# How to Model and Predict the Scalability of a Hardware-In-The-Loop Test Bench for Data Re-Injection?

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

Tobias Konheiser
*Test System Development*
*ZF Mobility Solutions GmbH*
Ingolstadt, Germany
tobias.konheiser@zf.com

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 describes a novel application of an empirical network calculus model based on measurements of a hardware-in-the-loop (HIL) test system. The aim is to predict the performance of a HIL test bench for open-loop re-injection in the context of scalability. HIL test benches are distributed computer systems including software, hardware, and networking devices. They are used to validate complex technical systems, but have not yet been system under study themselves. Our approach is to use measurements from the HIL system to create an empirical model for arrival and service curves. We predict the performance and design the previously unknown parameters of the HIL simulator with network calculus (NC), namely the buffer sizes and the minimum needed pre-buffer time for the playback buffer. We furthermore show, that it is possible to estimate the CPU load from arrival and service-curves based on the utilization theorem, and hence estimate the scalability of the HIL system in the context of the number of sensor streams.**

***Index Terms*—hardware-in-the-loop simulation, computer performance evaluation, network calculus, scalability evaluation**

## I. Introduction

The HIL-based testing approach enables testing and validation of complex technical systems. Since the device under test (DUT) works under hard real-time (RT) conditions, the HIL test bench needs to be RT capable to enable the testing and to be able to check the RT properties of the DUT. To reduce the real-time requirements to soft real-time for the streaming software (SW), a well-known technique is used: the playback buffer. The playback buffer and the corresponding pre-buffer time need to be dimensioned appropriately, so that no overflow or underflow occurs. HIL systems need to be verified and validated during their development process. The formal basics for analyzing streaming systems with NC are proposed by Le Boudec in [8]. In our last paper [7] we applied the well-known methods of NC from the field of networking and performance engineering to a HIL simulator. We researched for and experimented with more appropriate service-curve approximation methods, taking the soft real-time requirements of the system under study into account. We have shown these methods in the Workshop for NC (WoNeCa) [6] and have written a pre-print paper [5] about it. With this work we extend the single stream system to a system with multiple parallel streams, describing a more complex system under study. We evaluate the performance in terms of scalability by increasing the number of parallel streams of the system. This kind of system is common in the area of autonomous driving systems, with a huge sensor set generating multiple parallel streams. The basics of scheduling multiple flows is well known in the NC community and can be described by first-in-first-out (FIFO) residual curves. We furthermore approximate the CPU utilization by setting the measured mean processing rates in relation to the mean rates of the input flow and compare it to the measured average CPU load on the HIL system. We create a CPU load model for the HIL system to make predictions. The predicted CPU load is validated using measurements with different number of streams.

## II. Methods

### A. Terminology

NC is the mathematical framework for deterministic delay and backlog bound calculation based upon the min-plus algebra invented by Cruz in 1991 [2]. We use the term cycle-time to describe a fixed time-interval between two successive messages or packets. The pre-buffer time or pre-buffer delay is the time that is used to fill the playback buffer in the HIL PC before starting to stream in real-time to the DUT.

### B. Arrival and Service-Curve Approximation

Arrival and service curves can be approximated with linear curves. The most common linear curve functions are burst-rate, rate-latency, delay and rate functions. These linear functions are basic functions and can be found in any basic literature about NC, like in [8], [2], [3] and [10]. Another approach are piecewise continuous functions, as described in [9], [11], [1] or [4]. As we found that the disadvantages of using piecewise continuous functions exceeds the benefits, we focus in this work on the approximation method for lower service curves and lower arrival curves with rate-latency functions described by the formula 1:

