Cyber-Physical Systems in Production Engineering

“Designing safe, predictable, and high performance embedded platforms for next generation Cyber-Physical Systems.”

Modern Cyber-Physical Systems (CPS) are the next generation of engineered systems in which computing, communication, and control technologies are tightly integrated. Applications include system automation, Internet of Things (IoT), smart buildings, smart manufacturing, smart cities, digital agriculture, robotics, and autonomous vehicles. The Chair of Cyber-Physical Systems in Production Engineering was founded in September 2018. In 2020, research activities of the Chair focused on following topics: a) develop new Reinforcement Learning strategies for CPS, b) design and implement novel resource management policies for embedded real-time systems running on high-performance heterogeneous platforms, and c) design novel deep learning approaches for high precision manufacturing. Other research activities are also focusing on the secure and safe integration of machine learning algorithms with digital controllers for CPS. Members of the chair were involved in the peer review process of several international conferences/journals in real-time embedded systems and CPS, including IROS 2020, ICRA 2021, ICCPS 2020, CDC 2020, RTSS 2020 and ACC 2020, as well as IEEE TC, IEEE TCOM, ACM TCPS and IEEE Access.

Reinforcement Learning for Cyber-Physical Systems

Reinforcement learning (RL) for planning and control of cyber-physical systems is receiving increasing interest for its promise to solve complex optimization problems through interactions with the environment alone. To do so, the RL agent explores possible actions within an environment, receiving a reward. The goal of the agent is to maximize the cumulative reward it earns over time.

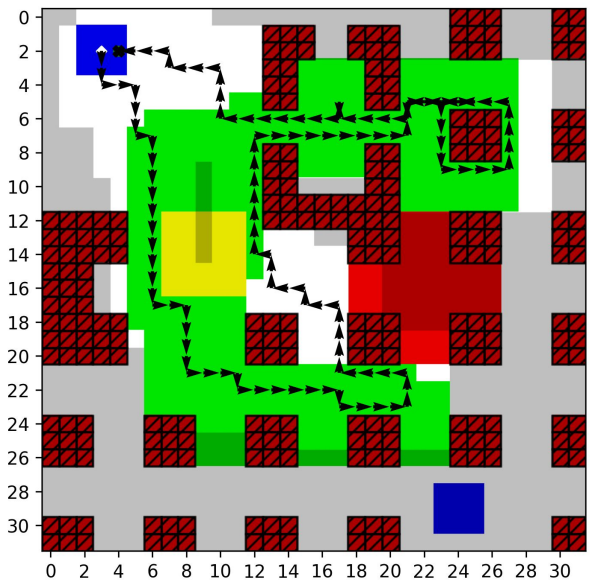
If an RL agent is supposed to be applicable as a path planner for an autonomous vehicle, such as an unmanned aerial vehicle (UAV), it has to be able to generalize over multiple scenarios. To generalize, the agent either has to find one robust solution that solves all possible problems or it needs to adapt to differences in scenario parameters. In path planning, scenarios typically vary significantly such that there exists no one solution for all of them. Consequently, the agent has to adapt its policy to address scenario changes. This is only possible if the agent can observe or infer the varying parameters.

The focus of our research in this field is to identify, develop, and evaluate methods to feed environment information to the RL agent, such that it can generalize over the varying scenario parameters. For this we focus on path planning for UAVs for two applications; coverage path planning (CPP) and wireless data harvesting (DH). In both applications, the agent is navigating a UAV through an urban environment with dedicated start and landing zones, obstacles such as buildings, and regulatory no-fly zones. In CPP the agent is tasked to explore one or more target

