Online Algorithm for Arrival & Service Curve Estimation

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

Pablo Marín García
Department Computer Science
University Erlangen-Nuremberg
Erlangen, Germany
pablo.marin@fau.de

Reinhard German
Department Computer Science
University Erlangen-Nuremberg
Erlangen, Germany
reinhard.german@fau.de

Kai-Steffen Hielscher
Department Computer Science
University Erlangen-Nuremberg
Erlangen, Germany
kai-steffen.hielscher@fau.de

***Abstract*—This paper presents a novel concept to extend state-of-the-art buffer monitoring with additional measures to estimate arrival- and service-curves. The online algorithm for arrival- and service-curve estimation replaces the state-of-the-art timestamp logging, as we expect it to overcome the main disadvantages of generating a huge amount of data and using a lot of CPU resources to store the data to a file during operation. We prove the accuracy of the online-algorithm offline with timestamp data and compare the derived bounds to the measured delay and backlog. We also do a proof-of-concept of the online-algorithm, implement it in LabVIEW and compare its performance to the timestamp logging by CPU load and data-size of the log-file. However, the implementation is still work-in-progress.**

***Index Terms*—hardware-in-the-loop streaming system, network calculus, service-curve estimation, performance monitoring**

I. Introduction

The concept we propose is an approach to estimate service-curves of each software service in a hardware-in-the-loop (HIL) streaming chain.

Our approach is based on an iterative online-algorithm inspired by network calculus (NC) that replaces the state-of-the-art timestamp logging to evaluate the processing performance of software modules. It reduces the amount of logging data enormously, as in case of one software process, three variables are saved. The variables are based on the timestamps and the number of messages in the queue. Due to their iterative calculation we assume low calculation efforts on a CPU. By that, it allows this approach to be used during operation of a HIL test bench in streaming operation mode.

The estimated service curves provide an overview of the performance of the HIL system during operation and give a hint on any system influences and changes. By saving them in a data-base together with logging data from the system, any system influences can be analysed.

Furthermore, the service curves can be used to calculate delay and backlog bounds with NC. If new input data with another arrival curve behavior needs to be processed by the HIL, the buffer size and pre-buffer delay parameter will change. The concept can be used to suggest a pre-buffer delay to prevent buffer underflow but also we can also safe time when the recommended pre-buffer delay is much lower than the designed one. It also allows to predict the needed queue size between each software service.

Supported by ZF Friedrichshafen AG.

II. Related Work

Algorithms for estimating arrival and service curves using NC have been a common topic in the literature for the past years. For a more detailed review of arrival and service curve estimation methods we refer to our paper [1]. In this section, we present a few examples.

A. Service Curve Estimation

The work from Alcuri et al. [2] presents a method for estimating service curves for all type of systems, including non-FIFO ones. To estimate the service curve, the algorithm first segments the given input and output traffic measurements into backlogged periods (periods of time where the buffer is not empty). Proceeding iteratively for each backlogged period, the start time, end time, and the amount of output traffic at these timestamps are determined. It then computes the throughput r of the backlogged period as the number of bits that left the system during the period divided by the duration of the period. After that, a maximum estimation technique is used to guess the maximum throughput of all backlogged periods, here denoted as r_{max}. If the desired precision of the estimation is reached, it continues, otherwise it restarts with the next backlogged period. By tracing a line with r_{max} slope at all points of the departure process and projecting it onto the horizontal axis, the delay T is computed. It then computes the maximum delay T_{max} of every available backlogged period. Finally, the final service curve is obtained as rate-latency curve with rate r_{max} and latency T_{max} [2].