$$\beta_{r,L}(t) = r_\beta \cdot \lceil t - L_\beta \rceil \tag{1}$$

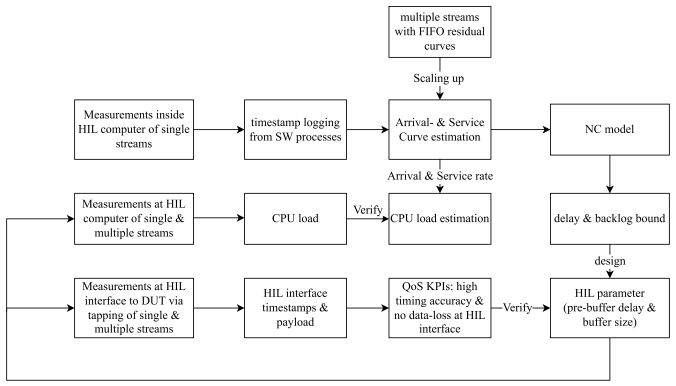
multiple streams with FIFO residual curves
Scaling up
Measurements inside HIL computer of single streams
timestamp logging from SW processes
Arrival- & Service Curve estimation
NC model
Arrival & Service rate
Measurements at HIL computer of single & multiple streams
CPU load
Verify
CPU load estimation
delay & backlog bound
design
Measurements at HIL interface to DUT via tapping of single & multiple streams
HIL interface timestamps & payload
QoS KPIs: high timing accuracy & no data-loss at HIL interface
Verify
HIL parameter (pre-buffer delay & buffer size)

Fig. 1. Computer performance evaluation models and how they are combined

In addition we use rate-burst functions for upper arrival curves described by the formula 2:

$$\alpha_{r,b}(t) = r_\alpha \cdot t + b_\alpha \tag{2}$$

The previous equations contain different parameters. $r_\beta$ and $r_\alpha$ describe the rate of the NC curve, $b_\alpha$ represents the burst parameter of the arrival curve and $L_\beta$ the latency value of the service curve.

## C. Research Questions

We derive the following research questions to estimate the performance of the HIL in terms of scalability of the streams.

- How tight are the bounds derived by NC for the maximum delay and the backlog in the system while scaling up the streams?
- Are the derived mean arrival & service-curve rates correlated to the mean CPU load?

To answer the questions we conducted the following approach.

## III. Approach

In Fig. 1, the performance evaluation methods and how they are combined and used in this study are shown. Measurements of the HIL system are the basis of this study. The measurements of timestamps at the input and output of processing servers at the HIL system are used to approximate arrival and service curves for NC equations. With NC solutions for streaming systems, the maximum delay bound (equal to the minimum pre-buffer time) as well as the maximum backlog bound are calculated. The delay and buffer bounds derived by NC are verified by timing measurements at the HIL interface to the DUT. Measurement of CPU utilization are used to verify the estimated arrival and service-curve rates. When the number of streams are scaled up on the HIL, the theoretical CPU utilization of the NC model is compared to the measured CPU load of the HIL device for verification purposes. For performance predictions an easy CPU model is generated. The aim is to predict the bottleneck of the CPU with the CPU model and the bottleneck of the memory with the NC model, while scaling up the streams.

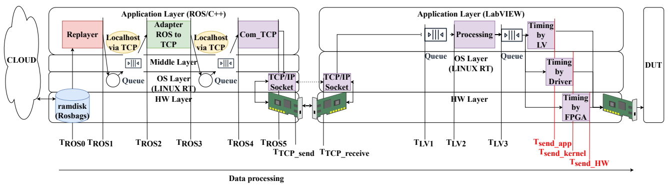

Fig. 2. Detailed conceptual model and software instrumentation

## A. Measurements

As a measurement system, a pure SW logger is used. In Fig. 2 the system under study is shown. It consists of a HOST PC where the SW is robot operating system (ROS) based and a HIL PC where the SW is LabView (LV) based, connected with a TCP connection. The SW is instrumented on both PCs in the user space to measure the SW processing latencies in µs. When data arrives and when it leaves a process, a system timestamps is generated for each step in the streaming chain. This kind of measurement for the service curve generation is called passive measurement in the NC community, according to the terminology used by Fidler in [4]. Additionally, we monitor the CPU load every second with the Linux top command.

