TBASCEM - Tight Bounds with Arrival and Service Curve Estimation by Measurements

Christoph Funda
ZF Mobility Solutions GmbH
Test System Development
Ingolstadt, Germany
christoph.funda@zf.com

Thomas Herpel
ZF Mobility Solutions GmbH
System Qualification & Product Test Strategy
Ingolstadt, Germany
thomas.herpel@zf.com

Reinhard German
Friedrich-Alexander-Universität Erlangen-Nürnberg
Computer Networks and Communication Systems
Erlangen, Germany
german@informatik.uni-erlangen.de

Kai-Steffen Jens Hielscher
Friedrich-Alexander-Universität Erlangen-Nürnberg
Computer Networks and Communication Systems
Erlangen, Germany
Kai-Steffen.Hielscher@informatik.uni-erlangen.de

ABSTRACT

This paper aims to solve the challenge of quantifying the performance of Hardware-in-the-Loop (HIL) computer systems used for data re-injection. The system can be represented as a multiple queue and server system that operates on a First-In, First-Out (FIFO) basis. The task at hand involves establishing tight bounds on end-to-end delay and system backlog. This is necessary to optimise buffer and pre-buffer time configurations. Network Calculus (NC) is chosen as the basic analytical framework to achieve this. In the literature, there are different techniques for estimating arrival and service curves from measurement data, which can be used for NC calculations. We have selected four of these methods to be applied to datasets of industrial Timestamp Logging (TL). The problem arises because these conventional methods often produce bounds that are much larger (by a factor of 1000 or more) than the measured maximum values, resulting in inefficient design of HIL system parameters and inefficient resource usage. The proposed approach, called TBASCEM, introduces a reverse engineering approach based on linear NC equations for estimating the parameters of arrival and service curves. By imposing constraints on the equation variables and employing non-linear optimization, TBASCEM searches for a burst parameter estimation which derives tight global delay bounds. In addition, TBASCEM simplifies the run-time measurement process, supporting real-time data acquisition to evaluate and optimise HIL system performance, and enhancing observability to adapt the HIL configuration to new sensor data. The benefits of TBASCEM are clearly that it enables an efficient performance logging of arrival and service curve parameters and with deriving tighter bounds in HIL systems, compared to evaluated state-of-the-art methods, making TBASCEM an invaluable tool for optimising and monitoring streaming applications in non-hard-real-time environments.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
ICPE '24, May 7–11, 2024, London, United Kingdom
© 2024 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 979-8-4007-0444-4/24/05…$15.00
https://doi.org/10.1145/3629526.3645031

CCS CONCEPTS

• **General and reference** → **General conference proceedings**; • **Computer systems organization** → **Embedded and cyber-physical systems**; • **Software and its engineering** → *Empirical software validation*; *Operational analysis*; • **Hardware** → *Sensors and actuators*; • **Information systems** → *Computing platforms.*

KEYWORDS

Performance Evaluation and Monitoring, Hardware-in-the-loop Test System, Streaming System, Network Calculus, Arrival and Service Curve Estimation

ACM Reference Format:
Christoph Funda, Thomas Herpel, Reinhard German, and Kai-Steffen Jens Hielscher. 2024. TBASCEM - Tight Bounds with Arrival and Service Curve Estimation by Measurements. In *Proceedings of the 15th ACM/SPEC International Conference on Performance Engineering (ICPE '24), May 7–11, 2024, London, United Kingdom.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3629526.3645031

1 INTRODUCTION

Autonomous vehicle technology requires millions of kilometres of recorded sensor data to be replayed with HIL systems to verify the perception technology on the Device Under Test (DUT). This process absorbs a significant amount of time and energy for the computing resources, as the recordings must be replayed in real-time from the HIL to the DUT. In addition, the replay must accurately reproduce a real-world scenario to ensure high quality HIL performance. It is essential that there is no data loss within the HIL system and that the timing to the DUT is accurate and precise, as needed by the DUT. We dive deeper into that topic in the Section 2. However, many HIL systems, beyond their hardware interfaces to the DUT, do not fall under the category of hard real-time systems. Instead, they fall into the realm of soft and non-hard real-time systems.

How can non-hard real-time computing machines meet the hard real-time requirements of the DUT? One approach involves utilizing a playback-buffer on the hard real-time capable interface-card. By filling the buffer with sufficient data before streaming to the DUT, uninterrupted streaming can be achieved.

The challenge lies now in designing the buffer size and pre-buffer time appropriately. To do so, it is crucial to determine the worst-case end-to-end delay and backlog bounds for buffer size dimensions and the pre-buffer time parameter.

Filling the playback buffer creates idle time before streaming to the DUT, leading to wasted time and energy of HIL computing resources. Hence, minimising pre-buffering time is crucial.

On the contrary, it is crucial to avoid an empty playback buffer during re-injection, as it disturbs the precise timing needed by mostly hard real-time DUTs. This hence leads to failed tests and again a loss of time and energy at the HIL computing resources. To achieve an efficient HIL performance, it is essential to adjust the pre-buffer time to be at least equal to or greater than the worst case end-to-end delay. The aim is to fill the playback buffer with a minimum of the burst parameter of the arrival curve of the sensor stream.

The methodology involves measuring the HIL system before operation, designing it accordingly, and monitoring its performance during operation to optimise its parameters if necessary. Using the measured maximum end-to-end delay directly would overestimate the required time, as waiting time in a queue could be included. Hence, it is necessary to consider queuing theory, which is accomplished by employing NC: a deterministic queuing system theory. Using NC, bounds for the backlog and end-to-end delay of a queuing system can be computed from its arrival and service curves. However, NC bounds should be tight, for our use-case and our system, at least the end-to-end delay bound needs to be as tight as possible, this is the major challenge.

There are basic concepts in the NC framework and mathematically defined in all basic literature about NC [13, 14]. In short, an arrival curve is the upper constraint of an input flow, and a service curve is the lower constraint of a flow provided by a service. The basic linear arrival curves and service curves, made used off in this work, are burst-rate curves and rate-latency curves. In hard real-time systems, their parameters can be found easily, they are often directly defined. However, in our non-real-time HIL system under study, they are not. So, they need to be measured, what is challenging. The analytical solutions for streaming devices with NC has been derived by Le Boudec et al. in [14]. We adapted and applied the NC solutions by linear equations on measurement data from a HIL test system in [11]. However, the results in [11] showed us, that there are improvements of the derived bounds needed for the estimation methods of the system service curves, to make them usable in practice.

One of the approaches mentioned in the survey study by Fidler et al. [8] to generate a strict service curve based on TL of the input and output of queuing server systems by Alcuri et al. [1] are used here in this paper. However, the results show, that the bounds are not tight at all with these methods. They often exceed by a factor of 1000.

Additionally, all the measurement methods, which are discussed in the survey, are based upon TL from the ingress and egress of the system. We are specifically interested to reduce the TL, to be able to do run-time measurements at the HIL system during operation, without generating massive data and performance overhead for the network or processor. On one hand, TL are occupying the memory. On the other hand, the TL need to be sent via network, what would occupy the processor and the network during operation.

To close the gap of estimating a service curve by measurements, with the aim to provide tight delay and backlog bounds of a streaming process within a computer system, without producing lots of TL, a new approach has been developed called TBASCEM — Tight Bounds with Arrival and Service Curve Estimation by Measurements. TBASCEM aims to provide a performant and efficient estimation method from TL as well as a technique for reducing the needed measurement data for arrival and service curve estimation, to be able to calculate tight bounds with NC.

In summary, our novel TBASCEM approach fills the gap for an effective and efficient method to measure and estimate service curves by measurements with the aim to calculate tight bounds, while maintaining low impact on CPU performance and memory allocation during operation.

The remainder of this paper is structured as follows: In Section 2 we start with basic technical backgrounds of HIL systems. In Sect. 3 we introduce our TBASCEM approach. We start with the description of the concept idea and the needed reduced measurements, followed by the reverse engineering approach for the estimation arrival and service curve parameters. In the following Section 4, we define our research questions, with a short explanation of the methodology on how to evaluate them and our hypothesis, followed by a description of the experimental setting, and finishing with discussing the results. In the following Section 5 we describe former work in literature and how it relates to our work. Finally, we end up with Section 6, where we conclude with a short summary and the contribution of our work. Our contributions include:

- Evaluating the tightness of delay and backlog bounds derived from four state-of-the-art service curve estimation methods by TL from the HIL test system with industrial workload, which highlights the need for TBASCEM, to analyse the performance of non-hard real-time streaming systems.
- Derivation of the TBASCEM approach by reverse-engineering of NC solution by linear arrival and service curves.
- Setting up of the optimization equation as well as the constraints for the TBASCEM algorithm.
- Implementing the TBASCEM Run-Time Measurement (RTM) in LabVIEW and the TBASCEM service curve generation in MATLAB for a proof-of-concept and performance evaluation.
- Quantitatively comparing the CPU performance of state-of-the-art TL with our TBASCEM RTM.
- Quantitatively comparing the bound tightness of four state-of-the-art approaches for estimating service curves from measurement with our TBASCEM estimation approach.

