Proceedings of the

# 46th IEEE Real-Time Systems Symposium - Brief Presentations Track (RTSS 2025 BP)

## Work-Already-Published and RTSS@Work papers

Boston, MA, USA

December 2-5, 2025

## Message from the Brief Presentations Chair

Welcome to the Brief Presentations (BP) Track of the 46th IEEE Real-Time Systems Symposium (RTSS 2025), held in Boston, MA, USA, in December 2025. The RTSS 2025 BP Track provides a platform for researchers to present their work in progress, announce relevant publications from journals or other sources that are of interest to the real-time systems community, and showcase demonstrations of prototypes, tools, simulators, or systems that advance the state of the art in technologies and techniques for real-time systems.

The BP Track shares the technical scope with RTSS 2025 and features three categories of submissions:

- Work-in-Progress (WiP) papers;
- Work-Already-Published (WAP) papers (i.e., journal papers not yet presented); and
- RTSS@Work demonstrations and experience reports.

This year, the BP Track accepted nineteen peer-reviewed papers. This booklet contains the accepted WAP extended abstracts and RTSS@Work demonstration summaries. Accepted WiP papers appear in the main RTSS 2025 proceedings published by IEEE.

I am grateful to the RTSS 2025 BP Technical Program Committee for their thorough evaulations and constructive feedback on the submitted papers, to the RTSS 2025 organizers for their support in arranging the RTSS 2025 BP Track, and to all authors who submitted their work.

Federico Aromolo – Scuola Superiore Sant'Anna, Pisa, Italy
RTSS 2025 Brief Presentations Chair

## Technical Program Committee

Tanya Amert – Carleton College, USA

Bjorn Andersson – Carnegie Mellon University, USA

Mohammad Ashjaei – Mälardalen University, Sweden

Vijay Banerjee – Washington State University, USA

Joshua Bakita – University of North Carolina at Chapel Hill, USA

Andrea Bastoni – Technical University of Munich, Germany

Matthias Becker – KTH Royal Institute of Technology, Sweden

Florian Brandner – Télécom Paris, France

Daniel Casini – Scuola Superiore Sant'Anna, Pisa, Italy

Weifan Chen – Boston University, USA

Hyunjong Choi – San Diego State University, USA

Cédric Courtaud – Huawei, France

Xiaotian (Steven) Dai – University of York, UK

Anaïs Finzi – TTTech Computertechnik AG, Austria

Giovani Gracioli – Federal University of Santa Catarina, Brazil

Monowar Hasan – Washington State University, USA

Qingqiang He – Great Bay University, China

Tomasz Kłoda – LAAS-CNRS, France

Ruoxiang Li – City University of Hong Kong, China

Yehan Ma – Shanghai Jiao Tong University, China

Alberto Marchetti-Spaccamela – Sapienza University of Rome, Italy

Federico Reghenzani – Politecnico di Milano, Italy

Yuxin Ren – Huawei Technologies, China

Sepideh Safari – Institute for Research in Fundamental Sciences, Iran

Marion Sudvarg – Washington University in St. Louis, USA

Binqi Sun – Technical University of Munich, Germany

Youcheng Sun – Mohamed Bin Zayed University of Artificial Intelligence, UAE

Ashutosh Tadkase – NVIDIA, USA

Harun Teper – TU Dortmund, Germany

Corey Tessler – University of Nevada, Las Vegas, USA

Yidi Wang – Santa Clara University, USA

Bingkun Yao – City University of Hong Kong, China

Patrick Meumeu Yomsi – Instituto Politécnico do Porto, Portugal

Raffaele Zippo – University of Pisa, Italy

# Contents



*Proceedings edited by Federico Aromolo - RTSS 2025 Brief Presentations Chair.*

# Work Already Published: Limited-Preemption EDF Scheduling for Multi-Phase Secure Tasks

Benjamin Standaert*, Fatima Raadia†, Marion Sudvarg*,
Sanjoy Baruah*, Thidapat Chantem‡, Nathan Fisher†, Christopher Gill*

*Department of Computer Science & Engineering, Washington University in St. Louis, St. Louis, MO, USA
†Wayne State University, Detroit, MI, USA ‡Virginia Tech, Blacksburg, VA, USA
*(b.g.standaert, msudvarg, baruah, cdgill)@wustl.edu †(fatima.fr, fishern)@wayne.edu ‡tchantem@vt.edu

***Abstract*—Safety-critical systems may use mechanisms such as Trusted Execution Environments or specialized coprocessors to isolate certain tasks and ensure security; however, this may result in task preemptions crossing an isolation boundary, leading to additional performance overhead. If this overhead is not considered during scheduling, it may result in deadline misses. For this reason, our prior work [2] introduced the Multi-Phase Secure (MPS) task model, which models tasks that execute in phases with different security requirements and corresponding setup/teardown costs. In our work-already-published [9], we make corrections to the prior work (including to a long-standing schedulability condition for EDF under limited preemption) and introduce optimizations which significantly improve the performance of our algorithm for determining schedulability of MPS task sets. This improvement allows us to test the schedulability of larger task sets; consequently, we gain a more detailed understanding of the benefit of applying the MPS task model to task sets with varying task parameters.**

***Index Terms*—Real-Time Systems, Limited-Preemption Scheduling, Trusted Execution Environments**

## I. Overview

In today's world, real-time systems are increasingly expected to operate securely, especially in domains like IoT, automotive, and edge computing. Mechanisms such as trusted execution environments (TEEs) provide strong isolation but introduce additional timing and resource constraints, particularly when tasks must execute across secure and non-secure contexts. These challenges impact traditional scheduling approaches due to phase-specific execution and preemption requirements.

Several works have addressed the trade-off between security and schedulability in real-time systems. [8] and [7] use MILP formulations to optimize system resilience under attack, while [10] explores runtime security overheads. However, these works overlook preemption-related costs introduced by security mechanisms. Limited-preemption scheduling has been studied as a means to balance blocking times with preemption overheads [1], [3], [6]. More recent work introduces the idea of selecting preemption points under the assumption of constant preemption overhead [4], [12], or per-block preemption costs [5], but does not address task models with phase-specific preemption overheads as considered in this paper.