Funda et al. [1] presented three estimation methods based on a literature review. The first one (based on WCET and proposed by Helm et al. [3]) takes the maximum measured processing latency and the minimum measured rate as latency and rate for the service curve, respectively. This method performs good for hard real-time requirements. However, this only works if the long-term rate of the system is lower than the mean rate, what was our main outcome of our case-study in [4]. The second method (based on MCET) estimates the service as the measured mean rate, as proposed by Helm et al. [3]. For the system latency, it computes the

maximum and the minimum deviation between the input flow and the mean flow. The system latency is computed as the difference between the maximum and minimum deviation. Latency outliers are then covered. The third method (based on BCET) resorts the measured data with the cumulative latencies in descendant order. Thus, a service curve can be derived. Then, a tangent at the rate-point of interested is traced and the intersection point with the time-axis determines the latency of the system. This third method can lead to highly overestimated bounds, especially when patters in the service flow exist or in the case of stochastic NC. This method can also be used to estimate the arrival curve: the measured data are sorted in ascending order by the inter-arrival time, This time, the intersection between the tangent and the y-axis (bytes) determines the burst parameter.

B. Arrival Curve Estimation

In our previous paper [1], we presented a review of estimation methods for arrival and service curves. In addition, we proposed a method for estimating the arrival curve, based on its definition. The upper curve was modeled as a rate-burst function and the lower curve as a rate-latency function. The rate r_α of both curves was computed using the arrival rate $1/dT_0$ of the measurements and averaging it:

$$r_\alpha = \frac{1}{mean(dT_0)} \tag{1}$$

It should be also noted that the arrival flow coming from the replayer has an arrival rate of $1/dT1$.

The burst parameter b_α of the upper curve was computed in a recursive way: in every step the burst parameter was increased by one and then shifted the origin of the arrival curve through the arrival flow. Any time there was an intersection point between both curves, the algorithm jumped to the next step. Once both parameters were computer, the arrival curve had the following form:

$$\alpha^u_{r,b} = r_\alpha \cdot t + b_\alpha \tag{2}$$

For the lower arrival curve, the latency L_α was computed as the maximum difference between the arrival flow and its mean rate. The rate is the same as for the upper arrival curve. The lower arrival curve had thus following form:

$$\alpha^l_{r,L} = r_\alpha \cdot \lceil t - L_\alpha \rceil \tag{3}$$

In [5], Bouillard presented a method for estimating the arrival curve that takes into account the changes in the input flow. This method should mitigate the effects of small variations of flows on the performance of the network. She computes and upper and lower arrival curve, which constrain the input flow x_n on the interval [m,n] if $\forall m \leq m' \leq n' \leq n$:

$$\underline{\alpha}(x_{n'} - x_{m'}) \leq n' - m' \leq \bar{\alpha}(x_{n'} - x_{m'}) \tag{4}$$

where $\underline{\alpha}$ and $\bar{\alpha}$ are the lower arrival curve and upper arrival curve, respectively, and $m < n$ are two non-negative integers. If eq. 4 is fulfilled, then the flow x_n is ($\underline{\alpha}$,$\bar{\alpha}$)-constrained on the interval [m,n]. The goal is to compute a long-term rate, which takes into account the drastic changes of flows. For that, she presents an iterative algorithm for obtaining the rate (both upper and lower arrival curve share this computed rate). Burst parameter b for the upper curve and latency parameter D for the lower curve are introduced manually and work as tuning parameters of the precision of the algorithm. The initial rate is $1/x_1$ and then in every step of the loop, eq. 4 is checked and in case it is not accomplished, a new rate is computed using the last point that was upper bounded and the point that does not fulfill anymore the lower bound, or conversely.

The algorithm is implemented in several layers for detecting better periodical behaviors. The first points that do not fulfill the bounds are saved as a sub-flow and given as input flow for the next layer.

III. Methods

A. Research Questions

How can we monitor the performance of our HIL streaming system during operation, without producing too much logging data and having a low impact on the CPU performance?

How accurate are the bound by the generated arrival- and service-curves compared to offline methods based on timestamps?

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

B. Performance Evaluation Approach

The performance evaluation approach is as follows:

- Analyse and rate the algorithm performance.
- Implement timestamp logging.
- Implement online algorithm.
- Operate the HIL with single and multiple streams with both implementations separately.
- Measure CPU load during operation and the size of the log-file at the end of operation with both implementations and compare them.
- Compare the accuracy of the algorithm with other algorithms and measurements of maximum delay and backlog.