2 FUNDAMENTALS

HIL test systems have emerged as invaluable tools in computer science research and engineering. Their ability to seamlessly integrate virtual simulations with physical hardware empowers researchers and engineers to evaluate and optimize complex systems efficiently. With applications ranging from software testing to fault diagnosis, HIL test systems continue to shape the future of innovative technologies.

The following subsections of this section will confidently explore important aspects of HIL test systems, with a specific focus on real-time constraints and designing HIL test systems based on NC.

2.1 Real-Time Constraints of HIL Systems

In HIL systems, the processing time of software or network processes is heavily influenced by the computing machine on which the processes run. For hard real-time systems, the worst-case processing time serves as a critical performance indicator, imposing stringent requirements on the computing machine. Hard real-time systems are costly but offer formal evaluation and guarantees of their Worst-Case Execution Time (WCET) through measurements of processing cycles and CPU frequency.

However, many HIL systems, beyond their hardware interfaces to the DUT, do not fall under the category of hard real-time systems. Instead, they fall into the realm of soft and non-real-time systems. For such systems, formal evaluation cannot be done and guarantees for a WCET cannot be given due to various factors. Processing times are influenced by caching mechanisms, memory system hierarchies, CPU frequency fluctuations during run-time, hardware travel times, interrupt requests, context switches, and other features of modern computer and operating systems. Additional delays and processing times may arise when software systems use middleware, making accurate measurement difficult with high variation. For instance, interprocess communication using a localhost network connection in the robot operating system (ROS) [18] exemplifies this complexity, like applied in the HIL test system [10].

In these non-real-time systems, relying solely on an observed WCET can also be overly pessimistic, leading to bottleneck assumptions and infinite bounds in theory, while practical results suggest more leniency, as demonstrated by us in [11]. Despite their limitations, non-real-time systems are more cost-effective, and their software development is simpler and less expensive. For many cases, a service based on mean-rate or even stochastic bounds suffices. The occasional timing overshot does not result in catastrophic events if overshoots are monitored and documented in the test results.

In practice, closed-loop simulations running on HIL systems often employ monitoring mechanisms to detect simulation service performance during run-time. Any task overruns are logged as warnings or errors, prompting re-evaluation of tests in case of excessive, unwanted overruns [7].

In the case of open-loop re-injection, the HIL system functions as a streaming device. It streams measured input data like previously captured sensor data to the DUT. The DUT processes the data, and the output is streamed back to the HIL system for evaluation as test results. This setup allows for comprehensive testing and evaluation of complex cyber-physical systems, providing insights into their behaviour and performance in real-world scenarios.

2.2 Design of HIL Test Systems with NC

To provide bounds for end-to-end delay and buffer size design of queuing systems, NC is a possible analytical framework to work with. NC was introduced by Cruz et al. in 1991 [6]. The analytical solutions for streaming devices have been derived by Le Boudec et al. in [14]. It facilitates the establishment of strict yet secure buffer and delay bounds in these kinds of streaming systems.

We adapted and applied the NC solutions by linear equations on measurement data from a HIL test system in [11]. But as the findings in [11] demonstrated, there is a need to refine the derived boundaries in order to make the system service curve estimate techniques practical The derived bounds with the WCET method are often growing to infinity.

To fill this gap, we started a review study on practical measurement methods for arrival and service curves and applied them to the HIL system, published in [9]. The methods for estimating service curves by measurement data proposed by Helm et al. [12] and Wandeler et al. [20] were applied by us in [9] on TL from a HIL test system. We used realistic industrial workload for streaming data to CAN and Automotive Ethernet interfaces for re-injection to the DUT. The study showed that the delay and backlog bounds, generated by these estimated service curves are not tight enough in all cases, to be used in practice. Compared to the maximum measured backlog or end-to-end delay, the calculated bounds were a factor of over 10k higher.

The subsequent chapter of this paper introduces our new approach TBASCEM, which enables monitoring of the streaming performance like latency, rate, and backlog of the HIL systems during operation. It estimates arrival and service curves by using measurements which provide tight bounds for delay and backlog.

3 TBASCEM APPROACH

Our proposed concept aims to measure and estimate arrival and service curves for non-hard real-time software services in a streaming chain of a HIL system. This methodology can be applied to other streaming systems, including video streaming services, to accurately measure worst-case latency, backlog, and estimate burst parameters. Furthermore, it can estimate arrival and service curves from this data for tight NC bound calculations.

The approach relies on an iterative online algorithm derived by reverse engineering of linear NC bound equations. It replaces the conventional TL used for evaluating the processing performance of software modules. By doing so, the amount of logging data is significantly reduced. Instead of saving two timestamps per message, our method only requires saving three variables in total for one software process, based on timestamps and the number of messages in the queue. The iterative calculation process is computationally efficient, making it suitable for real-time operation of a HIL test bench in streaming mode.

The estimated service curves provide an overview of the HIL system's performance during operation, offering insights into any system influences and changes. By logging system data and storing these curves in a database, it is possible to calculate the likelihood of these service curves and analyse system affects more easily. This information may then be utilized with the stochastic NC framework.

Basically, the service curves play a crucial role in determining delay and backlog bounds with NC. If the HIL system needs to process new input data with a different arrival curve behaviour, the buffer size and pre-buffer time parameter will require adjustments. Our concept helps suggest an appropriate pre-buffer time to prevent buffer underflow at the playback buffer, and can also save time if the recommended pre-buffer time is significantly lower than the initially designed value. Furthermore, it allows for predicting the

required buffer size between each software service, with a defined safety factor, represented by the desired tightness factor.

3.1 Concept Idea of TBASCEM

The following Figure shows the queue and server system that can be abstracted by our streaming SW in the HIL system and the available measurement points.

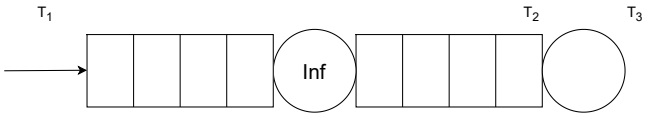

Figure 1: Queue and software service

During operation, our algorithm generates fundamental measurements by monitoring two key metrics: the maximum delay and maximum backlog between the input to a queue of a server (represented by T_1 in Figure 1) and the output of the server (represented by T_3 in Figure 1). To compute the burst parameter of the arrival curve, we utilize the given mean-rate from the original data and determine the vertical deviation from the mean-rate curve in each iteration, like illustrated in Figure 2a. We then store only the maximum positive and minimum negative values of this deviation.

The objective of the reverse engineering algorithm is to derive the rate-latency service curve parameters using the measured worst-case delay, backlog and burst parameter. The final reverse calculation from these three parameters to the service curve parameters are illustrated in Figure 2b. Furthermore, Figure 2c illustrates the reverse calculation in the negative burst domain.

The algorithm takes the following data as input for its calculations: for each packet i, the timestamp of the packet, t_i, and the number of bytes at each packet, b_i. During operation, we measure the maximum queue length, q_{max}, and the maximum delay, l_{max}. The algorithm utilizes the inter-arrival time Δt_i, which represents the time difference between two successive incoming packets at time T_i and time T_{i-1}.

$$\Delta t_i = T_i - T_{i-1} \tag{1}$$

The algorithm used for the arrival curve run-time measurement initially calculates the cumulative sum of inter-arrival time for each step i:

$$\sum_{i=0}^{i} \Delta t_i = \tilde{T}_i \tag{2}$$

Let $\tilde{T}_i$ represent the time passed until packet i, measured in seconds, and $\tilde{B}_i$ denote the total number of bits sent until packet i:

$$\sum_{i=0}^{i} b_i = \tilde{B}_i \tag{3}$$

The mean input rate, r_{in}, is calculated from the timestamp T_0 of the measurement data, which will be re-injected to the DUT.