Work funded by NSF Grants CPS-1932530, CNS-2141256, CNS-2229290, IIS-1724227, CNS-2038609, CCF-2118202, CNS-2211641, CPS-2038726.

In this work, we consider the *Multi-Phase Secure* (**MPS**) task model, in which tasks are divided into execution phases across distinct security mechanisms that each incur unique startup and teardown overheads. Based on this model, we adapt the Earliest Deadline First (EDF) scheduling algorithm to allow preemption only at controlled points, effectively balancing blocking times and preemption costs. To enforce this, we adopt the *fixed-preemption point model* [11], where preemption is only allowed at statically inserted points determined prior to run-time. Given a maximum chunk size $\beta_i$ for each task $\tau_i$, the algorithm selects a set of preemption points within each phase such that no non-preemptive region exceeds $\beta_i$. These points are selected to trade off between increased preemption overhead and reduced blocking of higher-priority tasks. Importantly, the algorithm is *optimal* in the sense that if a schedulable set of preemption points exists that satisfies all timing and security constraints, it will find such a selection.

Our work-already published [9] extends, corrects, and refines prior results on Multi-Phase Secure (MPS) task scheduling [2], introducing several important contributions. First, it **corrects a longstanding EDF schedulability condition for limited-preemption tasks**, improving the theoretical foundation of the analysis. Building on this correction, the paper **presents a pseudo-polynomial schedulability tests for MPS task sets with bounded utilization**, significantly enhancing the efficiency of existing algorithms, particularly for tasks with implicit deadlines. The paper compares MPS-based scheduling to traditional approaches that are non-preemptive or that allow only a single preemption between task phases. It demonstrates that **the MPS task model improves task set schedulability while reducing analysis complexity**, thereby broadening its applicability to both safety-critical and general-purpose embedded systems. Additionally, the paper **resolves inconsistencies between the theoretical model and its implementation** by adopting a continuous-time representation, resulting in clearer and more accurate performance comparisons with non-preemptive schedulers.

## II. Summary of Results

### A. Schedulability

In an MPS system, introducing preemption points reduces blocking time, which improves schedulability; but it also intro-

duces additional preemption overhead, which can act to reduce schedulability. Therefore, the amount by which schedulability is improved using our algorithm varies depending on the task set parameters. Prior work [2] examined only a small number of task sets. In our work-already-published [9], we examine a much larger space of task sets. We observe the following relationships between the task set parameters and schedulability:

- Our algorithm improves schedulability for task sets with moderate-to-high utilization of a single processor, but whose total utilization remains under 1. For systems with $U = 1$, it is not possible to insert additional preemption points, as doing so would increase utilization.
- Task sets with larger task counts or more phases per task have higher schedulability ratios. In these cases, the system's execution time is split among more task phases, lowering the average blocking time of a single phase.
- Tasks with small periods are much less likely to be schedulable; we hypothesize that these tasks are less expandable with additional preemption points.
- For *constrained-deadline systems*, our algorithm also improves schedulability, but increasing the number of tasks or phases has less of an impact on schedulability.

In [9], we present plots of the schedulability ratios obtained while varying all of these dimensions of the task set, as well as comparisons of our algorithm with a phase-nonpreemptive approach.

### B. Performance improvements

Prior work [2] presented results only for systems containing 3 tasks; larger systems took too long to evaluate. In [9], we correct implementation issues in the original and rewrite the original Python implementation in C++. For sets of 4 tasks, these changes alone improve the median execution time by around $150\times$ and the mean execution time by $3500\times$.

We also reduce the time complexity of the original algorithm in [2]. We do so first by proving exactness of a pseudo-polynomial testing set for bounded-utilization task sets. We also introduce an optimization for implicit-deadline task sets by showing that we need only test only up to the maximum deadline of any single task in the set, which dramatically improves performance. Figure 1 shows the improvement obtained in execution times compared to the larger original testing set. These performance improvements enable testing of much larger task sets. In [9], we test sets containing up to 20 tasks, and provide further details regarding the performance and schedulability results obtained on these larger systems.

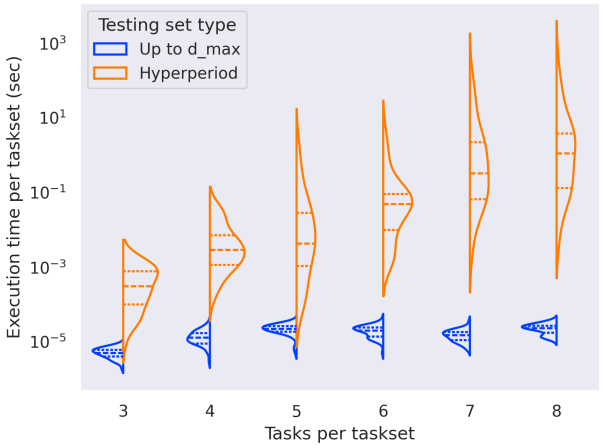



Fig. 1. As presented in [9]: Performance impact of varying the testing set size in the C++ implementation, either testing up to the hyperperiod or stopping at $D_{\max}$. Task sets have implicit deadlines, periods of 10–30 time units, 1–4 phases, and utilization of 0.9. Note the logarithmic scale.

## References


[1] Sanjoy Baruah. The limited-preemption uniprocessor scheduling of sporadic task systems. In *Proceedings of the EuroMicro Conference on Real-Time Systems*, 2005. https://dx.doi.org/10.1109/ECRTS.2005.32.

[2] Sanjoy Baruah, Thidapat Chantem, Nathan Fisher, and Fatima Raadia. A scheduling model inspired by security considerations. In *2023 IEEE 26th International Symposium on Real-Time Distributed Computing (ISORC)*, pages 32–41. IEEE, 2023. https://dx.doi.org/10.1109/ISORC58943.2023.00016.

