Electronic Communications of the EASST
Volume 080 (2021)

Conference on Networked Systems 2021
(NetSys 2021)

Discrete event simulation for the purpose of real-time performance evaluation of distributed hardware-in-the-loop simulators for autonomous driving vehicle validation

Christoph Funda[a], Reinhard German[b], Kai-Steffen Hielscher[b]
5 Pages

Guest Editors: Andreas Blenk, Mathias Fischer, Stefan Fischer, Horst Hellbrück, Oliver Hohlfeld, Andreas Kassler, Koojana Kuladinithi, Winfried Lamersdorf, Olaf Landsiedel, Andreas Timm-Giel, Alexey Vinel
ECEASST Home Page: http://www.easst.org/eceasst/ ISSN 1863-2122

Discrete event simulation for the purpose of real-time performance evaluation of distributed hardware-in-the-loop simulators for autonomous driving vehicle validation

Christoph Funda[a], Reinhard German[b], Kai-Steffen Hielscher[b]

[a] Zukunft Mobility GmbH (a company of ZF Group), Marie-Curie-Str. 5 /5a, 85055 Ingolstadt, Germany
[b] University Erlangen-Nuremberg, Department Computer Science, Martensstr. 3, 91058 Erlangen, Germany

Abstract: Hardware-in-the-loop test benches are distributed computer systems including software, hardware and networking devices, which require strict real-time guarantees. To guarantee strict real-time of the simulator the performance needs to be evaluated. To evaluate the timing performance a discrete event simulation model is built up. The input modeling is based on measurements from the real system in a prototype phase. The results of the simulation model are validated with measurements from a prototype of the real system. The workload is increased until the streaming source becomes unstable, by either exceeding a certain limit of bytes or exceeding the number of parallel software processes running on the cores of the central processing unit. To evaluate the performance beyond these limits, the discrete event simulation model needs to be enriched by a scheduler and a hardware model. To provide real-time guarantees an analytical model needs to be built up.

Keywords: distributed system, networks, hardware-in-the-loop simulation, HIL, real-time performance evaluation, discrete event simulation, DES, OMNeT++

1 Introduction

Autonomous driving functions are safety-critical and have to be tested thoroughly. The hardware-in-the-loop (HIL)-based testing approach is an enabler for this and has been established in the automotive industry for years as an effective and efficient method of validating control units, interfaces, and functions. Since the system under test works under hard real-time (RT) conditions, the HIL test bench needs to be RT capable to enable the testing and be able to check the RT properties of the system under test. The focus here is on the use-case of the so-called open-loop reinjection [1], as streaming of measurement data to the device under test (DUT) under strict RT constraints. The HIL test system needs to be assessed regarding the RT capability of the whole streaming chain. Although streaming tones down the RT requirements within the chain to soft RT and with the help of over-provisioning and pre-buffering the design process can be eased, there are a lot of dimensioning issues for buffers and throughput left, for that purpose we have to analyze the system more deeply.

2 Approach

A HIL is a distributed computer system, for which common methods from computer system performance evaluation can be applied to evaluate the RT performance. At first, a very simplified conceptual model has been developed to understand the system, set up requirements and be able to develop a model from it.

The HIL system under analysis consists of two distributed computer systems as depicted in Figure 1 connected via network interface cards (NIC) and an Ethernet connection using the TCP-IP protocol. Each computer has an application layer, a Linux operating system (OS) and a hardware layer (HW). Additionally, the left PC has the Robot Operating System (ROS) [2] as a middle layer between the application layer and the Linux OS with RT preempt patch [3]. The streaming application is running as a multi-threaded multi-processing ROS application and it is instrumented at the user-code level in C++ to gain timing information when a process starts and when it is finished. The measurements from the system will be used for input modeling in the simulation model.