Next, we iteratively compute the deviation from the ideal curve (see Figure 2a):

$$r_{in} \cdot \tilde{T}_i - \tilde{B}_i = \Delta b_i \tag{4}$$

Where Δb quantifies the deviation in terms of bits or messages.

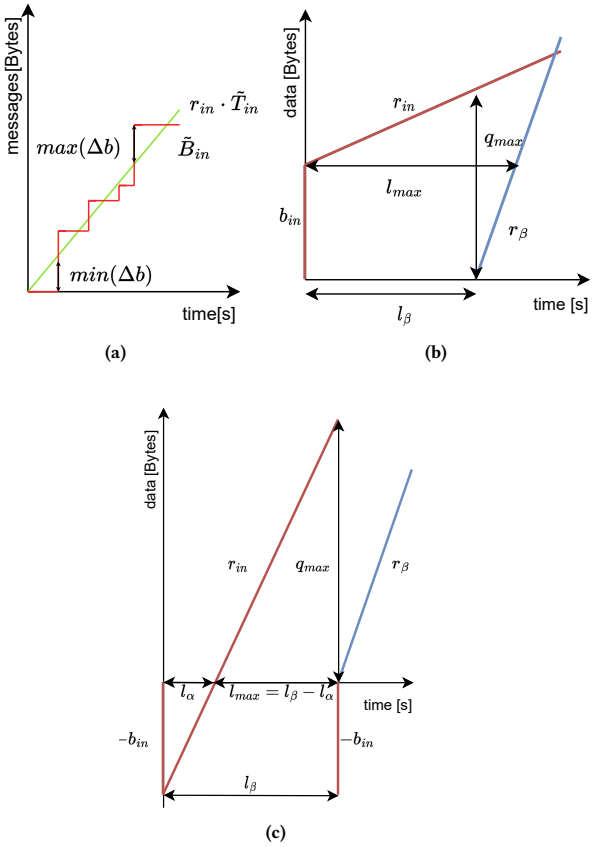

Figure 2: (a) Visualisation of burst parameter Δb estimation for the arrival curve; (b) l_β and r_β parameter estimation of the service curve ; (c) NC solution with negative burst

The algorithm determines the burst parameter in the arrival curve during operation as follows:

- Measure $min(\Delta b_i)$ and store it as b_{min} if the value is smaller than the previous saved value.
- Measure $max(\Delta b_i)$ and store it as b_{max} if the value is higher than the previous value.
- Calculate $b_{in_{pos}}$ as $(\Delta b_i) - b_{min}$ and update the maximum value of $b_{in_{pos}}$ if the calculated value is higher than the old value.
- Calculate $b_{in_{neg}}$ as $b_{max} - (\Delta b_i)$ and update the maximum value of $b_{in_{neg}}$ if the calculated value is higher than the old value.

The TBASCEM approach for estimating the service curve rate and latency parameters is based on the following measurement values:

- burst parameter b_{in} of the arrival curve at the service input, divided into a positive $b_{in_{pos}}$ and a negative $b_{in_{neg}}$ deviation from the mean rate

- burst parameter b_{out} of the arrival curve at the service output, divided into a positive $b_{out_{pos}}$ and a negative $b_{out_{neg}}$ deviation from the mean rate
- maximum latency l_{max} between service input and output
- maximum queue length q_{max} between service input and output, however, we determine the final maximum queue length as the maximum of q_{max} and $b_{out_{pos}}$. This approach is based on the NC theory, as the output bound is equal to the backlog bound.

3.2 Reverse Engineering Algorithm

The reverse engineering approach to calculate the service curve parameter rate r_β and latency l_β, involves the following equations, based on the basic geometric connections shown in Figure 2b.

For the calculation of the latency parameter l_β, the following equation can be used:

$$l_\beta = \frac{q_{max} - b_{in}}{r_{in}} \tag{5}$$

The parameter l_β is expressed in seconds.

Next, we obtain the rate parameter r_β using the following equation:

$$r_\beta = \frac{b_{in}}{l_{max} - l_\beta} \tag{6}$$

where the parameter r_β is given in units of $\left[\frac{Bytes}{s}\right]$ or $\left[\frac{bit}{s}\right]$ or $\left[\frac{msg}{s}\right]$.

However, this reverse engineering approach has two risks of incorrect parameter estimation: The first risk occurs when the numerator of Equation (5) becomes ≤ 0, resulting in $l_\beta \leq 0$. This can happen if the approximation of Δb from the input flow is overestimated. The second incorrect estimation can occur when the denominator of Equation (6) becomes smaller than 0, leading to r_β becoming negative. This can happen if l_β is higher than l_{max}, what can be caused by an underestimated b_{in}.

To handle these risks, we introduce the estimation parameter $\widetilde{b_{in}}$ and use it instead of the measured parameter b_{in}, to distinguish it from the measurements.

To determine a good value for $\widetilde{b_{in}}$, we furthermore analyse the tightness calculation of the two bounds, with d_{bound} as the delay bound and t_D as the delay bound tightness. Respectively, we analyse the backlog bound with q_{bound} and t_B as the backlog bound tightness. We establish the equations for calculating the tightness of the bounds, starting with the backlog bound:

$$q_{bound} = r_{in} \cdot \widetilde{l_\beta} + b_{in} \tag{7}$$

$$t_B = \frac{q_{bound}}{q_{max}} = \frac{r_{in} \cdot \frac{q_{max} - \widetilde{b_{in}}}{r_{in}} + b_{in}}{q_{max}} = \frac{q_{max} + b_{in} - \widetilde{b_{in}}}{q_{max}} \tag{8}$$

The tightness needs to be ≥ 1 to ensure that the bounds are not undershot. Therefore, an additional constraint is derived from Equation (8):

$$C3 : \widetilde{b_{in}} \leq b_{in} \tag{9}$$

The tightness formula for the delay bound is set up as follows:

$$d_{bound} = \frac{b_{in}}{\widetilde{r_\beta}} + \widetilde{l_\beta} \tag{10}$$

We calculate the tightness of the delay bound using the following equation:

$$t_D = \frac{d_{bound}}{l_{max}} = \frac{\frac{b_{in}}{\widetilde{b_{in}}} + \widetilde{l_\beta}}{l_{max}} = \frac{\frac{b_{in}}{\widetilde{b_{in}}} \cdot l_{max} + \frac{q_{max} - \widetilde{b_{in}}}{r_{in}} \cdot (1 - \frac{b_{in}}{\widetilde{b_{in}}})}{l_{max}} \tag{11}$$

To ensure that the delay bound is not undershot, the tightness needs to be ≥ 1, and from equation 11, we get the additional constraint:

$$t_D \geq 1 \Longrightarrow \frac{b_{in}}{\widetilde{b_{in}}} + \frac{q_{max} - \widetilde{b_{in}}}{r_{in} \cdot l_{max}} \cdot (1 - \frac{b_{in}}{\widetilde{b_{in}}}) \geq 1 \tag{12}$$

Tightness is important because loose bounds lead to inefficient HIL system design. Specifically, in the case of a backlog bound, it leads to a waste of memory resources, and in the case of a delay, it leads to a loss of time.

We derive the following two conditions from Equation (5) and (6) that must be handled separately to derive a service curve with valid bounds by the measurements.

3.2.1 Basic Conditions for Valid Bounds. The basic assumption is that there are no bottlenecks in the service. So, we set up the following inequality at first and derive the basic condition from it.

$$\begin{aligned} & & A1 : r_\beta \geq r_{in} \\ \Longrightarrow \frac{\widetilde{b_{in}}}{l_{max} - l_\beta} \geq r_{in} & \Longleftrightarrow \frac{\widetilde{b_{in}}}{lmax - \frac{q_{max} - \widetilde{b_{in}}}{r_{in}}} \geq r_{in} \\ & \Longleftrightarrow \widetilde{b_{in}} \geq r_{in} \cdot l_{max} + \widetilde{b_{in}} - q_{max} \\ & \Longrightarrow CD1 : \frac{q_{max}}{l_{max}} \geq r_{in} \end{aligned} \tag{13}$$

The assumption $A1$ and setting in Equation (6) leads to the condition $CD1$, what we prove at first. If the condition is not fulfilled, we assume another possible theoretical solution. If $l_\beta > l_{max}$ the inequality changes to the following term:

$$\begin{aligned} A1 : r_\beta \geq r_{in} \Longrightarrow \frac{\widetilde{b_{in}}}{l_{max} - l_\beta} \geq r_{in} \\ If : \ l_{max} - l_\beta \leq 0 \Longrightarrow \widetilde{b_{in}} \leq r_{in}(l_{max} - l_\beta) \\ \widetilde{b_{in}} \leq r_{in}(l_{max} - \frac{q_{max} - \widetilde{b_{in}}}{r_{in}}) \Leftrightarrow \widetilde{b_{in}} \leq rin \cdot l_{max} + \widetilde{b_{in}} - q_{max} \\ \Longrightarrow CD2 : \frac{q_{max}}{l_{max}} \leq r_{in} \end{aligned} \tag{14}$$

The visualisation of the theoretical concept for $CD2$ and the negative solution space of $\widetilde{b_{in}}$ is shown in Figure 2c.