[3] Marko Bertogna and Sanjoy Baruah. Limited preemption EDF scheduling of sporadic task systems. *IEEE Transactions on Industrial Informatics*, 2010. https://dx.doi.org/10.1109/TII.2010.2049654.

[4] Marko Bertogna, Giorgio Buttazzo, Mauro Marinoni, Gang Yao, Francesco Esposito, and Marco Caccamo. Preemption Points Placement for Sporadic Task Sets. In *2010 22nd Euromicro Conference on Real-Time Systems*, pages 251–260, 2010. https://dx.doi.org/10.1109/ECRTS.2010.9.

[5] Marko Bertogna, Orges Xhani, Mauro Marinoni, Francesco Esposito, and Giorgio Buttazzo. Optimal selection of preemption points to minimize preemption overhead. In *2011 23rd Euromicro Conference on Real-Time Systems*, pages 217–227. IEEE, 2011. https://dx.doi.org/10.1109/ECRTS.2011.28.

[6] Giorgio C Buttazzo, Marko Bertogna, and Gang Yao. Limited preemptive scheduling for real-time systems. a survey. *IEEE Trans. Industr. Inform.*, 9(1):3–15, February 2013. https://dx.doi.org/10.1109/TII.2012.2188805.

[7] Cailani Lemieux-Mack, Kevin Leach, Ning Zhang, Sanjoy Baruah, and Bryan C Ward. Optimizing runtime security in real-time embedded systems. In *Proc. of Workshop on Optimization for Embedded and Real-time Systems (OPERA)*, December 2024.

[8] Sandro Di Leonardi, Federico Aromolo, Pietro Fara, Gabriele Serra, Daniel Casini, Alessandro Biondi, and Giorgio Buttazzo. Maximizing the security level of real-time software while preserving temporal constraints. *IEEE Access*, 11:35591–35607, 2023. https://dx.doi.org/10.1109/ACCESS.2023.3264671.

[9] Benjamin Standaert, Fatima Raadia, Marion Sudvarg, Sanjoy Baruah, Thidapat Chantem, Nathan Fisher, and Christopher Gill. Limited-preemption EDF scheduling for multi-phase secure tasks. *Leibniz Transactions on Embedded Systems*, 10(1):3–1, 2025. https://dx.doi.org/10.4230/LITES.10.1.3.

[10] Yujie Wang, Cailani Lemieux-Mack, Thidapat Chantem, Sanjoy Baruah, Ning Zhang, and Bryan C Ward. Partial context-sensitive pointer integrity for real-time embedded systems. In *2024 IEEE Real-Time Systems Symposium (RTSS)*, pages 415–426. IEEE Computer Society, 2024. https://dx.doi.org/10.1109/RTSS62706.2024.00042.

[11] Gang Yao, Giorgio Buttazzo, and Marko Bertogna. Feasibility analysis under fixed priority scheduling with fixed preemption points. In *2010 IEEE 16th International Conference on Embedded and Real-Time Computing Systems and Applications*, pages 71–80, 2010. https://dx.doi.org/10.1109/RTCSA.2010.40.

[12] Gang Yao, Giorgio Buttazzo, and Marko Bertogna. Feasibility analysis under fixed priority scheduling with limited preemptions. *Real-Time Systems*, 47(3):198–223, 2011. Publisher: Springer. https://dx.doi.org/10.1007/s11241-010-9113-6.

# Work Already Published: Learning-Assisted Schedulability Analysis: Opportunities and Limitations

Sanjoy Baruah
*Washington University*, USA
baruah@wustl.edu

Pontus Ekberg
*Uppsala University*, Sweden
pontus.ekberg@it.uu.se

Marion Sudvarg
*Washington University*, USA
msudvarg@wustl.edu

***Abstract*—Our work-already-published [2] presents the first (to our knowledge) Deep-Learning based framework for real-time schedulability-analysis that guarantees to never incorrectly mis-classify an unschedulable system as being schedulable, and is hence suitable for use in safety-critical scenarios. We relate applicability of this framework to well-understood concepts in computational complexity theory: membership in the complexity class NP. We apply the framework upon the widely-studied schedulability analysis problem of determining whether a given constrained-deadline sporadic task system is schedulable on a preemptive uniprocessor under Deadline-Monotonic scheduling (a problem which is known to be NP-complete in general). A proof-of-concept implementation demonstrates a predictive accuracy exceeding 70% for systems of as many as 20 tasks *without making any unsafe predictions*. Furthermore, the implementation has very small ($<1$ ms on two widely-used embedded platforms; $<4$ $\mu$s on an embedded FPGA) and highly predictable running times.**

With Deep Learning (DL) already widely used in autonomous Cyber-Physical Systems (CPS's) for purposes of perception, research efforts are underway to also use it to *speed up computation*. In a recently published paper [2], we investigate the use of DL to speed up *schedulability analysis*. Many basic and fundamental forms of schedulability analysis are known to be computationally intractable and hence applying DL to speed it up seems a reasonable goal. However, schedulability is frequently a safety-critical property: incorrectly mis-classifying an unschedulable system as being schedulable could have potentially catastrophic consequences. There is, to our knowledge, no prior DL-based schedulability analysis that guarantees to never return 'false positives' – to incorrectly declare some unschedulable system to be schedulable. In our work-already-published, *we propose the first conceptual framework for using Deep Learning for schedulability analysis that guarantees to return no false positives*, and is hence suitable for use in safety-critical systems [2].

## I. A Naïve Approach

Our work explores whether we can train *Learning-Enabled Components* (LECs) to classify system specifications as either satisfying a given schedulability property, or failing to do so. As a first step towards achieving this goal, we trained simple *multilayer perceptrons* (MLPs) to perform binary classification for predicting FP and EDF schedulability for sporadic task systems of 4 tasks in accordance with the framework of Fig. 1.