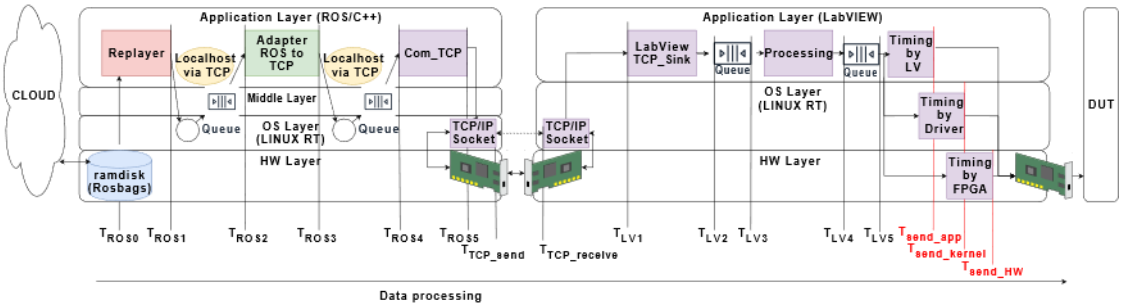

Figure 1: Detailed Conceptual Model and Software (SW) Instrumentation.

The streaming process starts at the far-left SW process, the ROS replayer, that grabs the data from the RAM disk and sends it, according to its timestamps, to the next node. The interprocess-communication is done via TCP/IP and the data is sent to a queue in the ROS middle layer. This is done by every process until the data reaches the last node. Then the data is sent over the TCP socket to the network stack of Linux depicted in the kernel layer to the network interface card (NIC). On the right PC, the process is similar, but starting from the NIC to receive the TCP segments via interrupts over the kernel to the first application process in LabVIEW. Within this first process, the segments are collected and reassembled or segmented to the original frame size, depending on the original frame size in relation to the maximum segment size (MSS). When the original frame is restored, it will be sent to the next process until the last queue, before sending it out to the device under test (DUT). This last queue has a very important function, which is the compensation of the delay and jitter introduced by the whole processing and queuing chain.

To evaluate the performance of a computer system, according to [4] at least two of the following three methods are needed to validate each other: measurements, simulation and analytical modeling. If the system is physically available, measurements should be performed to understand the actual system behavior, and for practical purposes to recognize and correct malfunctions. In addition, a simulation or an analytical model should be generated to extrapolate the results to unavailable systems. The generated model should be validated with measurements.

In the next step, a discrete-event simulation (DES) model [7] must be built to gain a better understanding of the system. Discrete event simulation is particularly suitable for evaluating the stochastic behavior of processes in computer and communication systems. [8]

3 Discrete Event Simulation Model

The model is build up according to the conceptual model of Figure 1 in OMNeT++ with standard servers, queues and delay elements and with the INET framework for the networking unit using the TCP/IP protocol implementation.
For validation purposes, the model is fed with measurements, called trace-driven simulation [4]. Then the simulation output is compared to the measurements. Latency measurements are taken as service times of the servers in the model, and we compare these service times of the simulation output of the model and the measurements as depicted in Figure 2.