C. Concept Idea and Description of the Online Algorithm

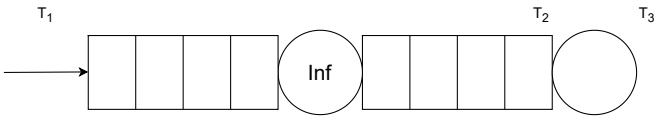


Fig. 1. Queue and software service

Our algorithm generates service curves during operation by monitoring two metrics: maximum delay and maximum backlog between the input to a queue of a server (represented by T_1 in Fig. 1) and the output of the server (represented by T_3 in Fig. 1) . We calculate the burst parameter of the arrival curve by the given mean-rate from the original data and derive the

vertical deviation from that mean-rate curve in each iteration, saving just the maximum and the minimum of that deviation.

In Fig. 2b, an illustration of the last two calculations can be seen: The online algorithm aims to extract the rate-latency service-curve parameters by using the measured worst-case delay as well as backlog. The following data are provided as input for the calculations: for each packet *i*, the time stamp of the packet, t_i, and the number of bytes at each packet, b_i. We measure the maximum queue length q_{max} and the maximum delay l_{max} during operation. In this algorithm, we use the inter-arrival time, which is the time difference between two successive incoming packets:

$$\Delta t_i = T_i - T_{i-1} \tag{5}$$

The online algorithm computes first, for each step *i*, the cumulative sum of inter-arrival time:

$$\sum_{i=0}^{i} \Delta t_i = \tilde{T}_i \tag{6}$$

where $\tilde{T}_i$ is the time passed until packet *i*, in seconds, and the bits:

$$\sum_{i=0}^{i} \Delta b_i = \tilde{B}_i \tag{7}$$

where $\tilde{B}_i$ is the number of Bits sent till packet *i*. The mean input rate r_{in} is calculated then from the timestamp T_0 of the measurement data, which will be re-injected to the DUT. A deviation from the ideal curve can be then iteratively calculated (see Fig. 2a):

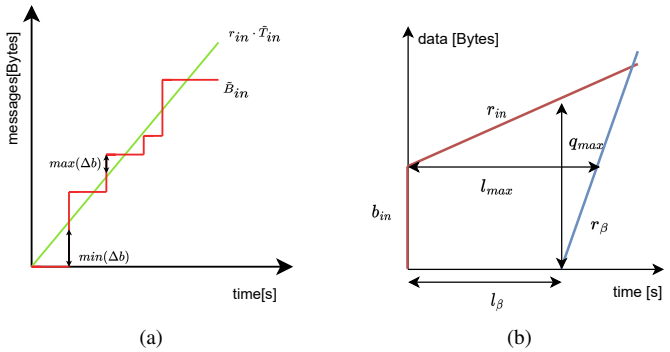
$$r_{in} \cdot \tilde{T}_i - \tilde{B}_i = \Delta b \tag{8}$$

where Δb measures the deviation, in bits or msgs.

Saving just the maximum and the minimum of Δb, we can calculate the maximum burst parameter b_{in}, of the arrival curve in bits or msgs, as:

$$b_{in} = max\{\Delta b, 0\} - min\{\Delta b, 0\} \tag{9}$$

where $max\{\cdot\}$ and $min\{\cdot\}$ calculate the maximum and minimum of their argument, respectively.



(a) (b)

Fig. 2. Estimation of Δb (a) and calculation of l_β and r_β (b)

For the calculation of the latency parameter l_β the following equation is used:

$$l_\beta = \frac{max\{q_{max} - b_{in}, 0\}}{r_{in}} \tag{10}$$

where the parameter l_β is in seconds. Lastly, we can obtain the rate parameter r_β as follows:

$$r_\beta = \frac{b_{in}}{max\{l_{max} - l_\beta, 0\}} \tag{11}$$

where the parameter r_β is in $[\frac{bit}{s}]$ or $[\frac{msg}{s}]$.

IV. Results of Algorithm Performance Evaluation and Discussion

A. Algorithm comparison and performance rating

Algorithms for estimating arrival and service curves using NC have been a common topic in the literature for the past years. In Table I and in II we present some of them, explaining their characteristics.