## B. Network Calculus for Multiple-Stream Systems

The system under study consists of a local storage, a replayer, which replays the data, and the subsequent streaming chain. The SW processes in the streaming chain can be reduced to one system service. At the end of the streaming chain, the packet flow fills the playback buffer. After that buffer the packets flow with a time shift to the DUT. An analysis of streaming systems regarding playback buffer dimensioning and pre-allocation with NC was done by Le Boudec in [8]. For basic mathematical notation, we refer to his work and we refer to our last paper [7] for a mathematical proof and detailed mathematical notations. We furthermore model the scheduling on the PCs and the Ethernet network link with FIFO residual curves. We assume a fair distribution of the TCP streams i,j on the link:

$$\beta_i(t) = C \cdot \lceil t - L \rceil - \sum_{\forall j \neq i} \alpha_j(t) \tag{3}$$

We are limiting the residual service of the Ethernet link with 40 Gbps bandwidth to the maximum service offered by the TCP algorithm, which is estimated by measurements as 5,24 Gbps, denoted as $C$ in 3. We furthermore assume no data-loss on the Ethernet link. For the parallel running SW processes of the streaming chain, we assume a fair distribution of the SW processes of different streams scheduled on the CPU cores, described by the streams per core equation 4:

$$spc = \lceil \frac{streams}{cores} \rceil \tag{4}$$

We model the residual service of each processing unit according to equation 5:

$$\beta_i(t) = \begin{cases} \bar{r}_\beta \cdot \lceil t - L_\beta \rceil & spc \leq 1 \\ \bar{r}_\beta \cdot \lceil t - L_\beta \rceil - \sum\limits_{\forall j \neq i}^{spc} \alpha_j(t) & spc > 1 \end{cases} \quad (5)$$

The calculation of $D_{max}$ and $B_{max}$ is derived and explained in [7].

### *C. Calculation of CPU Utilization and CPU Regression Model from Measurements*

The HIL system is distributed on two devices and therefore a CPU utilization estimation for each device is necessary. While running the system the CPU load is recorded. These recordings not only contain streaming related workload, but also background and logging tasks. This needs to be considered when comparing the CPU load to the utilization estimations from the NC approach. As the additional processes are not represented by the NC approach, we will have an offset y and a factor x to match both systems, resulting in the following equations 6 for each number of streams $i$:

$$U_{measured}(i) = x \cdot i \cdot U_{NC}(i) + y \quad (6)$$

We used simple regression to generate a CPU load model and derive values for the offset and the linear multiplication factor. To calculate the CPU utilization from the estimated NC curves, we derived equation 7:

$$U_{NC}(i) = \frac{100}{num\_cores} \cdot \sum_{n=1}^{i} \frac{r_\alpha(i)}{r_\beta(i)} \quad (7)$$

## IV. Case Study

### *A. Case Study Description*

The case study is based on a multiple RADAR data stream with variable bit-rate. We performed measurements of SW processing unit timestamps and CPU utilization logs with 1 to 10 streams in parallel and used these for input modeling and NC model generation. Latency and backlog bounds were calculated with NC and the minimum pre-buffer delay $D_{max}(T8-T1)$ and system backlog $B_{max}(T8-T1)$, which is assumed to be equal to the maximum playback-buffer backlog.

### *B. Measurement and NC Results for Delay and Backlog*

Table I shows the measured values and the results derived from the NC approach. The tightness is calculated by dividing the NC result by the measured value.

TABLE I
NC and Measurement Comparison

| num streams | 1 | 4 | 8 | 10 |
|---|---|---|---|---|
| $D_{max}^{measure}[s]$ | 0.15 | 0.12 | 0.21 | 2.10 |
| $D_{max}^{NC}[s]$ | 0.66 | 0.56 | 0.53 | 2.39 |
| $D_{max}^{tightness}$ | 4.25 | 4.73 | 2.52 | 1.14 |
| $B_{max}^{measure}[MiB]$ | 0.17 | 0.13 | 0.19 | 1.80 |
| $B_{max}^{NC}[MiB]$ | 0.68 | 0.60 | 0.57 | 2.18 |
| $B_{max}^{tightness}$ | 4.11 | 4.61 | 2.94 | 1.20 |

The tightness factor is always greater than 1, so we derive always valid bounds. However, the factor decreases from 4 to 1 as we scale up from 1 to 10 streams.

### *C. Comparing HIL Measurements, NC Model and Regression Model for CPU Utilization*

The linear regression approach to estimate the CPU utilization of the Host PC results in equation 8. Similarly it can be calculated for the HIL PC in equation 9.

$$U_{ROS}^{\epsilon} = 0.71 \cdot U_{ROS}^{NC} + 3.03 \quad (8)$$

$$U_{LV}^{\epsilon} = 0.84 \cdot U_{LV}^{NC} + 20.86 \quad (9)$$

Table II displays the measured, estimated and calculated utilization using the NC arrival & service-rates and a linear regression model.