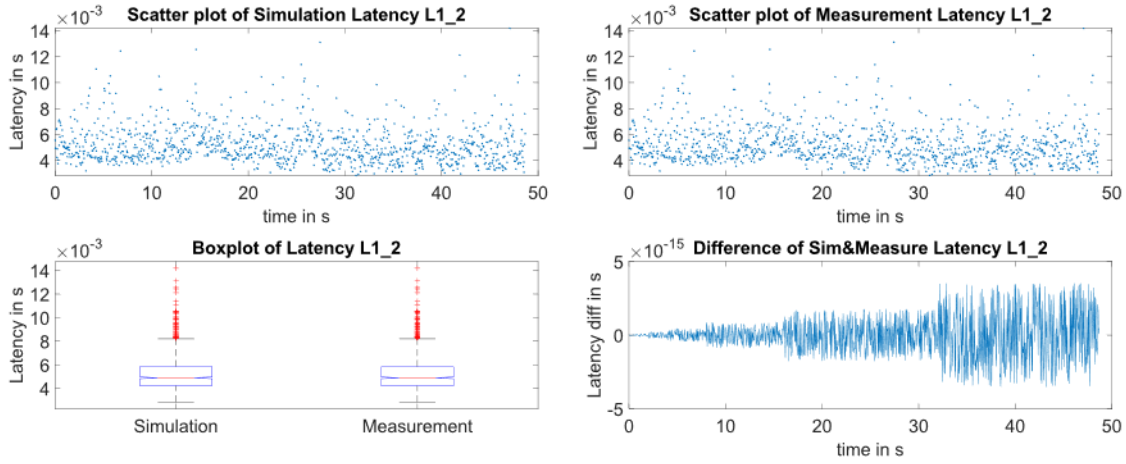

Figure 2: Example of process service times; from left to right:
upper figures: scatter plot of simulation and measurement data,
lower figures: Boxplot of Simulation and Measurements, Difference between Simulation and Measurements at each point over time.

As can be seen from Figure 2, the simulated and the measured data are highly comparable. From the boxplots, we can see that they are distributed equally. The last right plot shows the calculated differences between the simulation and measurement latency. This is in the range of 10^{-15} s for processing latency and 10^{-7} s in case of end-to-end latency. The conclusion is that the conceptual model of one data stream is representing the actual system in a very exact way.
As the simulation results of the model are satisfying, a mapping of the stochastic process behavior with theoretical distribution functions can be started. This will bring independence of a specific trace and simulate the behavior over a longer period and with random state combinations.
For distribution fitting from the measurement data, MATLAB is used to fit the data into different statistical distributions, which are supported by OMNeT++. In a further step, the distribution is truncated to the minimum and maximum measured values. Then a goodness of fit analysis as described in [5] is performed. The distribution with the highest mean p-value is taken and feed into the model. As can be seen exemplarily in Figure 3, the process latencies are lognormal distributed with a mean p-value of 37%, which met the requirement to be at least at 5%.

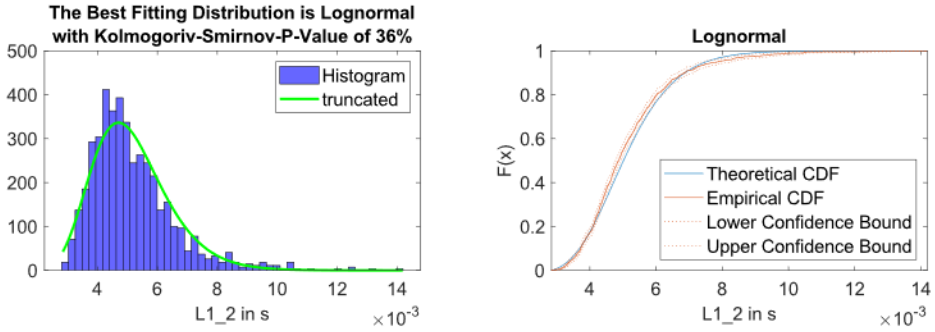

Figure 3: from left to right: Histogram and Distribution Fit of ROS Process Latencies, Comparison between Theoretical & Empirical Cumulative Distribution Function (CDF).

The limits of the model can be seen in Figure 4 compared to a nearly singular distributed stream on the right as a baseline. The influence of increasing the byte size of the workload can be seen in the left diagram. In the middle graph, we see the influence on the distribution when performing many streams in parallel, so that there are more processes than CPU cores available.