We know that the arrival curve shown in Figure 2c is not a valid arrival curve according to the NC definition in the min-plus algebra, because it does not fulfil the subadditivity property. However, in the

max-plus algebra and in the Real-Time Calculus (RTC) framework, it is used. So, we also use it as an arithmetic solution to estimate the service curve parameters if the condition $CD1$ is not fulfilled and a possible solution can be found in the negative solution space. We later use the arrival curve with the positive burst parameter for the calculation of the bounds.

Also, the equation (14) for the service rate is not geometrically tractable, and cannot directly be applied to the solution. We can assume that there is no bottleneck and that the service rate corresponds to the input rate, and the input rate can be estimated by this formula. So, the inequality in Equation (14) is still valid.

However, estimating the $\widetilde{b}_{in}$ parameter may lead to overestimation, as the maximal and minimal deviation from the average rate, like $b_{in_{pos}}$ and $b_{in_{neg}}$ are measured, may not necessarily occur at subsequent arrivals, as shown in Figure 2a. The deviation could happen at entirely different epochs in the trace.

To find a suitable value for $\widetilde{b}_{in}$ that ensures a valid calculation of the service rate and latency parameter, it must be located within certain limits, not necessarily matching the measured value $b_{in_{pos}}$. We derive the constraints for these limits in the following sections.

3.2.2 Basic Constraints for Valid Bounds under Condition 1. To ensure a valid value for the latency l_β parameter, the following constraint needs to be met, based on Equation (5):

$$q_{max} - \widetilde{b}_{in} \geq 0 \Longrightarrow C1 : \widetilde{b}_{in} \leq q_{max} \tag{15}$$

To ensure a valid value for the rate parameter, the following constraint needs to be fulfilled, based on Equation (6):

$$l_{max} - l_\beta > 0 \; with (5) \Longrightarrow C2 : \widetilde{b}_{in} > q_{max} - l_{max} \cdot r_{in} \tag{16}$$

Since the burst parameter under Condition 1 should not be smaller than 0, a final constraint is added:

$$C4 : \widetilde{b}_{in} \geq 0 \tag{17}$$

3.2.3 Basic Constraints for Valid Bounds under Condition 2. Since the burst parameter under Condition 2 should be smaller than 0, a first constraint $C5$ is added:

$$C5 : \widetilde{b}_{in} \leq 0 \tag{18}$$

To ensure a valid value for the latency l_β parameter, the constraint needs to be met, based on Equation (5):

$$C6 : q_{max} - \widetilde{b}_{in} \geq 0 \Longrightarrow \widetilde{b}_{in} \leq q_{max} \tag{19}$$

To ensure a valid value for the rate parameter, the following constraint needs to be fulfilled, based on $CD2$ described in Equation (14):

$$l_{max} - l_\beta < 0 \; with \; (5) \implies C7 : \widetilde{b}_{in} < q_{max} - l_{max} \cdot r_{in} \tag{20}$$

The final constraint is the same as constraint 3 derived by Equation (8):

$$C8 : \widetilde{b}_{in} \leq b_{in} \tag{21}$$

3.2.4 The Optimization Function — Finding a Solution for the Arrival and Service Curve Parameter Estimation Problem. As a basis, we set up an optimization function which shall be minimized. It can be derived by finding the minimum of both summed tightness factors t_D and t_B:

$$\begin{aligned} t_B + t_D \\ = \frac{q_{max} + b_{in} - \widetilde{b}_{in}}{q_{max}} + \frac{b_{in}}{\widetilde{b}_{in}} + \frac{q_{max} - \widetilde{b}_{in}}{r_{in} \cdot l_{max}} \cdot (1 - \frac{b_{in}}{\widetilde{b}_{in}}) \\ = \underbrace{-(\frac{q_{max} + l_{max} r_{in}}{q_{max} l_{max} r_{in}}) \widetilde{b}_{in}}_{linear\ part} + ... \\ \underbrace{(b_{in} - \frac{b_{in} q_{max}}{r_{in} l_{max}}) \frac{1}{\widetilde{b}_{in}}}_{hyperbolic\ part} + ... \\ \underbrace{(1 + \frac{b_{in}}{q_{max}} + \frac{b_{in} + q_{max}}{r_{in} l_{max}})}_{constant\ part} \end{aligned} \tag{22}$$

The optimization function (22) consists of a linear, a hyperbolic, and a constant part, using $\widetilde{b}_{in}$ as the variable. We use the built-in MATLAB function *solve*, to solve it as an optimization problem fulfilling either constraint 1-4 and inserting $b_{in_{pos}}$ as b_{in} for condition 1, or constraint 5-8 and inserting $b_{in_{neg}}$ as b_{in} for condition 2. However, the minimum could also be derived by graphical analysis of the optimization function (22) within the constraints.

In the first step of our algorithm, we verify, if condition $CD1$ holds true. Using the constraints $C1$ to $C4$ and $b_{in_{pos}}$ as the parameter b_{in} in the optimization function, we can determine a suitable parameter for $\widetilde{b}_{in}$ to find proper service curve parameters.

If $CD1$ it is not satisfied, $CD2$ must hold true. So, we derive the service curve based on our measurements for the negative burst parameter $b_{in_{neg}}$. We insert $b_{in_{neg}}$ as the measured parameter b_{in} into the formula of the optimization function (22) and solve it. We always use the $b_{in_{pos}}$ for the burst parameter of the arrival curve, to derive a valid arrival curve fulfilling the subadditivity property.

4 EVALUATION AND RESULTS

In this chapter, we state three research questions (RQ1-RQ3), explain why the question is important to us, we explain the methodology we use to answer the research question, we state a hypothesis, explain our experimental setup, and discuss the results.

RQ1: How tight are the end-to-end delay and backlog bounds by the estimated arrival and service-curves derived by different estimation methods based on TL from real industrial workload?

There are various methods which can be found in literature for estimating service curves from TL. Our aim is to decrease the TL and to enable a RTM technique during operation. To this end, we have developed the TBASCEM method. However, prior to implementing the RTM in software, it is worth evaluating the tightness of the bounds we can generate with the TL already collected from the HIL system.

Methodology: To evaluate the tightness, we implement the method in MATLAB and apply it offline on HIL TL. We furthermore compare it to other methods from literature, applied on the same dataset.

Hypotheses: Our Hypothesis is that TBASCEM derives the tightest bounds compared to all other applied methods.

Experimental Setup: To assess the performance of various arrival and service curve algorithms, we employ a dataset comprising 81 instances of TL with each, 61116 data points divided into 9 meaningful subserver configurations. This leads to a set of 44 million data points in total, collected from our prototype HIL system. We used realistic industrial workload; emulating radar data send as Ethernet packets over a 100 Mbps channel to the DUT with an average rate of 7.222 Mbps and a maximum rate of 100 Mbps while using Ethernet packets of 1538 byte length. TBASCEM is implemented in MATLAB and is subsequently applied to the recorded timestamps originating from the software processes. We extract both, latency and backlog bound from the arrival curve yielded by a single RTM algorithm, as well as from the service curve generated by four other state-of-the art algorithms, namely Best-Case Execution Time (BCET), Mean-Case Execution Time (MCET), WCET and the approach by Alcuri et al.. Additionally, we conduct a comparison of the bounds' tightness by dividing the NC bound by the maximum measured values of queue length and end-to-end delay. We used software timestamps as a baseline for our performance evaluation of the HIL streaming system. Having in mind that these Software (SW) timestamps are not as precise at Hardware (HW) timestamps. However, as the system processing time works in the ms range and their precision lays in the μs range, they can be presumed as precise enough.