TABLE II
CPU Utilization

| num streams | 1 | 4 | 8 | 10 |
|---|---|---|---|---|
| $U_{ROS}^{measure}$ | 1.41 | 4.60 | 7.68 | 11.35 |
| $U_{ROS}^{\epsilon}$ | 3.04 | 3.59 | 6.54 | 11.87 |
| $U_{ROS}^{NC}$ | 0.02 | 0.20 | 0.62 | 1.25 |
| $U_{LV}^{measure}$ | 10.93 | 29.05 | 24.80 | 29.44 |
| $U_{LV}^{\epsilon}$ | 20.87 | 21.07 | 22.28 | 30.00 |
| $U_{LV}^{NC}$ | 0.01 | 0.06 | 0.21 | 1.09 |

## V. Discussion

The tightness of the NC bounds increases with the number of streams in our measurements, which is the opposite of what we expect. However, the bounds are tight until the maximum number of streams we could measure. Beyond 10 streams we were not able to log timestamps anymore. Despite the dependency on measurements, the FIFO residual curve approach seems to be a valid method for estimating NC bounds for multi-stream systems. The CPU utilization estimation for the ROS computer works well. The LV computers utilization estimation does not scale well. One reason for this could be the estimation method for the service-curve rate. However, even the CPU load measurements do not scale well in LV.

## VI. Conclusions

The NC bounds calculated with the FIFO residual curve approach result in valid upper bounds for the end-to-end delay and the system backlog. However, the increasing tightness of the bounds, as the number of streams increases, seems implausible. More number of streams need to be evaluated, but we reached a performance bottleneck of timestamp logging at 10 streams. With the NC model, we mainly want to estimate how many streams we are able to scale up in terms of memory usage by the backlog bound. The CPU utilization approach using the regression model results in valid estimations, but the model needs to be refined with more measurements and extrapolation must be incorporated into the model in order to estimate the bottleneck of CPU utilization by the number of streams. This will be addressed in future work.

## References

[1] Anne Bouillard, Laurent Jouhet, and Eric Thierry. Service curves in network calculus: dos and don'ts, 2009.

[2] R.L. Cruz. A calculus for network delay. i. network elements in isolation. *IEEE Transactions on Information Theory*, 37(1):114–131, 1991.

[3] R.L. Cruz. A calculus for network delay. ii. network analysis. *IEEE Transactions on Information Theory*, 37(1):132–141, 1991.

[4] Markus Fidler. Survey of deterministic and stochastic service curve models in the network calculus. *IEEE Communications Surveys and Tutorials*, 12(1):59–86, 2010.

[5] Christoph Funda. Arrival- and service-curve estimation methods from measurements to analyze and design soft real-time streaming systems with network calculus, 12 2022.

[6] Christoph Funda. Performance evaluation of HIL systems with NC. https://plassart.github.io/WoNeCa/2022/, 2022. Accessed: 2023-05-02.

[7] Christoph Funda, Tobias Konheiser, Thomas Herpel, Reinhard German, and Kai-Steffen 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)*, pages 1–7, 2022.

[8] Jean-Yves Le Boudec and Patrick Thiran. *Network Calculus: A Theory of Deterministic Queuing Systems for the Internet*, volume 2050. Springer Berlin, Heidelberg, 2001.

[9] L. Thiele, S. Chakraborty, and M. Naedele. Real-time calculus for scheduling hard real-time systems. In *2000 IEEE International Symposium on Circuits and Systems (ISCAS)*, volume 4, pages 101–104 vol.4, 2000.

[10] Amaury Van Bemten and Wolfgang Kellerer. Network calculus: A comprehensive guide, 10 2016.

[11] Ernesto Wandeler. *Modular Performance Analysis and Interface-Based Design for Embedded Real-Time Systems*. PhD thesis, ETH Zurich, 09 2006.

## Acknowledgment

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

## Appendix

### *System under study computer devices*

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

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 GBit 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.