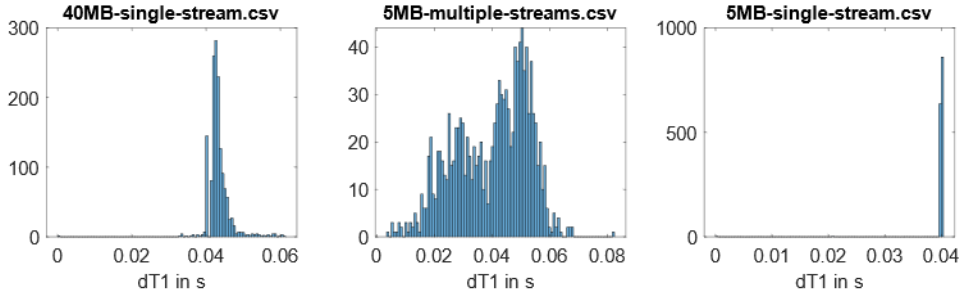

Figure 4: Histogram plots of 40ms cycle-time measured stream; from left to right: 40MB single stream; 5MB stream with 16 other streams in parallel, 5MB single stream.

4 Conclusions

The simulation model is representing the real-world prototype in a good way, and it is valid until we reach the HW bottlenecks of the system with our workload. Enrichment of the model with an OS scheduler and a CPU model, a so-called SW&HW co-simulation as described in [10][10][12] will overcome these model limits including the coupling of network simulation as described in [13][14]. With this approach we can give at least soft RT guarantees within a given statistical confidence. The final goal is to make predictions about holding RT limits of our SW&HW System for multi-streaming purposes. The predictions will be based on a certain set of sensors with a payload and cycle times as SW parameters in combination with various HW parameters like CPU frequency and the number of cores as an optimization parameter. To give guarantees and a full proof of hard RT capability, an analytical model must be used. Analytical models that are worthwhile to explore in more detail are the most used techniques under the hood of many performance engineering tools like layered queueing networks, process algebra, Petri nets and scheduling theories of real-time systems like rate monotonic analysis.[15][16][17]

5 References

[1] N. Brayanov and A. Stoynova, “Review of hardware-in-the-loop -a hundred years progress in the pseudoreal testing”, 2019

[2] M. Quigley et al., „ROS: an open-source Robot Operating System“, 2009

[3] F. Reghenzani, G. Massari and W. Fornaciari, „The real-time linux kernel: A survey on Preempt_RT“,2019

[4] R. Jain, “The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation, and Modeling”, 1. Edition. New York: Wiley, 1991.

[5] D. Nurmi, J. Brevik, and R. Wolski: “Modeling Machine Availability in Enterprise and Wide-area Distributed Computing Environments”, 2005

[6] M. Quigley, B. Gerkey, K. Conley, J. Faust, T. Foote, J. Leibs, E. Berger, R. Wheeler, and A. Ng, “ROS: an open-source Robot Operating System,”, 2009

[7] J. Banks, J. S. Carson, B. L. Nelson, D.M. Nicol. “Discrete-Event System Simulation”, 2000

[8] A. M. Law, W. D. Kelton. “Simulation Modeling & Analysis”, 2000

[9] A. Varga; R. Hornig. “An overview of the OMNeT++ simulation environment”, 2008

[10] P. Razaghi, A. Gerstlauer, „Host-compiled multicore RTOS simulator for embedded real-time software development“, 2011

[11] P. Razaghi, A. Gerstlauer, „Host-Compiled Multicore System Simulation for Early Real-Time Performance Evaluation“, 2014

[12] O. Bringmann *et al.* „The Next Generation of Virtual Prototyping: Ultra-Fast Yet Accurate Simulation of HW/SW Systems“, 2015

[13] Z. Zhao, V. Tsoutsouras, D. Soudris, A. Gerstlauer, „Network/system co-simulation for design space exploration of IoT applications“, 2017

[14] G. Amarasinghe, M. D. de Assunção, A. Harwood, S. Karunasekera, „ECSNeT++ : A simulator for distributed stream processing on edge and cloud environments“, 2020

[15] U. Herzog, “Formal Methods for Performance Evaluation”, 2001

[16] A. Purhonen, “Performance Evaluation Approaches for Software Architects“, 2005

[17] L. Etxeberria “Method for analysis-aided design decision making and quality attribute prediction”, 2010