Results:

We evaluate the tightness of the TBASCEM and the other algorithms offline using TL. Subsequently, we compare the computed bounds of all methods as tightness factors by dividing them with the maximum measured delay or backlog. These comparisons are presented in Figure 4 as box plots, which displays the following information: the median, the lower and upper quartiles, any outliers (computed using the interquartile range), and the minimum and maximum values that are not outliers. The box plot or box chart is described by MathWorks™according to [16] as follows. The sample median is the line inside each box. The top and bottom edges of each box are the upper and lower quartiles, respectively, where the upper quartile corresponds to the 0.75 quantile and the lower quartile corresponds to the 0.25 quantile. The distance between the top and bottom edges is the Interquartile Range (IQR). Outliers are values that are more than 1.5 IQR away from the top or bottom of the box. The whiskers are lines that extend above and below each box. A whisker connects the upper quartile to the non outlier maximum and to the non outlier minimum respectively. A tightness factor of smaller than 1 is an undershot and hence rated as a violation of the bound by the maximum measured value. We refer the interested reader to the full description of MathWorks™defined box charts to this source [16]. A tightness factor between 1 and 10 is rated as high tightness. A tightness factor between 10 and 10^3 is rated as medium tightness. And a tightness factor over 10^3 we rate as low tightness. However, it depends on the absolute value at the end if the tightness factor can be used as a safety factor for the playback-buffer size or pre-buffer time. If the measured service delay is in the μs range and the maximum measured backlog is in the byte range, a safety factor of 10k or even 1000k would be still in practice feasible. However, it would lead to inefficient use of computing resources. It is a trade-off between system robustness and resource capabilities. Remarkably, the delay tightness of the TBASCEM algorithm is mostly 1 in our datasets with outliers at 1.1 what was our aim, to have a very tigtht delay bound. Compared to the boxplots of the other algorithms, the TBASCEM bounds are significantly tighter. It is followed by the Alcuri and MCET method. They can't be significantly distinguished, as their boxes overlap. However, the median is significantly different, with 25 for Alcuri and around 60 for the MCET method. The BCET follows on the third place and the WCET method derives the untightest results with 60k in median but seems to have almost no variation, however this mainly depends on the logarithmic scale of Figure 4. While diving into the numbers in Table 1, we see that its whiskers and its outliers range between 53k and 70k and its IQR is around 562. For the backlog tightness, the results look a bit different. The Medium tightness of TBASCEM is with 2 the lowest of all algorithms. However, as the boxes overlay with the Alcuri and MCET method, they are not significantly distinguishable. The median value of the Alcuri algorithm is 10. The Median of the MCET method lays significantly higher at 90. The BCET method lays with its median of 300 significantly higher as the three before mentioned methods, its boxes lay between 100 and 11k also significantly higher and its whiskers range between 11 and 30k what is also significantly higher. Its outliers are near to the upper whisker also at around 28k. The WCET reaches again the significantly untightest median bound with 25k, while its box lays between 14k and 31k and its whiskers range between 500 and 58k without any outliers. However, it highly depends on the system parameters and the absolute value if the bounds derived by the arrival and service curve values can be used in practice. If they cannot be used, the maximum measured end-to-end delay and backlog can be used instead. However, as the TBASCEM algorithm produces tight bounds in our datasets for the delay, we can use the results to design the pre-buffer delay of our system. Even with a backlog tightness up to 10k we also can use the approach to design our buffers, as the Random Access Memory (RAM) resources are in the Gigabyte range while the calculated absolute bounds would be still in the Megabyte range. However, the design in total would be still just medium efficient, but is comparable to the other state-of-the-art methods.

RQ2: How high is the impact on the CPU performance of the eased runtime measurement method for TBASCEM compared to TL?

Explanation: To monitor our HIL streaming system's performance while it is in use, the RTM must reduce the saved variables while utilizing a relatively small amount of additional computational power. Producing less logging data would mean to reduce the measurements and implement an online algorithm for the measurements based on the TBASCEM approach. The logging data would be highly reduced, however, what would that mean for the CPU performance, if that algorithm runs in parallel on the computing machine?

Methodology: Implementation of the runtime measurement algorithm for TBASCEM and run it without logging, with TL and with TBASCEM logging and log the CPU load, compare the measured CPU load in a box plot. If the boxes cover each other and

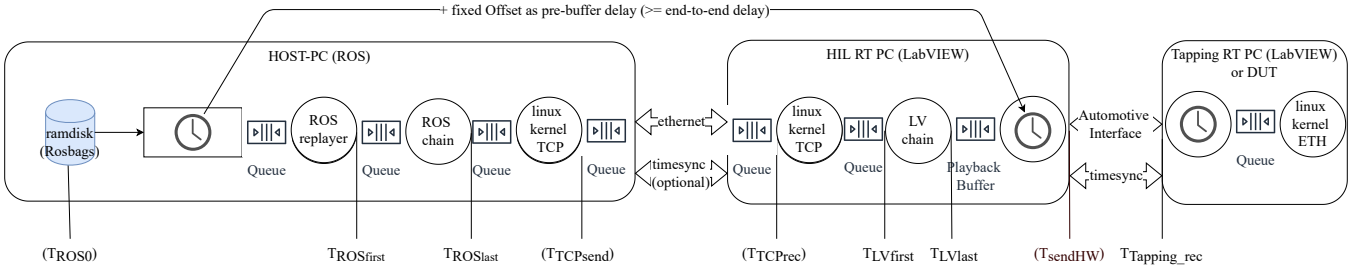

Figure 3: HIL streaming and queuing system with SW instrumentation to collect timestamps

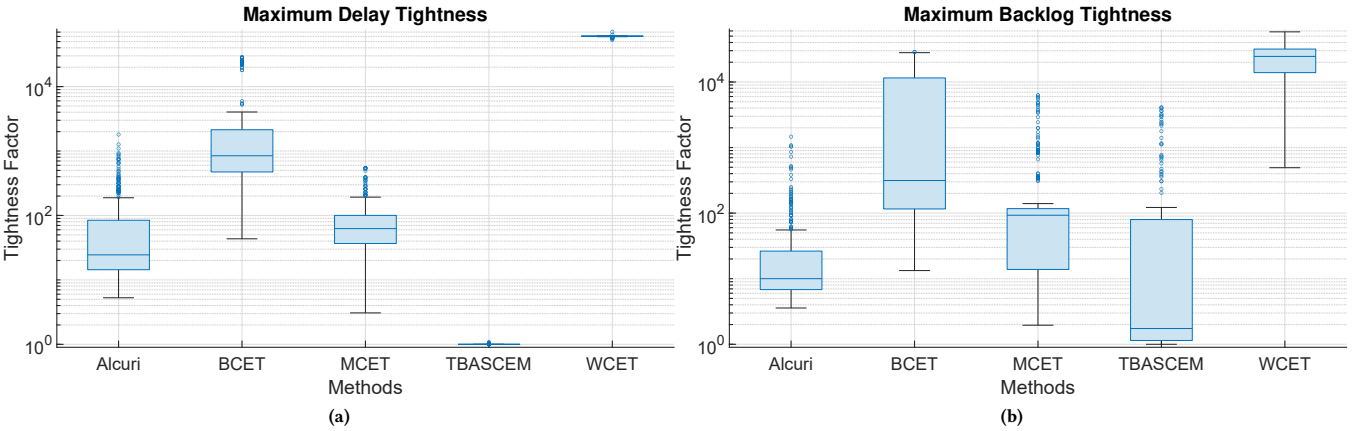

Figure 4: Performance evaluation: Maximum tightness of end-to-end delay bound (a) and backlog bound (b) over different servers and derived with different methods

Table 1: Summary Statistics for Delay Tightness

Method	Alcuri	BCET	MCET	TBASCEM	WCET
Upper Outlier	1802.2	28731.1	547.6	1.1	70817.8
Upper Whisker	188.3	4033.3	192.4	1.0	61265.0
0.75 Quantile	84.3	2139.2	100.3	1.0	60812.2
Median	24.5	844.0	62.5	1.0	60574.6
IQR	69.9	1665.7	63.5	0.0	562.2
0.25 Quantile	14.4	473.5	36.8	1.0	60250.0
Lower Whisker	5.3	43.4	3.1	1.0	59452.5
Lower Outlier	NaN	NaN	NaN	NaN	53263.4

Table 2: Summary Statistics for Backlog Tightness

Method	Alcuri	BCET	MCET	TBASCEM	WCET
Upper Outlier	1464.5	28706.2	6271.6	4112.0	NaN
Upper Whisker	55.4	27984.2	139.8	122.1	58111.9
0.75 Quantile	26.4	11524.5	117.1	79.8	31762.1
Median	10.0	314.3	92.9	1.7	24543.5
IQR	19.6	11408.6	103.4	78.7	17879.9
0.25 Quantile	6.8	115.9	13.8	1.1	13882.2
Lower Whisker	3.6	13.3	2.0	1.0	492.5
Lower Outlier	NaN	NaN	NaN	NaN	NaN

there is no gap between them, they are assumed as not significantly distinguishable.