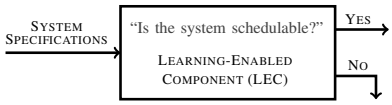
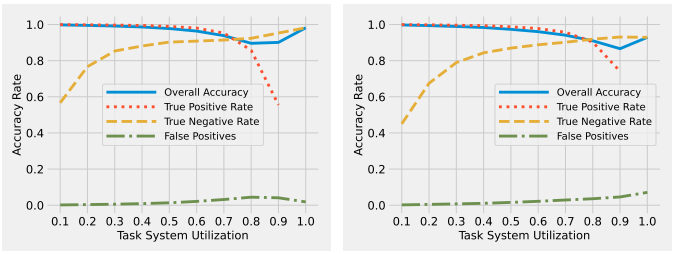



Fig. 1. LEC-based schedulability analysis




(a) FP Schedulability (b) EDF Schedulability

Fig. 2. Performance of DNN schedulability classifiers for systems of 4 tasks.

The observed performance of these networks are presented in Fig. 2. While DL appears to be very effective in classifying systems as schedulable or not, it makes occasional mistakes: classification accuracy is not 100% for either FP or EDF schedulability analysis. We must understand the consequences of its errors, and take mitigative steps to ensure they do not compromise system safety, before we can use LEC-based schedulability analysis in safety-critical systems.

We point out that classification errors are of two kinds: a FALSE NEGATIVE, with a schedulable system incorrectly classified as being unschedulable; or a FALSE POSITIVE, whereby an unschedulable system is classified as being schedulable. While a false negative may result in a schedulable system being needlessly rejected as being unschedulable, *false positives present a safety hazard* since a potentially unschedulable system is misidentified as being schedulable. Though the number of false-positives for our binary classifiers were low (of the systems of 4 tasks that we generated, 1.8% were incorrectly deemed DM schedulable and 2.1% EDF schedulable), *the only acceptable rate for safety-critical systems is zero* and so we must be able to eliminate *all* false positives if we are to use DL for schedulability-analysis for safety-critical systems.

## II. A Safe Framework

To eliminate the possibility of false positives, when DL-based components declare a system to be schedulable, they should also generate a *justification* for this decision in the form

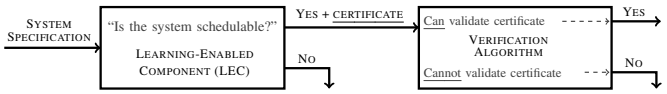



Fig. 3. A framework for LEC-based safety verification.

of a ***certificate***. This certificate must be *efficiently verifiable* by a (different) algorithm that is based on 'traditional' algorithmic techniques in that it does not make use of Deep Learning and related AI techniques; it is only if this verification algorithm agrees that the certificate validates schedulability do we deem the system specifications to have passed the schedulability-analysis test. This proposed enhanced framework for DL-based schedulability analysis is depicted in Fig. 3.

Recall that our goal in using DL for schedulability analysis is to obtain greater run-time efficiency. There is ample research on how one should implement LECs to have efficient (and predictable) running times (see, e.g., [7]); we expect that one can use the results of this research to obtain very efficient implementations of the LEC in Fig. 3 (indeed, we demonstrate examples of this in [2]).

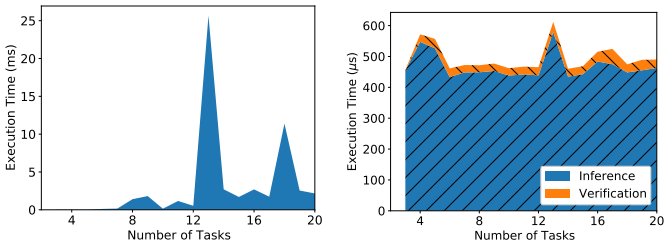
We also want the verifier to be efficient. We argue that it is reasonable to require that it should have running time no worse than a polynomial in the size of the considered task system. This requirement immediately relates the applicability of the framework to well-studied concepts in computational complexity theory [9], [1], in particular, the complexity class NP – "NP *is the class of [problems] that can be verified by a polynomial-time algorithm.*" [3, p. 1058]. Hence the requirement that the certificate be verifiable in polynomial time implies that the framework is applicable to schedulability-analysis problems that are in NP.

Hence, in order to determine whether a schedulability-analysis problem can be verified using DL through our framework or not, it is necessary to demonstrate its membership (or non-membership, respectively) in NP. To prove that a schedulability-analysis problem belongs to NP, one must furnish a polynomial-time verification algorithm for the problem. However, how can one demonstrate its *non*-membership in NP? In this case, established results from computational complexity theory come into play. There exist various complexity classes that are very widely believed to be distinct from NP, meaning they contain problems $\notin$ NP. A problem is considered *hard* for a complexity class if it is at least as computationally difficult to solve as every other problem within that class. Thus, *showing a schedulability-analysis problem to be hard (or complete) for any complexity class believed to be distinct from* NP *(such as* coNP*) provides substantial evidence that it is not a member of* NP.

## III. Examples

As FP-schedulability analysis is NP-complete [5] and therefore in NP, it fits within our framework. EDF-schedulability analysis, however, does not; it is coNP-complete [4] and therefore not in NP (assuming NP $\neq$ coNP).

It has been shown [6], [8], [10] that a necessary and sufficient FP-schedulability condition for task system $\Gamma$ is that for each $\tau_i \in \Gamma$, the recurrence $R_i \geq C_i + \sum_{\tau_j \in \mathrm{hp}(\tau_i)} \left\lceil \frac{R_i}{T_j} \right\rceil \cdot C_j$ should




(a) Response-Time Analysis (b) LEC Framework

Fig. 4. Worst-observed execution times on a Raspberry Pi 4.

have a positive solution for $R_i$ that is no larger than $\tau_i$'s relative deadline $D_i$. A certificate for the FP-schedulability of $\Gamma$ could simply be a value for $R_i$ per task; given such a certificate, the module labeled VERIFICATION ALGORITHM in Fig. 3 can clearly efficiently verify that for each $\tau_i \in \Gamma$, the provided value of $R_i$ does indeed satisfy the recurrence and is $\leq D_i$.

