷ delays of each instance
$\overline{\delta}(k) \leftarrow \sum_{j=1}^{k} \overline{r}(j)$ ▷ cumulative delay function
for $i \leftarrow 1, \ldots, n$ **do** ▷ for any runtime interval
  for $\Delta \leftarrow 1, \ldots, i-1$ **do** ▷ $(\wedge, +)$-deconvolution
    $Max \leftarrow 0$
    for $j \leftarrow 1, \ldots, i - \Delta$ **do**
      if $(\overline{\delta}(j+\Delta) - \overline{\delta}(j) > Max)$ **then**
        $Max \leftarrow \overline{\delta}(j+\Delta) - \overline{\delta}(j)$
      end if
    end for
    $\overline{dR_i^+}(\Delta) \leftarrow Max$ ▷ delay density
  end for
  DF-DD = $\sup_{i \in \mathbb{N}} \left\{ dR_i^+ \right\}$
end for

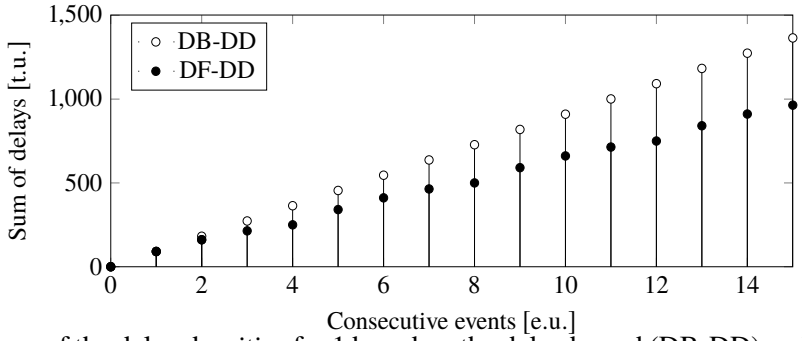

Figure 3: Comparison of the delay densities for 1 based on the delay bound (DB-DD) and the delay function (DF-DD).

behavior of the system. In this paper we have shown that it is possible to derive a safe upper bound which is tighter than the classical bound for periodical task activations with jitter. It even holds when the control system is stopped and launched again at any time. This bound is less pessimistic than assuming the worst-case response time occurs for every triggered event. For future work we intend to investigate methods for predicting stability of networked control systems based on delay densities rather than the worst case response time. (In [2], Ljapunov stability is assumed a priori.)

References

[1] Aminifar, A., P. Tabuada, P. Eles, and Z. Peng: *Self-triggered controllers and hard real-time guarantees*. In *2016 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 636–641, 2016.

[2] Bund, Tobias: *Verifikation sicherheitskritischer Regelsysteme unter Beachtung des Zeitverhaltens einer verteilten Rechenplattform*. PhD thesis, Universität Ulm, 2017.

[3] Bund, Tobias and Frank Slomka: *A delay density model for networked control systems*. Proceedings of the 21st International conference on Real-Time Networks and Systems - RTNS '13, page 205, 2013.

[4] Cullyer, J.: *Safety-critical control systems*. Computing & Control Engineering Journal, 2(5):202–210, 1991.

[5] Goswami, D., R. Schneider, and S. Chakraborty: *Co-design of cyber-physical systems via controllers with flexible delay constraints*. In *Proceedings of the 16th Asia and South Pacific Design Automation Conference*, pages 225–230. IEEE Press, 2011.

[6] Künzli, Simon, Arne Hamann, Rolf Ernst, and Lothar Thiele: *Combined approach to system level performance analysis of embedded systems*. In *Proceedings of the 5th IEEE/ACM international conference on Hardware/software codesign and system synthesis*, pages 63–68. ACM, 2007.

[7] Lincoln, Bo and Anton Cervin: *Jitterbug: A tool for analysis of real-time control performance*. In *Decision and Control, 2002, Proceedings of the 41st IEEE Conference on*, volume 2, pages 1319–1324. IEEE, 2002.

[8] Mirkin, Leonid: *On the extraction of dead-time controllers and estimators from delay-free parametrizations*. IEEE Transactions on Automatic Control, 48(4):543–553, 2003.

[9] Nilsson, Johan, Bo Bernhardsson, and Björn Wittenmark: *Stochastic analysis and control of real-time systems with random time delays*. Automatica, 34(1):57–64, 1998.

[10] Richter, Kai: *Compositional Scheduling Analysis Using Standard Event Models*. PhD thesis, Technical University Carolo-Wilhelmina of Braunschweig, 2005.

[11] Shen, Chijun: *Evaluierung einer oberen schranke für latenzdichten in verteilten,ereignisbasierten regelsystemen*. Master's thesis, Universität Ulm, 2017.

[12] Tolic, D. and S. Hirche: *Stabilizing transmission intervals for nonlinear delayed networked control systems*. IEEE Transactions on Automatic Control, 62(1):488–494, Jan 2017, ISSN 0018-9286.

[13] Wandeler, Ernesto: *Modular Performance Analysis and Interface-Based Design for Embedded Real-Time Systems*. PhD thesis, PhD Thesis ETH Zurich, 2006.

[14] Xu, Yang, Karl Erik Årzén, Enrico Bini, and Anton Cervin: *Lqg-based control and scheduling co-design*. IFAC-PapersOnLine, 50(1), 2017.

[15] Xu, Yang, Anton Cervin, and Karl Erik Årzén: *Harmonic scheduling and control co-design*. In *22nd IEEE International Conference on Embedded and Real-Time Computing Systems and Applications, RTCSA 2016*, 2016.