Hypotheses: We postulated that the CPU performance demand for the run-time algorithm would be significantly lower than that of TL, given our observation that file writing places a substantial burden on CPU resources.

Experimental Setup: In order to validate this hypothesis, we executed a proof-of-concept implementation of the TBASCEM run-time measurement algorithm using LabVIEW. Subsequently, we carried out a comparative analysis, evaluating the LabVIEW-based TBASCEM implementation against TL, as well as a scenario without any logging. The assessment focused on parameters such as CPU utilization. CPU utilization measurements were taken every second using the Linux *top* command. This was chosen to provide granularity while avoiding overloading the system with monitoring measurements. We captured a snapshot of the CPU utilization of all cores as a percentage. We stream Ethernet data via 1,4,8,10,16 streams in parallel while logging via TBASCEM, TL and no logging while logging the CPU load every second via *top* command.

Results: The results of the measurements with different logging configurations, while scaling up the number of streams is shown in Figure 5. We see in the results, that the CPU utilization of TBASCEM, TL, and without logging differ not much per stream. They lay all in a comparable range, and the boxes overlap, so they are not

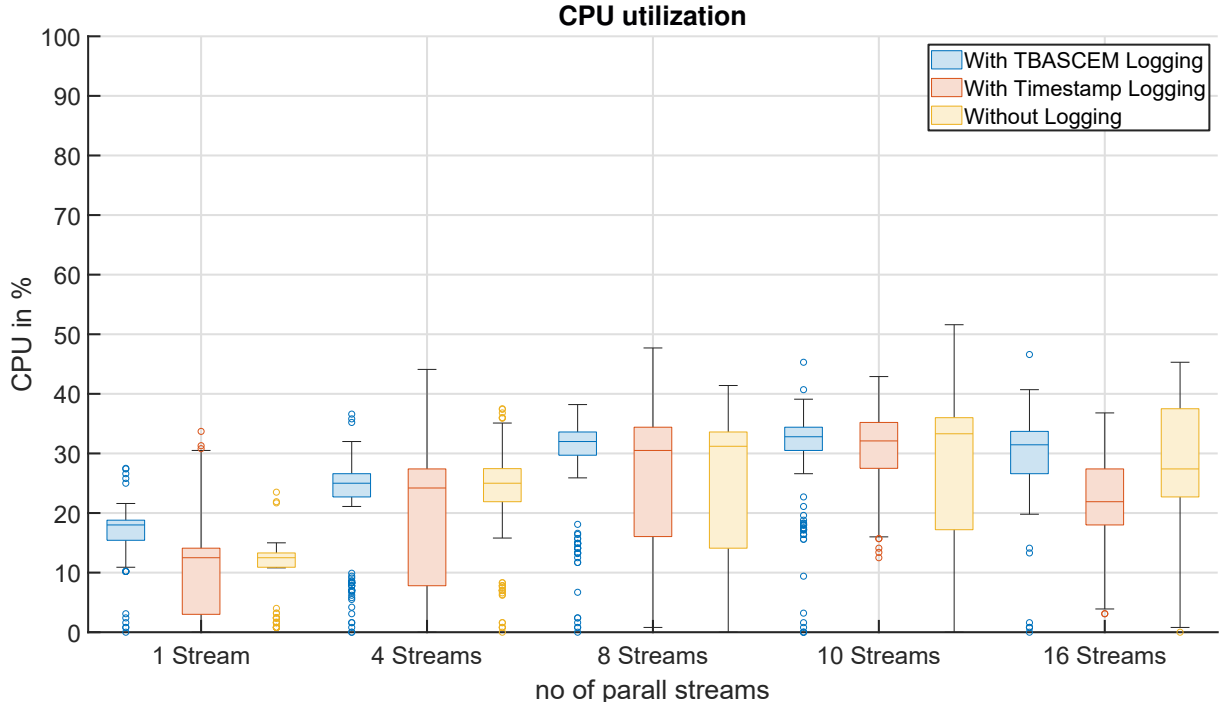

Figure 5: Measurement of CPU utilization of different logging methods

distinguishable. The measurements of 1, 4, 8, and 10 streams show consistent results for the higher peaks in CPU utilization of TL compared to TBASCEM logs. For 16 streams, the TBASCEM logging peak is higher. So, the TL and writing to a file, could indeed need more often high CPU resources. We don't see this always, like in the example of 16 streams, as the granularity of the CPU logs are too low. We could increase that with a higher frequency of *top* logs, but this would also increase the CPU load itself. However, the difference of the CPU load for 1 and 16 streams between the methods with TL and TBASCEM logging are just about 5%. So, the TBASCEM RTM approach may not need less performance, however it is comparable to the TL method and does not need much more CPU performance. Furthermore, the data size would be reduced, since the algorithm merely stores four values instead of the two TL per sample, a difference that becomes pronounced with a high number of samples and testing time. Table 3 displays the file sizes for both a TL at a single point and the TBASCEM log written to a CSV file.

Table 3: File sizes of streams with different duration with different logging methods.

	timestamp log in [kB]	**TBASCEM log in [kB]**
90s stream	1970	1
180s stream	3940	1
360s stream	7880	1
5000h stream	394.000.000	1

Considering the parallel streaming of up to 40 streams and the requirement for at least two points with timestamps to calculate delay and backlog, the overall file size can quickly increase up to several TB for a stream duration of approximately 5000h just for TL. Transferring the data from the HIL system to a different server for post-processing may be necessary, even though memory may not be an issue. This needs additional time, what could be used for the normal operation of the HIL system instead.

RQ3: How performant are the arrival- and service-curve estimation methods implemented as online algorithms compared to state-of-the-art TL?

Explanation: We want to compare the performance needs of the different methods for the memory and CPU, to rate their scalability and their tightness. These are the most relevant requirements for a run-time monitoring method.

Methodology: We assess the computational effort, memory usage, and scalability qualitatively by considering how the method operates and must be implemented as a runtime RTM. We describe the effort and provide a qualitative rating. The tightness of the bounds can be evaluated quantitatively by examining the results of the initial evaluation and comparison. The results have been categorised into qualitative values, with values under 10 considered high, those between 10 and 1000 considered medium, and values above 1000 considered low. This approach ensures consistency with other qualitatively rated factors. Additionally, we outline our requirements for the run-time monitoring approach.

Results: Various algorithms for estimating arrival and service curves using NC have been extensively discussed in the literature in recent years. The subsection offers valuable insights into the different approaches proposed by researchers for estimating arrival

curves and service curves, helping readers understand the diversity and nuances of the existing methods.

Service Curve Performance Estimation: When evaluating the suitability of the measurement method for normal run-time operation of a computer system, our primary focus is on achieving high performance, characterized by low computational effort and minimal memory usage, to ensure excellent scalability. A high level of tightness for the delay and a medium level of tightness for the backlog bound is considered sufficient for our specific objectives. Emphasizing efficiency and the ability to handle larger workloads is essential in practical applications, guiding our approach to ensure the method's practicality and effectiveness in real-world applications. The approach proposed by Alcuri et al. [1] follows an iterative method that analyses backlogged periods first for mean-rate estimation and then for maximum latency estimation. However, due to its high computational effort and memory usage, its scalability is rated as low, making it unsuitable as a run-time measurement algorithm. The WCET algorithm, proposed by Helm et al. [12] and applied to a HIL test system in Funda et al. [9] exhibits good performance but suffers from low tightness, particularly in non-hard real-time streaming systems where the mean service rate is close to the mean input rate. Any short-term WCET that results in a service rate lower than the input rate will create infinite bounds, limiting its practical application. The WCET method displayed in Figure 4 often yields no results due to an infinite bound, which is not illustrated.

The MCET algorithm, also proposed by Helm et al. [12] and detailed in Funda et al. [9] shows medium suitability from a performance perspective, with medium to low tightness. However, it requires saving a significant amount of data as all processing times are accumulated to derive rate and latency parameters.