To investigate whether we could get LECs to generate such certificates, we trained an alternative set of MLPs to predict the $R_i$ values via regression, rather than simply providing a binary classification. Although accuracy overall decreases slightly with verification (from $85.1\%$ to $82.7\%$), **unsafe false positives are eliminated entirely**. Moreover, analysis completes with **fast** ($<$1 ms on two widely-used embedded platforms; $<4$ $\mu$s on an embedded FPGA) **and highly predictable running times**. See, for example, the worst-observed execution times in Fig. 4.

More details on the chosen DNN architectures, training methods, and accuracy results may be found in [2]. Also detailed are execution time statistics comparing the LEC for FP-schedulability to traditional response-time analysis, FPGA implementation results, and preliminary **results for the harder (though still in** NP**) problem of partitioned FP-schedulability**. As future work, we intend to collaborate with AI/ML experts to improve upon the predictive power for such hard problems, with the goal of enabling **fast, accurate schedulability analysis free of false positives**.

## References

[1] S. Arora and B. Barak. *Computational Complexity - A Modern Approach*. Cambridge University Press, 2009.
[2] S. Baruah, P. Ekberg, and M. Sudvarg. Learning-assisted schedulability analysis: opportunities and limitations. *Real-Time Systems*, Jul 2025.
[3] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms.* MIT Press, fourth edition, 2022.
[4] F. Eisenbrand and T. Rothvoß. EDF-schedulability of synchronous periodic task systems is coNP-hard. In *Proceedings of the Annual ACM-SIAM Symposium on Discrete Algorithms*, Jan 2010.
[5] P. Ekberg and W. Yi. Fixed-priority schedulability of sporadic tasks on uniprocessors is NP-hard. In *2017 IEEE Real-Time Systems Symposium, RTSS 2017, Paris, France, December 5-8, 2017*, pages 139–146. IEEE Computer Society, 2017.
[6] M. Joseph and P. Pandya. Finding response times in a real-time system. *The Computer Journal*, 29(5):390–395, Oct. 1986.
[7] W. Kang and J. Chung. DeepRT: predictable deep learning inference for cyber-physical systems. *Real-Time Systems*, 55(1):106–135, Jan 2019.
[8] J. Lehoczky, L. Sha, and Y. Ding. The rate monotonic scheduling algorithm: Exact characterization and average case behavior. In *Proceedings of the Real-Time Systems Symposium - 1989*, pages 166–171, Santa Monica, California, USA, Dec. 1989.
[9] C. H. Papadimitriou. *Computational Complexity*. Addison-Wesley, 1994.
[10] A. Wellings, M. Richardson, A. Burns, N. Audsley, and K. Tindell. Applying new scheduling theory to static priority pre-emptive scheduling. *Software Engineering Journal*, 8:284–292, 1993.

# RTSS@Work: Toward Fine-grained Performance Tracing on ARM-based SoC/FPGA platforms

Patrick Carpanedo*, Denis Hoornaert†, Marco Caccamo†, Renato Mancuso*
*Boston University, †Technical University of Munich
*{pfcarp21, rmancuso}@bu.edu, †{denis.hoornaert, mcaccamo}@tum.de

## I. Introduction

As modern System-on-Chip (SoC) architectures and workloads become increasingly complex, so does their interplay. Hence, in disciplines such as real-time systems, where stringent timing guarantees must be upheld, workload profiling has become a crucial design step. The objectives are to gain an understanding of (1) the hardware/software interplay, (2) the SoC's resource utilization, and (3), in particular, the root cause of inter-core interference.

Efforts in systems design communities have produced *static* and *online* software methods for profiling and analysis purposes. Static approaches analyze portions of code offline to provide insight. This class of methods is faster but offers limited analysis due to strong assumptions of hardware and execution. Modern online approaches, such as RT-bench [1], rely on performance counters to track a limited amount of resource statistics related to the workload execution. This has the advantage of swiftly and faithfully representing the software/hardware interplay. However, a trade-off must always be found between granularity and overheads, as dedicated thread(s) must continuously pool and store the performance counters' values. Furthermore, as the polling is asynchronous, the acquired data cannot be reliably mapped onto segments of the source code. Alternatively, hardware-based approaches, such as [2], [3], are well-rounded, closed-source solutions primarily focused on debugging purposes, which limit the total amount of information that can be represented.

We posit that an affordable and straightforward alternative exists; one that marries the user-friendliness of software-based approaches with the exactitude and low-overhead of hardware-based approaches. Our hardware/software co-designed approach named Debug over Ethernet (DoEth) takes advantage of the highly reprogrammable nature of modern SoC/FPGA platforms, such as the AMD-Xilinx UltraScale+ MPSoC series, to offer unmatched access to low-level activity of the SoC and privileged access to a plethora of high-performance peripherals. In this demonstration, we will briefly (1) outline our vision and (2) describe how our proposed prototype gathers fine-grained SoC's activity information and makes it accessible to the end-user.

This research was supported by the National Science Foundation (NSF) under grant number CSR-2238476. Marco Caccamo was supported by an Alexander von Humboldt Professorship endowed by the German Federal Ministry of Education and Research.

## II. System Overview

In a DoEth setup, we distinguish two actors: a *Guest* and a *Host* Computer. The former is the embedded computing system we desire to monitor the progress fo the workloads and resource utilization. Ideally, the guest's SoC should feature a tightly connected FPGA and direct access to high-bandwidth IO or hardwired peripheral controllers (e.g., Ethernet, SFP+) on which the data of interest will be placed. Importantly, the SoC's FPGA must have the possibility to collect debug information and performance information about the system (e.g., L2 refills). The host computer should have similar IO to receive the guest-originated data and should be fast enough to handle the incoming data rate.

### A. Prototyping on a Commercial Platform

For our DoEth prototype, we identified the UltraScale+ MPSoC family manufactured by AMD-Xilinx as a prime target. Following the organization depicted in Fig. 1, the trace data path can be divided into three segments.