1) Service curve performance estimation: An algorithm is rated to be usable online, if the performance is high, which means the computational effort and the memory usage are both low, which leads to a high scalability. We rate scalability as more important than its accuracy. A medium accuracy is sufficient.

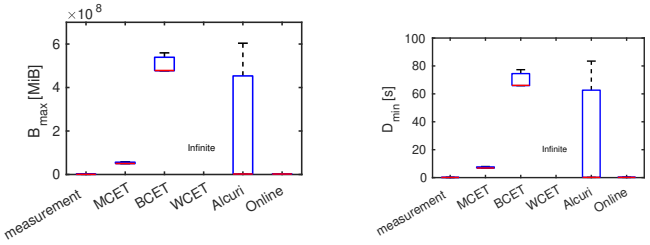
TABLE I
Comparison of Methods for Estimating Service Curves

Method	Computational effort	Memory usage	Suitability for online algorithm	accuracy of bounds
Alcuri et al. [2]	High	High	Low	High
Funda et al.- WCET [1]	Low	Low	High	Low
Funda et al.- MCET [1]	Medium	Medium	Medium	Medium
Funda et al.- BCET [1]	High	High	Low	Low
Online Algorithm	Low	Low	High	High
timestamp logging	Medium	High	Low	High

The approach of [2] is an iterative approach that analyses at first all backlogged periods for the mean-rate and afterwards for the maximum latency. So we rate its computational effort as well as its memory usage as high. Its scalability is rated as low and therefore it is not suitable to be implemented as an online algorithm. The WCET algorithm inspired by [3] and described in detail in [1] is suitable from a performance perspective, but its accuracy is too low and it leads to infinite bounds faster. The MCET algorithm inspired by [3] and described in detail in [1] rates medium suitable from a performance perspective, even if it has medium accuracy, what would be enough. The BCET algorithm of [1] is just not suitable from a performance perspective and has too low accuracy. For the online algorithm we describe here in this paper, we estimate low performance requirements and a high accuracy. The validation of its performance and its accuracy are the purpose of this paper.

TABLE II
COMPARISON OF METHODS FOR ESTIMATING ARRIVAL CURVES

Method	Computational effort	Memory usage	Suitability for online algorithm	Precission of bounds
Bouilliard [5]	Medium	High	Medium	High
Funda et al. [1]	Low	High	Medium	Medium
Online Algorithm	Low-Medium	Low-Medium	High	Medium
timestamp logging	Medium	High	Low	High



(a) (b)

Fig. 3. Performance evaluation: Maximum queue length (backlog) (a) and end-to-end delay (b)

2) Arrival curve performance estimation: The algorithm of Bouilliard is implemented in several layers for detecting better periodical behaviors. The first points that do not fulfill the bounds are saved as a sub-flow and given as input flow for the next layer.

We rate its computational effort from its description as medium and its memory usage as high, so its scalability is medium.

However, the algorithm can not be used in our case, as we do not use a traffic shaper like a token-bucket and therefore, we cannot give the maximum burst parameter before-hand, what is needed by that algorithm. We need to estimate the burst parameter online.

On the other hand, the iterative algorithm presented in [1] provides a good basis for implementing it online. With some basic knowledge of NC and some simple measurements, this algorithm could be extended to estimate the minimum service-curve.

The direct iterative approach mentioned by Funda et al. in [1] estimates the burst and mean-rate parameter and would need less computational effort, however it needs to save all the data over time, so its memory usage increase over time and is rated as too high. On the other hand, it achieves the highest accuracy, as it implements the definition of arrival-curves and can be used as an algorithm for comparison.

In this paper, we assume low performance requirements and a medium accuracy for the online algorithm, since there is a risk of overestimating the burst parameter, as we will discuss later. We also verify the validity of these assumptions by performing an experimental performance evaluation of the online algorithm.