The BCET algorithm, inspired by Wandeler et al. [20] and applied to a HIL test system by Funda et al. in [9] is unsuitable in terms of performance since it requires storing and sorting all measured processing latencies. Moreover, its tightness is insufficient, as it achieves medium to low tightness of the measured values as displayed in Figure 4. In contrast, the TBASCEM algorithm proposed in this paper has been evaluated to have medium computational requirements and low memory usage and high tightness for the delay and high to medium tightness for the backlog. The performance evaluation and tightness assessment of the TBASCEM algorithm are the primary focus of this paper and have been presented in the preceding chapters.

Arrival Curve Performance Estimation: The algorithm developed by Bouillard [5] is structured in multiple layers to detect more refined periodical behaviours. Initial data points that fall outside the specified bounds are identified as a sub-flow and are used as input for subsequent stages. From its description, we assess its computational effort to be of medium magnitude and its memory usage to be high, resulting in a medium level of scalability. However, this algorithm is not applicable in our context since we do not employ a traffic shaper like a token-bucket, which is a prerequisite for this algorithm. As a result, we cannot provide the required maximum burst parameter in advance; instead, we need to measure it during run-time.

On a different note, the iterative algorithm introduced in [9] lays a strong foundation for run-time implementation. With a foundational understanding of NC and basic measurements, this algorithm can be extended to estimate the minimum service curve.

The direct iterative approach discussed by Funda et al. in [9] estimates both the burst and mean-rate parameters with reduced computational effort. However, it necessitates the continuous storage of data over time, leading to a gradual increase in memory usage, which we rate as excessive. Nevertheless, it attains the highest level of tightness since it aligns with the definition of arrival curves. This algorithm serves as a valuable benchmark for comparison purposes.

In this paper, we operate under the assumption of low-performance requirements and an at least medium level of tightness for the run-time measurement demands. To validate these assumptions, we conducted a comprehensive experimental performance evaluation of the RTM for the reverse engineering TBASCEM algorithm. The TBASCEM algorithm proposed in this paper has been evaluated to have medium computational requirements and low memory usage and medium to high tightness.

5 RELATED WORK

Algorithms for estimating service curves using NC have been a common topic in the literature for the past years [1, 3–5, 8, 9, 12, 15, 20]. In this section, we present some examples of them, which we have chosen due to their applicability for our use-case of a FIFO queuing server system.

The work by Alcuri et al. [1] introduces a method to estimate service curves for various types of systems, including non-First-In-First-Out ones. The algorithm segments input and output traffic measurements into backlogged periods (periods when the buffer is not empty) and iteratively determines the start time, end time, and the amount of output traffic for each backlogged period. Throughput r of each period is computed as the bits leaving the system divided by the duration of the period. A maximum estimation technique finds the maximum throughput r_{max} among all backlogged periods. By tracing a line with a slope of r_{max} at all points of the departure process and projecting it onto the horizontal axis, the delay T is computed. Maximum delay T_{max} of each backlogged period is then obtained. The service curve is represented as a rate-latency curve with rate r_{max} and latency T_{max}[1]. This approach aligns with the definition of strict service curves as mentioned by Le Boudec et al. in [14].

The first method, proposed by Helm et al. [12], utilizes WCET to estimate the service curve. It takes the maximum measured processing latency and the minimum measured rate as the latency and rate for the service curve, respectively. While this approach performs well for hard real-time requirements, it requires that the minimum system service rate to be higher than the mean input rate. For systems involving streams of limited length and playback buffers, uninterrupted streaming is possible even if the mean system rate is not higher than the mean input rate.

To address this, the second method is based on the MCET and estimates the service using the measured mean rate. It computes the system latency by considering the maximum and minimum deviation between the input flow and the mean flow, effectively

accounting for latency outliers. This approach is inspired by Helm et al. [12].

The third method, based on BCET, orders the measured data with cumulative latencies in descending order to derive a service curve. A tangent at the rate-point of interest is traced, and the intersection point with the time-axis determines the system latency. However, this method may lead to highly overestimated bounds, particularly when patterns in the service flow exist. Additionally, it can be employed to estimate the arrival curve by sorting measured data in ascending order by the inter-arrival time. The intersection between the tangent and the y-axis (bytes) determines the burst parameter. This approach is inspired by the sliding window approach mentioned in the real-time calculus (RTC) framework by Wandeler et al. [20]. This method is particularly useful for assessing the feasibility of streaming a time-limited stream.

There are other measurement-based estimation methods for service curves mentioned in a literature review and survey study, that was conducted by Fidler et al. on various service curve models in the NC framework [8]. Especially in the section about "measurement-based service curve estimation" [8], are different methods mentioned, like the one from Alcuri et al. [1]. We have not re-implement all the methods and compared to our method. Especially the method by Undheim et al. [19] seek to estimate the latency and rate parameter during a burst and backlogged period like Alcuri et al. [1]. The main difference is that Undheim et al. use the max-plus algebra instead of the min-plus algebra. It could be, that the max-plus algebra-based estimation derives tighter bounds than the min-plus algebra-based solution used in [1]. This was already observed ones by Xie et al. in [21], while applying NC with min-plus algebra and max-plus algebra on priority scheduling for deriving delay bounds.

6 CONCLUSIONS

This paper introduces the TBASCEM methodology. The method delivers arrival and service curves which produce tight delay bounds and involves an efficient measurement process. It combines a RTM technique and a reverse engineering algorithm to estimate linear arrival and service curve parameters from measured data. However, it can also be applied to state-of-the-art TL from the input flow and output flow of any FIFO server queuing system. We applied the alternative methods on realistic data from the HIL system and received bounds with a tightness factor from several hundred up to several thousands. So, we found a gap for service curve estimation from measurement data, which produces tight bounds.

According to our empirical investigation, the reverse engineering algorithm TBASCEM provides tighter bounds compared to other methods in the literature. It can be applied on either TL of input and output flows of a server queuing system, or on reduced measurements we proposed here in this paper. Furthermore, we found out that the TBASCEM RTM adequately uses similar CPU performance like state-of-the-art TL, while it reduces limitations by data explosion associated with conventional TL, as confirmed by empirical evaluation of CPU utilisation and exemplary calculation of logging data sizes.

The TBASCEM RTM can be used in any streaming system to generate arrival and service curves without significantly impacting performance. It can be employed to evaluate performance, design buffer sizes and pre-buffer time, and continuously monitor performance during operation, to make service outliers visible.

Future work could involve comparing other measurement-based estimation methods for service curves mentioned in [8], that we did not implement and compare to our method. Additionally, our concept estimates the burst parameter. One could argue that it would be better to measure this parameter in a more appropriate manner. It is also critical to note that we assume all maximum parameters occur simultaneously, which is highly probable, but cannot be guaranteed. It may be more beneficial to measure the arrival curve burst parameter and the service curve rate and latency parameter directly online during operation from current values, saving only the maximum latency and minimum rate parameter for future reference. The TBASCEM optimization algorithm could also be adapted to find a solution for a given tightness factor for either the latency or the backlog within the solution space of the functional share of arrival and service curves. Future research could also include to extend the approach to the stochastic NC framework for formal validation of soft real-time streaming systems.

ACKNOWLEDGMENT

We gratefully acknowledge the invaluable contributions of: Lisa Maile (Friedrich-Alexander Universität Erlangen-Nürnberg), Dušan Okanović and Dhruv Jagga for insightful proofreading and feedback. Prof. Christian Berger (University of Gothenburg) for thought-provoking questions. Hampel Software Engineering GmbH for implementing the TBASCEM RTM in LabVIEW. Frank Keck (ZF Mobility Solutions GmbH) for unwavering support. This research was supported by ZF AG.

REFERENCES

[1] Luigi Alcuri, Giuseppe Barbera, and Giuseppe D'Acquisto. 2005. Service Curve Estimation by Measurement: An Input Output Analysis of a Softswitch Model. In *Quality of Service in Multiservice IP Networks*, Marco Ajmone Marsan, Giuseppe Bianchi, Marco Listanti, and Michela Meo (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 49–60.

[2] Bulgaria Aleksandrov, Chavdar Acad, Bulgaria Rumenin, Christian Magele, Stoyanov, Bulgaria Sotirova, Ritchie, Toepfer, Hartmut Brauer, Marin Hristov, Repetto, Bulgaria Antchev, Bulgaria Mihailov, Bulgaria Romansky, Bulgaria Vasilev, Japan Tanaka, Ventsislav Valchev, Vladimir Shelyagin, Ukraine Acad, and Anna Stoynova. 2019. Review of hardware-in-the-loop -a hundred years progress in the pseudo-real testing. 54 (12 2019), 70–84.