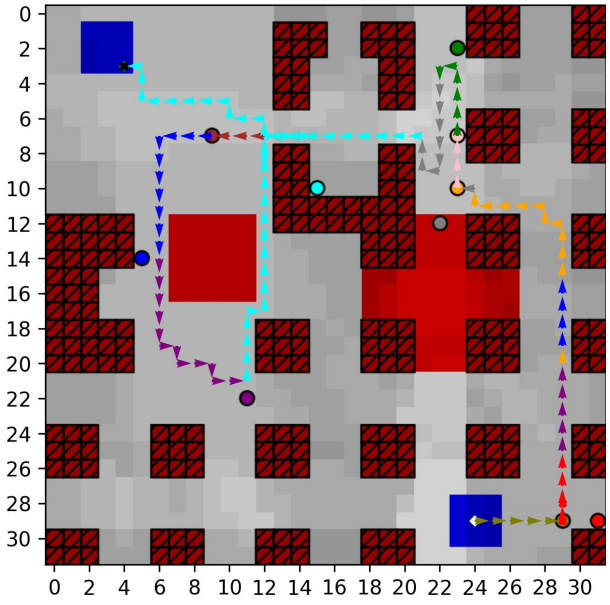
zones within the environment by flying above them, taking images with an onboard camera. The goal is to cover as much of the target area as possible while returning and landing in the dedicated landing zone before the power runs out. In the DH application, the agent is tasked to gather data from distributed Internet-of-Things (IoT) devices that only have short-range communication capability. Therefore, the UAV is navigating the environment, reducing the distance to the IoT devices, gathering as much of their data as possible, while maintaining battery constraints. While the urban environment is static and unchanging from scenario to scenario, the target zone in CPP and location and data of IoT devices in the DH problem are subject to change between scenarios. Therefore, the agent needs to generalize over all possible target zones and device placements in the two problems, respectively.

Our approach for solving these path planning problems is to describe the environment through a map with the map-layers describing start and landing zones, obstacles, no-fly-zones, and target areas or IoT devices. This multi-layer map is fed to an agent that uses convolutional neural network layers to extract relevant features, inferring decisions based on the relative position and remaining power. In our published work, we first show the applicability of the map-based approach for simple CPP problems [1], extending it to the DH application domain in [2]. Furthermore, in [2], we show that centering the map around the agent's position greatly improves its performance and enables the ability to generalize well over different scenarios. In two preprints we show that representing the environment information as a global and a local map significantly improves scalability [3] and that the map-based approach is applicable to multi-agent systems [4]. The python-based training environment was developed as part of this research, it is open-source and it has been made available in [S2].

Some example trajectories from [3] are shown in the following Figures. The first Figure shows a CPP scenario with start and landing zones in blue, obstacles red with black hashing, no-fly zones in red, and target zones in green. The agent's trajectory is indicated with black arrows, starting and ending in the top-left landing zone. The covered area is indicated through increased brightness. This specific target zone was never seen by the agent during training, but the agent managed to cover most of the area while flying back in time avoiding a crash.



The next Figure shows the trajectory of the agent in a DH scenario, where it is tasked to collect the data from the colored IoT devices. The color of the trajectory, starting at the bottom-right and ending in the top-left, indicates which device the agent communicated with. The different shades of gray indicate line-of-sight to the devices. The brighter the cell, the more devices have a line-of-sight



connection to the UAV in this cell. As before, this constellation of IoT devices was not seen during training. The agent managed to collect the majority of the available data while landing before the power ran out.

Future work in this field will focus on bringing the path planning problem closer to the real world by adding continuous actions, more elaborate power models, and stochastic effects in the environment, such as wind. To implement our approach on a real-world UAV, we are further developing a remote learning framework, which allows the agent to be deployed on embedded platforms with constrained performance.

Application of 6D Pose Estimation to create novel deep learning approaches for high precision manufacturing

Manufacturing is experiencing a transition to higher customization and smaller batches. The chair of Cyber-physical Systems in Production Engineering is creating robot automated processes and infrastructure to bridge the gap between the need for flexible automation and the current manufacturing tools, namely the ability to pick and handle new objects with extreme precision in a production or assembly line without the need for robot reprogramming. This is accomplished by using a deep learning approach known as “6D pose recognition” and a dedicated training station to manage the learning of new manufacturing parts.

The 6D Pose Recognition is performed based on a pixel wise voting network [D1] (PVNet), which has several characteristics that make it suitable for high precision manufacturing.

First, the network is a one-shot approach to pose recognition, making its execution predictable and real-time. In our tests, the performance exceeded 20 frames per second on mobile/embedded hardware. In an industrial setting, this is important because variations in performance can result in unexpected delays in the picking or in the handling of workpieces, which would result in disruption of production throughput.

Moreover, due to its internal architecture, PVNet is quite robust when it comes to detecting partially occluded parts. The reasons lie in how the neural network detects the key points of the object: not through the local features near them, but by using the whole object and its geometry to then infer their position. PVNet, much like humans, learns to recognize key point positions (e.g. the tip of a pencil, or the neck of a bottle) in relation to other parts of the object. Thus, it can figure out the position of a pencil tip, even if the pencil is partially occluded.