**System-on-Chip side (guest)**. The SoC's side CPU cores (ARM Cortex A53) are equipped with a Performance Monitoring Unit (PMU) and are connected to the system's CoreSight. The latter can be instrumented to monitor (1) predefined watch points in the workloads' code and (2) a selected set of the PMU's performance events (e.g., L2 refill, instructions retired). Typically, the CoreSight produces packets upon a CPU core hitting one of the watch points or when a performance event has occurred a user-defined number of times.Importantly, the CoreSight fabric can be programmed to route packets to many destinations, one of which directly leads to the FPGA.

**FPGA side (guest)**. Deployed on the FPGA, DoEth is on the CoreSight's receiving end (see Fig. 1). Its goal is to relay the trace information (i.e., the packets) to the outside world (i.e., the host) via Ethernet over a transceiver connection. DoEth is composed of three components through which the ingress data flows: (1) a queue, (2) a *FrameFormer*, and (3), finally, an Ethernet controller. The queue is necessary as a Clock Domain Crossing (CDC) bridge between the fixed CoreSight bus frequency ($\pm 250$ MHz) and the fixed Ethernet controller frequency ($\pm 156$ MHz). Once the clock domain is crossed, the FrameFormer creates eligible Ethernet frames.[1] The frames are directly forwarded to the Ethernet controller for transmission to the host computer. The physical medium selected for DoEth

[1]The process consists of concatenating several CoreSight packets to form the frame's payload, before wrapping it in metadata.

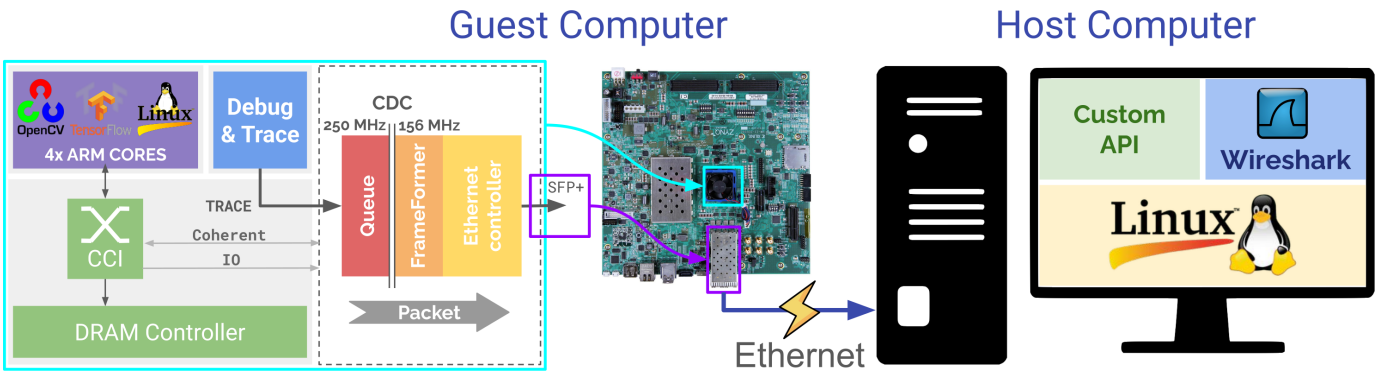



Fig. 1. Overview of the demonstration setup and in-FPGA DoEth architecture. Trace data flows from the *guest computer* to the *host computer*. The data (1) originate from the CoreSight, (2) traverse the FPGA's queue, the FrameFormer, and the ethernet controller, (3) reach the host over ethernet, and (4), finally, is picked up by the host using Wireshark and processed/displayed by our custom API.

is a dedicated transceiver line (SFP+). For simplicity, we implement AMD-Xilinx's high-performance Ethernet IP [4].

**Host desktop computer**. The host utilizes Mellanox ConnectX-2 Dual SFP+ NiC with the respective Short Reach (SR) optical transceivers to receive the payload from the FPGA. The packets are captured using programs that utilize libpcap (e.g., Wireshark) and saved in standard ASCII format. Thereafter, the information given by Wireshark is cleaned from all extraneous metadata and padding, reformatted to correct the endianness flip of the Ethernet controller, and parsed through a set of custom programs.

## III. Demonstration Plan

For the demonstration, we will showcase a prototype of DoEth implemented on the AMD-Xilinx ZCU102, an UltraScale+ development board. The prototype will be deployed alongside an OEM desktop computer equipped with a Network Interface Controller (NiC) card, which will act as our host computer. Both the ZCU102 and the desktop computer will be linked via an SFP+ transceiver, as illustrated in Fig. 1.

The objective of the demonstration is to highlight DoEth's ability to not only monitor various performance counters, but also to track curated watch-points in the guest computer's workloads. To this end, we devise a workload structured as an acyclic graph of steps that the spectator can interact with. More precisely, the ZCU102 will be equipped with a webcam whose frame will be fed into an Artificial Neural Network (ANN). The ANN determines whether a human is present in the frame (step a). Upon detecting a human, the frame is passed through a filter of the user's choice, selected via the keyboard (step b). Otherwise, the process restarts. The outcome of the filter is then compressed and stored.

Simultaneously, the host desktop computer receives the packets from DoEth and displays the results as they come. We envision a few plot types including (1) resource utilization over time, (2) time to watch-points, and (3) a live roofline model. Importantly, the interface will display when and which watch-point is hit, showcasing DoEth's ability to follow a Control Flow Graph (CFG).

## IV. Future Work

The paper presents DoEth, our work-in-progress prototype to provide no-overhead fine-grained tracing information of software and hardware interactions on commercial off-the-shelf SoCs to practitioners and system designers. The outlined demonstration aims to showcase, through an interactive scenario, the ability of DoEth to provide live tracking of workloads progress and their run-time resource utilization.