[3] A.A. Baybulatov and V.G. Promyslov. 2019. Control System Availability Assessment Via Maximum Delay Calculation. In *2019 International Conference on Industrial Engineering, Applications and Manufacturing (ICIEAM)*. 1–6. https://doi.org/10.1109/ICIEAM.2019.8743012

[4] A.A. Baybulatov and V. G. Promyslov. 2017. A Technique for Envelope Regression in Network Calculus. In *2017 IEEE 11th International Conference on Application of Information and Communication Technologies (AICT)*. 1–4. https://doi.org/10.1109/ICAICT.2017.8687034

[5] Anne Bouillard, Laurent Jouhet, and Eric Thierry. 2009. Service curves in Network Calculus: dos and don'ts.

[6] R.L. Cruz. 1991. A calculus for network delay. I. Network elements in isolation. *IEEE Transactions on Information Theory* 37, 1 (1991), 114–131. https://doi.org/10.1109/18.61109

[7] dSPACE GmbH. 2017. FAQ 242 - Handling Overrun Situations. https://www.dspace.com/shared/support/faqpdf/faq242.pdf

[8] Markus Fidler. 2010. Survey of deterministic and stochastic service curve models in the network calculus. *IEEE Communications Surveys & Tutorials* 12, 1 (2010), 59–86. https://doi.org/10.1109/SURV.2010.020110.00019

[9] Christoph Funda, Pablo Marín García, Reinhard German, and Kai-Steffen Hielscher. 2023. Arrival and Service Curve Measurement-Based Estimation Methods to Analyze and Design Soft Real-Time Streaming Systems with Network Calculus. In *2023 3rd International Conference on Electrical, Computer, Communications and Mechatronics Engineering (ICECCME)*. 1–8. https://doi.org/10.1109/ICECCME57830.2023.10253001

[10] Christoph Funda, Kai-Steffen Jens Hielscher, and Reinhard German. 2021. Discrete event simulation for the purpose of real-time performance evaluation of distributed hardware-in-the-loop simulators for autonomous driving vehicle validation. *Electron. Commun. Eur. Assoc. Softw. Sci. Technol.* 80 (2021).

[11] Christoph Funda, Tobias Konheiser, Thomas Herpel, Reinhard German, and Kai-Steffen Hielscher. 2022. An industrial case study for performance evaluation of hardware-in-the-loop simulators with a combination of network calculus and discrete-event simulation. In *2022 International Conference on Electrical, Computer, Communications and Mechatronics Engineering (ICECCME)*. 1–7. https://doi.org/10.1109/ICECCME55909.2022.9988051

[12] Max Helm, Henning Stubbe, Dominik Scholz, Benedikt Jaeger, Sebastian Gallenmüller, Nemanja Deric, Endri Goshi, Hasanin Harkous, Zikai Zhou, Wolfgang Kellerer, and Georg Carle. 2021. Application of Network Calculus Models on Programmable Device Behavior. In *2021 33rd International Teletraffic Congress (ITC-33)*. Avignon, France, 1–9. https://gitlab2.informatik.uni-wuerzburg.de/itc-conference/itc-conference-public/-/raw/master/itc33/hel21ITC33.pdf?inline=true

[13] Wolfgang Kellerer and Amaury Van Bemten. 2016. *Network Calculus: A Comprehensive Guide*. Technical Report 201603. Technische Universität München Lehrstuhl für Kommunikationsnetze, Arcisstr. 21, 80333 München, German.

[14] Jean-Yves Le Boudec and Patrick Thiran (Eds.). 2001. *Network Calculus*. Springer Berlin Heidelberg, Berlin, Heidelberg, 3–81. https://doi.org/10.1007/3-540-45318-0_1

[15] Ralf Lübben and Markus Fidler. 2017. Service Curve Estimation-Based Characterization and Evaluation of Closed-Loop Flow Control. *IEEE Transactions on Network and Service Management* 14, 1 (2017), 161–175. https://doi.org/10.1109/TNSM.2016.2638471

[16] MATHWORKS. [n. d.]. Box chart (box plot) - MATLAB boxchart - MathWorks Deutschland — de.mathworks.com. https://de.mathworks.com/help/matlab/ref/boxchart.html. [Accessed 22-02-2024].

[17] Franc Mihalič, Mitja Truntič, and Alenka Hren. 2022. Hardware-in-the-loop simulations: A historical overview of engineering challenges. *Electronics* 11, 15 (2022), 2462.

[18] Morgan Quigley, Ken Conley, Brian P. Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, and Andrew Y. Ng. 2009. ROS: an open-source Robot Operating System. In *ICRA Workshop on Open Source Software*.

[19] Astrid Undheim, Yuming Jiang, and Peder J. Emstad. 2007. Network Calculus Approach to Router Modeling with External Measurements. *2007 Second International Conference on Communications and Networking in China* (2007), 276–280. https://api.semanticscholar.org/CorpusID:6662110

[20] Ernesto Wandeler. 2006. *Modular Performance Analysis and Interface-Based Design for Embedded Real-Time Systems*. Ph. D. Dissertation. ETH Zurich.

[21] Jing Xie and Min Xie. 2013. Delay bound analysis in real-time networks with priority scheduling using network calculus. In *2013 IEEE International Conference on Communications (ICC)*. IEEE, 2469–2474.

A APPENDIX

A.1 Technical Information about the HIL Systems Under Study

The HIL cluster consists of a HOST PC and a HIL RT-PXI.

A.1.1 HOST PC. The HOST PC is equipped with a 2012 Intel Xeon E3-1230 V2 3.3 GHz processor (four physical CPU cores) and 16 GB system memory. All worker nodes are connected via 40 Gbit Ethernet in a single-switch star topology. Each node runs Gentoo Linux (kernel version 3.6.11) and Java 1.7.0.13.

A.1.2 HIL PC. The HIL RT-PXIe-8880 is equipped with an Intel(R) Xeon(R) CPU E5-2618L v3 @ 2.30GHz (8 physical CPU cores) and 24 GB system memory. All worker nodes are connected via 40 Gbps Ethernet in a single-switch star topology. Each node runs NI Linux Real-Time x64 4.14.146-rt67 and other NI LabVIEW Runtime and NI Drivers.

GLOSSARY

BCET Best-Case Execution Time (BCET) is the lowest observed execution time of a software process running on a dedicated computing machine.. 7, 10, 11

DUT Device Under Test (DUT) is the technical device, what is integrated into the hardware-in-the-loop simulator and stimulated by measurement data from the real-world device or by simulation.. 1–4, 7

FIFO First-In, First-Out (FIFO) refers to a principle where the first item to enter a system or queue is the first to be processed or served.. 1, 10, 11

HIL Hardware-in-the-Loop (HIL) test system or simulator or test bench is a methodology and a technical system for testing and validation of a technical product. See [2, 17] for details.. 1–12

HW Hardware (HW) are machines, wiring, and other physical components of a computer or other electronic system.. 7

IQR interquartile range (IQR) refers to the distance between the top and bottom edges of a boxplot, corresponding to the upper quartile or 0.75 quantile and the lower quartile or 0.25 quantile, respectively. 7

MCET The Mean-Case Execution Time (MCET) is the mean or average observed execution time of a software process running on a dedicated computing machine.. 7, 10

NC Network Calculus (NC) is a system theoretical approach for calculating delay and backlog bounds by min-plus algebra.. 1–5, 7, 9–11

RAM Random Access Memory (RAM) is a type of computer memory that stores data that can be searched by programs.. 7

RTM Run-Time Measurement (RTM) approach are measurement methods for performance evaluation of software during system operation. In this paper mainly used to measure the maximum end-to-end delay and backlog between the input and output flow of a message stream in a software service process with queues in between. Additionally, the maximum burst parameter of the input and output flow is estimated.. 2, 6, 7, 9–11

SW Software (SW) are programs and other operating information used by a computer.. 7

TL Timestamp Logging (TL) is a state-of-the-art method for performance evaluation of software timing. Timestamp Logs (TL) are the respective data-files generated by the logging. In this paper mainly used to measure the input and output flow of a message stream in a software service process with queuing.. 1–3, 6–9, 11

WCET Worst-Case Execution Time (WCET) is the highest observed execution time of a software process running on a dedicated computing machine.. 3, 7, 10

Received 17 November 2023; Accepted 29 December 2023; Revised 11 March 2024