However, only recognizing key points is not sufficient to estimate the position of the entire object. Thus, after the key points have been estimated, a perspective-n-point (PnP) algorithm is used to determine the position and orientation of the object with respect to the camera. Once this is known, the coordinates are transformed into an absolute frame of reference by using the position of the camera provided by the robot.

This deep learning approach is coupled with our novel training process (see following training station setup). One of the critical processes in the implementation of a flexible automated production line is the introduction of new parts. This requires training the system to recognize the new parts, which can be almost as cumbersome as the traditional manual reprogramming of the robot with the existing technology. The automated training station makes this process vastly more efficient.

Fig 1. Training Station Setup

The physical setup of the training station can be seen in Figure 1. The training station under operation can be seen in Figure 2. The workpiece is placed on top of a monitor which acts as the image acquisition area. During data acquisition, the background image on the monitor can be changed, and the robotic arm will reposition the camera to take images of the workpiece from various angles. To operate the training station, a 3D model of the new part and a physical sample of it are needed. The model is uploaded through the graphical interface of the training station and the part is placed within the image acquisition area.

The process for retraining the network proceeds as follows:

1. The part is positioned in the middle of the image acquisition area.
2. The background monitor cycles through several standard backgrounds while the robot places the camera at a desired distance on top of the object with the optical axes of the camera perpendicular to the scanning area.
3. The object initial position is calibrated using a simple geometrical match with the uploaded 3D model.
4. The robot moves the camera, stopping at several different distances and angles in order to cover multiple points of view.
5. At each point of view, the monitor reproduces a set of backgrounds and, for each of those, an image is acquired.
6. A geometric algorithm computes the appropriate labels for each image using the position of the camera provided by the robot and the initial position of the object.
7. After one data acquisition cycle is completed, the operator can move the part into a different position (e.g. upside down, on the side), and perform a new acquisition cycle.
8. Once all the data acquisition cycles are completed, the PVNet instance is retrained using transfer learning.

Our contributions to date are an open-source HTTP and Django interface to send motion commands and read positional data from the robot[1], which is utilized in our training station[2]. Our current research directions are as follows.

1. Due to the delay between the measurement of robot position and processing of camera data, parallax effects may be observed. The impact of these effects on picking accuracy is currently unknown and warrants further investigation.
2. Current work has been done using an RGB camera and the PVNet architecture. Further investigation will be performed on different data acquisition platforms (such as RGB-D cameras) and neural networks (such as PVNet3D or integrating YOLO), as well as the redesigning and optimization of such networks to run on more constrained embedded hardware such as the OpenCV OAK-D AI Kit, subject to computational, timing, and

[1] https://github.com/seedship/cpsFanucInterface
[2] https://drive.google.com/file/d/1lzPo8f3oDDXA7lNJDzjfWjV3nVY_O3em/view?pli=1

accuracy constraints. We have submitted a proposal to the 2021 OpenCV AI Competition[3] to redesign and optimize PVNet3D to run on an OpenCV OAK-D subject to the requirements of our robot control loop and the OAK-D embedded hardware.

3. Initial prototyping has been done on our novel training station platform. This training station will be used to autonomously create large datasets of RGB and RGB-D pictures of objects with accurate ground truth position data, to augment or replace existing datasets such as 3D LINEMOD, for use in training new 6D Pose estimation networks.