We foresee many avenues for both technical and functional improvements. This includes expanding:

- DoEth to include a tiny re-programmable RISC-V core that will be responsible for programming the SoC's CoreSight and the performance monitoring unit. The objective is to streamline and simplify utilization by reducing the end-user's onboard manipulation.
- The Ethernet support receiving information from the host computer. This will open the possibilities for the host computer to directly communicate and instrument DoEth.
- The set of supported packets and integrates custom packets for information regarding FPGA activity and SoC-wide information such as the SoC's temperature [5].
- Support to more platforms with transceiver (e.g., Kria KR260) and with RJ45 Ethernet (e.g., Kria KV260).

## References

[1] M. Nicolella, S. Roozkhosh, D. Hoornaert, A. Bastoni, and R. Mancuso, "Rt-bench: an extensible benchmark framework for the analysis and management of real-time applications," in *Proceedings of the 30th International Conference on Real-Time Networks and Systems*, ser. RTNS '22. New York, NY, USA: Association for Computing Machinery, 2022, p. 184–195. [Online]. Available: https://doi.org/10.1145/3534879.3534888

[2] [Online]. Available: https://www.lauterbach.com/products/trace-extensions/powertrace-system/powertrace-iii

[3] [Online]. Available: https://www.ghs.com/products/probe.html

[4] Advanced Micro Devices Inc. (AMD), "10g/25g high speed ethernet subsystem product guide (pg210)," https://docs.amd.com/r/en-US/pg210-25g-ethernet/Introduction?tocId=reO6WKaRxFuxiCk6fDpLBQ, 2025, [Accessed 08-10-2025].

[5] ——, "Ultrascale architecture system monitor user guide (ug580)," https://docs.amd.com/v/u/en-US/ug580-ultrascale-sysmon, 2021, [Accessed 08-10-2025].

# RTSS@Work: VecSim, a Vehicular Edge Computing Simulator for Real-Time Applications

Chuanchao Gao, Arvind Easwaran
*College of Computing and Data Science*
*Energy Research Institute @ NTU, Interdisciplinary Graduate Programme*
*Nanyang Technological University, Singapore*
gaoc0008@e.ntu.edu.sg, arvinde@ntu.edu.sg

***Abstract*—This demo presents VecSim, a Vehicular Edge Computing (VEC) simulator designed to evaluate task offloading and resource allocation strategies for real-time applications in VEC. VecSim offers detailed and practical simulation of vehicle mobility, wireless communication in 5G networks, real-time service subscription architecture, and dynamic offloading control, bringing the gap between high-level algorithm design and practical deployment in VEC environments.**

***Index Terms*—Vehicular Edge Computing, Simulator**

## I. Introduction

Vehicular Edge Computing (VEC) has emerged as a promising paradigm to enhance computational efficiency and service quality in intelligent transportation systems. By leveraging advanced wireless technologies such as the ultra-reliable low-latency communication (URLLC) capabilities of 5G, VEC enables vehicles to offload computation-intensive and time-critical tasks (e.g., intelligent traffic management and online navigation) to nearby Roadside Units (RSUs). These RSUs offer both wireless bandwidth for task offloading and computational resources for task execution. By processing tasks on nearby RSUs, VEC reduces reliance on centralized cloud infrastructure and lowers communication latency between vehicles and servers, making it well-suited for latency-sensitive vehicular tasks. However, the limited bandwidth and computational resources available at RSUs necessitate efficient strategies for *task offloading* (determining the RSU for task service deployment) and *resource allocation* (optimizing the allocation of bandwidth and computational resources), while satisfying the system resource and task deadline constraints.

Given VEC's potential, there has been growing research interest in developing effective task offloading and resource allocation strategies. However, deploying and evaluating these strategies on real testbeds is often cost-prohibitive and technically complex, making simulators essential tools for experimental evaluation. Popular edge computing simulators (e.g., iFogSim [1] and EdgeCloudSim [2]) lack support for 5G network simulation, making them unsuitable for modeling online offloading control in VEC, where real-time wireless channel quality feedback is essential. Alternatively, network-focused simulators (e.g., Simu5G [3] and Fogbed [4]) offer fine-grained network modeling, but lacking necessary control logic to support real-time task offloading and joint bandwidth and computational resources optimization.

This work was supported by the MoE Tier-2 grant MOE-T2EP20221-0006 and the MoE Tier-2 grant MOE-T2EP20224-0007.

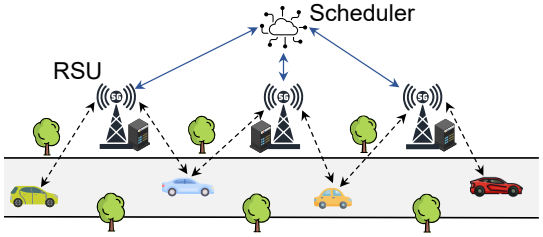



Fig. 1. VEC Simulator Environment

To bridge this gap, we develop VecSim[1], a novel VEC simulator that provides a comprehensive framework for real-time vehicular service subscription and resource scheduling. By integrating real-world data (e.g., traffic trace, task profiling) and 5G-based V2X network modeling, VecSim enables users to evaluate their scheduling strategies in a realistic VEC environment and assess various online offloading control mechanisms based on real-time channel quality feedback.

## II. Features of VecSim

A VEC (Fig. 1) comprises vehicles, RSUs, and a centralized scheduler. Vehicles communicate with RSUs over 5G networks, while RSUs are connected via wired Ethernet to a network switch, which is in turn connected to the scheduler. Each RSU provides wireless bandwidth for task offloading and computational resources for hosting vehicular services. The scheduler continuously monitors system status, including RSU resource utilization and the wireless channel quality between vehicles and RSUs, and periodically determines task offloading and resource allocation for vehicular requests. Given that many vehicular tasks, such as object detection, operate periodically to process continuous sensor data streams, our simulator primarily focuses on periodic vehicular tasks.

When a vehicle intends to subscribe to a vehicular service (e.g., to reduce the computational load of onboard devices or improve task performance), it first sends a request to the scheduler via a nearby RSU. The scheduler periodically invokes the scheduling algorithm to deploy services to vehicular requests. Once a request is scheduled, a grant is sent to the

[1]The source code is availavle at https://github.com/gaochuanchao/mecRT.