B. Algorithm Accuracy

We prove the accuracy of that algorithm offline with timestamp data and compare the derived bounds to the measured delay and backlog and to other algorithms in table III and in Fig. 3. The derived bounds by the online algorithm are

TABLE III
TIGHTNESS OF NC METHODS (USING MAX. VALUES)

Method	D_{min}[s]	B_{max}[MiB]	D_{min} Tightness	B_{max} Tightness
Measurement	0.15	390256	-	-
Funda- MCET	7.95	5.84×10^7	51.51	41.99
Funda- BCET	77.33	5.58×10^8	501.14	402.39
Funda- WCET	Inf	Inf	Inf	Inf
Alcuri et al.	83.47	6.04×10^8	540.99	434.32
Funda - Online Algorithm	0.19	2382992	1.26	1.71

the tightest ones compared to the other algorithms. However, the algorithm has also two risks of an incorrect parameter estimation. The first one occurs, if the numerator of eq. 10 becomes 0, so l_β becomes also 0. This can happen, if the approximation of Δb from the input flow is overestimated. This is due to the fact that we do not consider the order of occurrence of the minimum and maximum of Δb. However, this order plays a role in the estimation of the arrival curve burst parameter. So there is a risk in our approximation of overestimating the arrival curve burst parameter. The second incorrect estimation can occur, if the denominator of eq. 11 becomes 0 so r_β becomes infinite. This can happen if l_β is higher than l_{max}. It is really unlikely that both parameters are incorrectly estimated at the same time. However, l_β could theoretically be higher than the measured l_{max}.

C. Algorithm Performance

We will do a proof-of-concept of the online algorithm in LabVIEW and compare its performance to the timestamp logging by CPU load and data-size of the log-file. However, this is for the moment work-in-progress. We furthermore assume, that the CPU performance requirement is much lower for the online algorithm compared to the timestamp logging, as we observed the writing to a file as highly consuming for the CPU performance. The data-size will be lower, as the algorithm just saves 4 numbers instead of 2 timestamp logs per sample, assuming thousands of samples.

V. CONCLUSIONS

In this paper, we presented an online algorithm that offers a light-weight method for tight service curves estimations. We carried out a short review of the different estimation methods for arrival and service curves from the literature and we evaluated its suitability for being applied online. Furthermore, we introduced the online algorithm and explained how it estimates the maximum delay and backlog during operation. We rated and compared the different algorithm performance based on an analysis. Lastly, we proved the accuracy of the online algorithm offline using timestamp data. We observed that the online algorithm provides the tightest bounds compared to the measured bounds. In future work, we will implement the online algorithm in LabVIEW and compare its performance to the timestamp logging method.

ACKNOWLEDGMENT

This research was supported by ZF AG and ZF Mobility Solutions GmbH (a company of ZF group).

References

[1] C. Funda, P. Marin Garcia, R. German, and K.-S. Hielscher, "Arrival and service curve measurement-based estimation methods to analyze and design soft real-time streaming systems with network calculus," 2023, unpublished.

[2] L. Alcuri, G. Barbera, and G. D'Acquisto, "Service curve estimation by measurement: An input output analysis of a softswitch model," in *Quality of Service in Multiservice IP Networks*, M. Ajmone Marsan, G. Bianchi, M. Listanti, and M. Meo, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 49–60.

[3] M. Helm, H. Stubbe, D. Scholz, B. Jaeger, S. Gallenmüller, N. Deric, E. Goshi, H. Harkous, Z. Zhou, W. Kellerer, and G. Carle, "Application of network calculus models on programmable device behavior," in *2021 33rd International Teletraffic Congress (ITC-33)*, Avignon, France, Aug. 2021, pp. 1–9. [Online]. Available: https://gitlab2.informatik.uni-wuerzburg.de/itc-conference/itc-conference-public/-/raw/master/itc33/hel21ITC33.pdf?inline=true

[4] C. Funda, T. Konheiser, T. Herpel, R. German, and K.-S. Hielscher, "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)*, 2022, pp. 1–7.

[5] A. Bouillard, "Algorithms and efficiency of Network calculus," Habilitation à diriger des recherches, Ecole Normale Supérieure (Paris), Apr. 2014. [Online]. Available: https://hal.inria.fr/tel-01107384