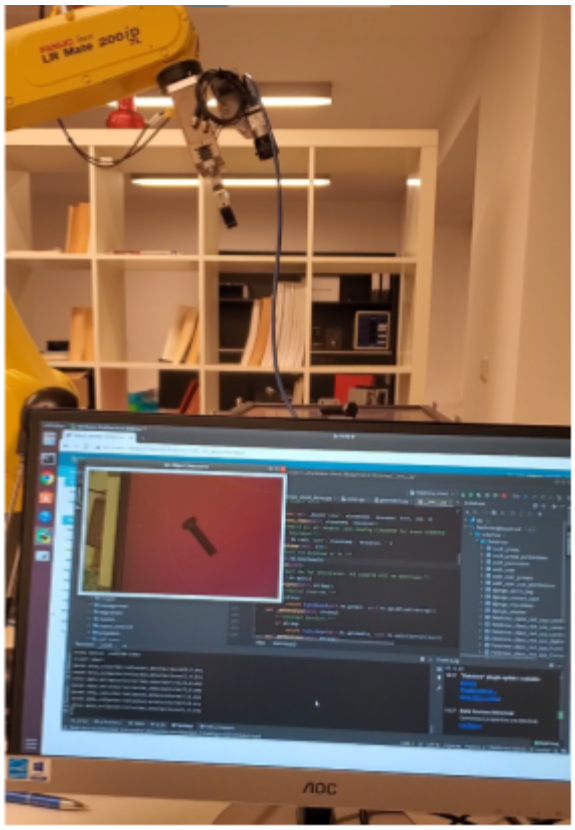


Fig 2. Training Station Operation

With online pose estimates of the workpiece, trajectories for pick-and-place operations can be planned. In complex configuration spaces, this task is non-trivial and falls back to a path planning problem. For solving this path planning problem, reinforcement learning can be utilized. At this point, our robotic research intertwines with our UAV path planning research. Lessons learned and methodologies developed for UAV coverage path planning and path planning for data harvesting can be applied to address challenges in robotic path planning. Specifically, the method of feeding known environmental parameters, such as obstacles, to the reinforcement learning agent can be inspired by the map-based approach of UAV path planning.

External Reference

[D1] Peng, S., Liu, Y., Huang, Q., Bao, H., & Zhou, X. (2018). PVNet: Pixel-Wise Voting Network for 6DoF Pose Estimation. 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 4556-4565.

[3] https://opencv.org/opencv-ai-competition-2021/

Predictable and high-performance resource management of CPS on heterogeneous platforms

The widespread use of artificial-intelligence (AI) algorithms in many Cyber-Physical Systems (CPS) such as autonomous cars, drones, and smart robots has driven the integration of specialized hardware accelerators (e.g., GPUs, FPGAs) on high-performance multiprocessor boards. The need to deploy these algorithms "online" has stimulated the rapid evolution of such heterogeneous multiprocessor systems-on-chips (MPSoC) also in safety-critical embedded contexts that were traditionally dominated by much simpler single-processor platforms. Current heterogeneous MPSoCs, specifically developed for the embedded market (e.g., Xilinx Ultrascale+, NVIDIA Xavier) offer different types of heterogeneous multicore CPU clusters as well as GPUs and/or FPGAs. Towards ensuring safety and real-time requirements, these platforms pose unprecedented challenges.

On the one hand, the implementation of complex CPS using these platforms generates increasing volumes of real-time (e.g., imaging) data flows causing the hardware memory hierarchy (the DRAM, the interconnect, and the cache hierarchy, especially the last level cache shared among multiple cores) to become a bottleneck and a source of temporal unpredictability. This phenomenon is further aggravated by the presence of accelerators (GPUs/FPGAs) that can independently access memory with high-bandwidth requests. On the other hand, traditional techniques to allocate and optimize the execution of real-time tasks on safety critical CPS do not consider the heterogeneity of the computing elements and the complexity of MPSoCs' memory hierarchy. In addition, classical task models widely adopted in the real-time scheduling domain fail to capture the parallelization and heterogeneous computing needs of the new workloads.

At the integration level, the adoption of isolation techniques for MPSoCs is hindered by the large space of configuration possibilities. Developers and integrators are daunted by the task of finding the right trade-offs between selecting the appropriate scheduling policies, assigning real-time tasks to (heterogeneous) cores, selecting the size of cache partitions, and determining adequate bandwidth to allocate to each communicating resource. Our work tackles these challenges with techniques that could be rapidly adopted by the industry, and that aim to practically simplify the deployment of real-time workload on MPSoC without sacrificing neither predictability nor performance. On the platform side, we research, develop, and evaluate techniques to restore isolation and temporal predictability of safety critical software. We specifically target solutions (e.g., [A1], [A2], [A3]) that prove to perform well "in practice", and we focus our integration effort at both Operating System (OS) and Hypervisor levels. Hypervisors (e.g., [A4], [A5], [A6], [A7], [A8], [A9]) have become the de-facto industry standard to ensure isolation in certified partitioned safety-critical systems, but do not provide satisfactory isolation and predictability properties when contention at cache, interconnect, and DRAM level is

considered. To facilitate the evaluation and the adoption of these techniques by the industry, in addition to publications (e.g., [5], [9], [10]), we actively participate in the development of open source hypervisors (e.g., [A4], [A5]) to make the developed techniques not only readily available, but also supported by an active community [S1]. On these topics, we also actively collaborate with highly skilled international teams [A10], [A11], which pursue objectives akin to ours.