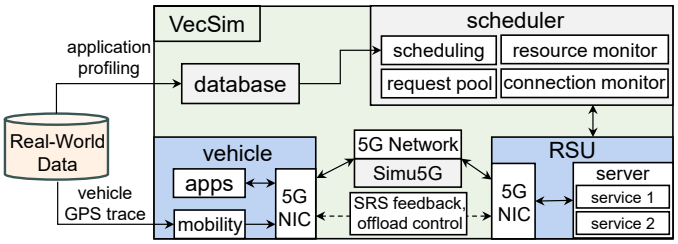



Fig. 2. VEC Simulator Architecture

designated RSU to initiate the vehicular service and enable the offloading of the corresponding task. Meanwhile, when a task is allowed to offload, its vehicle periodically transmits Sounding Reference Signals (SRS) to the RSU hosting its requested service for real-time wireless channel quality estimation. If the channel quality falls below a threshold, the RSU can instruct the vehicle to temporarily suspend task offloading, thereby avoiding potential deadline violations caused by increased transmission delays under poor channel conditions.

The architecture of VecSim is illustrated in Fig. 2. VecSim comprises three major modules: vehicle, RSU, and scheduler. The vehicular module manages vehicular applications, controls vehicle movement, and contains a 5G Network Interface Card (NIC) to simulate practical wireless data offloading in 5G networks. In addition, VecSim enables users to integrate either synthetic vehicle trajectories (e.g., using SUMO [5]) or real-world GPS traces to control vehicle movement. The RSU module comprise a 5G NIC for data offloading control in 5G networks, and a server module for vehicular services deployment based on the instructions from the scheduler.

The scheduler module buffers service requests from vehicles, monitors bandwidth and computational resource usage at RSUs, collects wireless channel quality metrics between vehicles and RSUs, and periodically determines the service deployment for vehicular requests based on the latest system status. To enhance the realism of VEC simulation, VecSim supports the use of profiling data obtained from real-world application execution. When a new scheduling cycle begins, the scheduler retrieves system status data from the system monitor and application profiling data to decide the service deployment and resource allocation for pending requests in the current scheduling cycle.

## III. Demonstration Description

In the demo, we construct a vehicular edge computing simulation environment and demonstrate how to conduct experiments on the simulator with various configurations. The simulation environment utilizes real-world taxi GPS trajectory data [6] to model vehicle mobility, collected in Shanghai on April 1, 2018, by the Shanghai Qiangsheng Taxi Company. Besides, we evaluate four image classification applications (ResNet-152, VGG-16, VGG-19, Inception-v4) and two object detection applications (SSD-Mobilenet-v2, SSD-Inception-v2), profiling their execution on an NVIDIA Jetson Nano (representing onboard devices) and NVIDIA RTX 3090/4090/4500 GPUs (representing RSU servers). The demonstration includes:

- Customize simulation scenario via configuration files.
- Visualization of vehicle mobility and task offloading.
- Analysis of simulation results.

## IV. Contribution

To facilitate reproducible and realistic evaluation of vehicular edge computing strategies, we develop VecSim, an integrated and extensible simulator that models real-time service subscription and resource scheduling in 5G-enabled VEC environments. Built upon the Simu5G framework, VecSim incorporates fine-grained 5G-based V2X communication modeling with dynamic channel quality estimation, enabling the study of online offloading control under realistic wireless conditions. It supports both synthetic and real-world vehicular mobility traces, task profiling from empirical measurements, and flexible configuration of vehicular application parameters such as task period, deadline, and data size. Moreover, VecSim introduces a modular scheduler design that allows researchers to implement and evaluate custom algorithms for joint task offloading and resource allocation with deadline guarantees. By bridging the gap between network-level simulation and application-level resource scheduling, VecSim provides a comprehensive and high-fidelity experimental platform for studying adaptive and real-time resource management in VEC systems.

## V. Conclusion

This demo highlights the capabilities of VecSim in filling the gap between high-level algorithm design and practical deployment. By supporting customizable scheduling algorithms and online offloading control mechanisms, VecSim provides a flexible platform for evaluating task deployment and resource management strategies in practical VEC systems.

## References

[1] H. Gupta, A. Vahid Dastjerdi, S. K. Ghosh, and R. Buyya, "ifogsim: A toolkit for modeling and simulation of resource management techniques in the internet of things, edge and fog computing environments," *Software: Practice and Experience*, vol. 47, no. 9, pp. 1275–1296, 2017.

[2] C. Sonmez, A. Ozgovde, and C. Ersoy, "Edgecloudsim: An environment for performance evaluation of edge computing systems," in *2017 Second International Conference on Fog and Mobile Edge Computing (FMEC)*, 2017, pp. 39–44.

[3] G. Nardini, D. Sabella, G. Stea, P. Thakkar, and A. Virdis, "Simu5g–an omnet++ library for end-to-end performance evaluation of 5g networks," *IEEE Access*, vol. 8, pp. 181 176–181 191, 2020.

[4] A. Coutinho, F. Greve, C. Prazeres, and J. Cardoso, "Fogbed: A rapid-prototyping emulation environment for fog computing," in *2018 IEEE International Conference on Communications (ICC)*, 2018, pp. 1–7.

[5] P. A. Lopez, M. Behrisch, L. Bieker-Walz, J. Erdmann, Y.-P. Flötteröd, R. Hilbrich, L. Lücken, J. Rummel, P. Wagner, and E. Wießner, "Microscopic traffic simulation using sumo," in *The 21st IEEE International Conference on Intelligent Transportation Systems*. IEEE, 2018. [Online]. Available: https://elib.dlr.de/124092/

[6] SODA, "Shanghai qiangsheng taxi gps data trace (2018-04-01)," 2018. [Online]. Available: https://github.com/hetianzhang/Edge-DataSet?tab=readme-ov-file\#taxi-trajectory-data