On the application and deployment side, our research considers optimization and scheduling techniques to hide the complexity of the configuration space from the integrators, while enforcing isolation (enacted via the above-mentioned low level solutions) and ensuring the real-time properties of the workload. Problems under analysis are: the optimization of real-time task allocation under simultaneous consideration of cache-size, core, and bandwidth constraints; scheduling of real-time resources with different quality-of-service guarantees to applications with multiple criticality levels (e.g., [6]); analysis of expressive real-time task models to support parallel computation models (e.g., [7], [A12]); analysis and design of real-time resource access protocols (e.g., [8]); analysis and applicability of the predictable execution model [A13] on heterogeneous MPSoCs (e.g., [11]).

In the remainder of this section, we present more details on our recent works [5] and [6]. These papers are representative of our research efforts in 2020 towards predictable and high-performance resource management of CPS on heterogeneous platforms.

[5]: Fixed-Priority Memory-Centric Scheduler for COTS-Based Multiprocessors

In commercial-off-the-shelf (COTS) multiprocessor systems, a task's execution time can increase by an order of magnitude due to shared main memory interference that is generated by tasks running simultaneously on other cores [B1]. Bounding or even eliminating this interference is highly desirable in real-time systems. The PRedictable Execution Model (PREM) [B2] and its extensions [B3], [B4], [B5] attempt to mitigate the memory interference problem by splitting a task's execution into separate phases for memory transactions and pure computation. Based on PREM, memory transactions of all cores can be scheduled to a sequence (memory-centric scheduling), which allows for the elimination of memory interference.

Many successful implementations of memory-centric schedulers [B2], [B5], [B6] have adopted time-division multiplexing (TDM or static fair-share) as the underlying scheduling principle. TDM-based arbitration distributes a resource (in this case main memory access slots) in a very predictable way. One known downside of this approach is the underutilization of the resource: if a processor has a reserved time slot but does not use it, the slot cannot be offered to another processor [B7].

To address these shortcomings and to enable a contentionless memory scheduling, we (1) designed a fixed-priority preemptive memory-centric scheduler for modern COTS multiprocessor

systems, (2) implemented the memory-centric scheduler within a hypervisor (Jailhouse [A4]) and a real-time operating system (Erika [B8]) that are based entirely on open-source software components, (3) evaluated the effectiveness and the limitations of the proposed approach with a set of microbenchmarks and real-world workloads, (4) provided the fixed-priority partitioned multiprocessor schedulability analysis for PREM-compliant task sets backed by schedulability experiments, which incorporate the actual memory arbitration overheads in the analytical model, and (5) identified a heuristic for task partitioning.

Measurements of the scheduler overhead demonstrate the applicability of the proposed approach, and schedulability experiments show a 20% gain in terms of schedulability when compared to contention-based and static fair-share approaches. Our solution does not rely on hardware features for predictable memory management of any kind, which reflects the situation for many COTS multiprocessor platforms today. The fixed-priority preemptive memory-centric scheduler preserves predictable execution, while reducing the worst-case response time for high priority tasks. Our implementation also shows that it is possible to achieve dynamic memory arbitration at the hypervisor level with relatively small overheads. Moreover, to the best of our knowledge, we are the first to propose a task-to-processor assignment algorithm integrated with PREM.

[6]: GoodSpread: Criticality-Aware Static Scheduling of CPS with Multi-QoS Resources

For safety-critical CPS, the control software needs to provide performance guarantees. Besides the control algorithm, the performance also depends on how the software uses the platform resources (e.g., computation, communication, and memory resources) [C1], [C2]. The software implementation influences the control timings (e.g., sampling period and closed-loop delay) that have a significant impact on the dynamics of the physical process. In practice, to offer maximum performance in all scenarios, the safety-critical control software often uses resources with high quality-of-service (QoS) (e.g., time-triggered resources) [C3].

Typically, a controlled process exhibits multiple criticality levels based on its physical state [C4]. For example, when the plant is in a steady state, the control software is in the low-criticality mode where delayed or infrequent control actions will not jeopardize the system's safety. Conversely, when there is a disturbance, the system must reject it within a specified time limit, and therefore, the software must run in the high-criticality mode with hard timing guarantees. Hence, with changing criticality level, the resource requirements of the control software change.

In many CPS domains, hardware platforms comprise resources with different QoS [C5]. For example, using a scratchpad for predictable and faster execution of software code has become a common practice to mitigate large variation in memory access times involving cache [C6].

Previous works have shown that criticality-aware implementation of control applications using heterogeneous resources enables prudent usage of high-QoS resources [C7]. This is important in cost-sensitive yet safety-critical domains like automotive where limited high-QoS resources

are available but in high demand. In a criticality-aware implementation, the controller mostly uses low-QoS resources and switches to high-QoS resources only for a minimal time while experiencing disturbances.

In the context of criticality-aware scheduling of CPS using multi-QoS resources, in [6], for the first time, we evaluate the potential of *static* schemes. We propose a multi-stage optimization framework *GoodSpread* that statically allocates heterogeneous resources to a set of control applications so that each of them meets its performance requirements in all scenarios while the usage of high-QoS resources is minimized. Our results suggest that GoodSpread *saves more than 50%* high-QoS resources in certain cases compared to existing schemes. Besides, static schemes are easier to certify and have no implementation overheads compared to dynamic schemes.

Considering that we can provision either the high- or the low-QoS resource for each control instance, there are exponentially many scheduling possibilities. Experiments show that a control application, along with its control performance objectives and disturbance arrival patterns, can be characterized by a *spread factor*. This factor quantifies how the high-QoS resources should be spread over time in order to accommodate the uncertainty in how the criticality level of the system might change. GoodSpread uses a *polynomial-time* algorithm that studies the closed-loop dynamics to derive the spread factor for an application. In the example of a DC motor position control system (illustrated in Fig. 1), the algorithm determines that allocating high-QoS resources once every four control instances is sufficient to meet the settling time requirement of 0.24s.

Given the spread factor for each control application, GoodSpread formulates an optimization problem to determine a concrete static schedule for the applications sharing the resources. As the main goal here is to prudently use the high-QoS resources, GoodSpread *maximizes the extensibility* of the schedule, i.e., the obtained schedule offers the maximum flexibility in accommodating future applications by using the remaining resources. Here, we propose an iterative approach that incrementally adds prospective applications with maximum resource requirements and co-schedules them with the given set. Such an approach guarantees maximum extensibility considering the premise, i.e., if the schedule can accommodate an application with a higher scheduling demand then it can also accommodate an application with a lower demand.

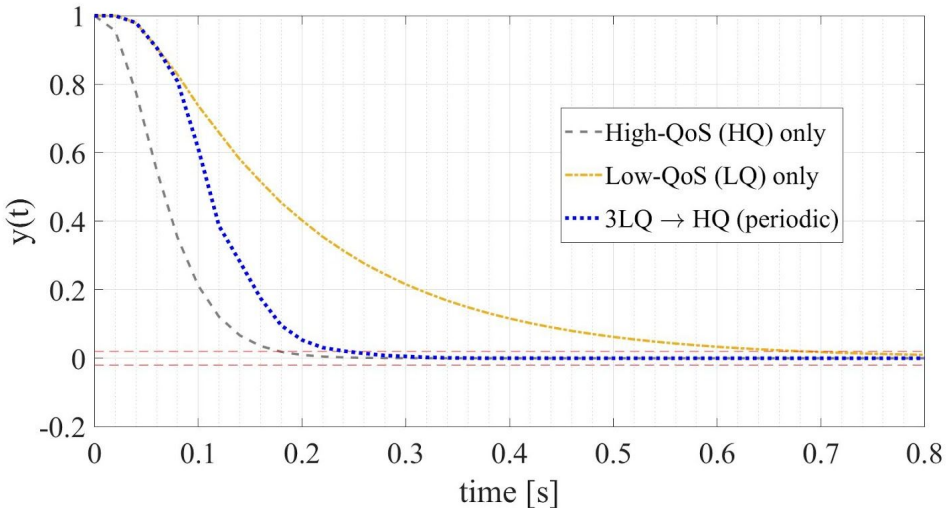


Fig 1: Control response for different resource allocation strategies. When high-QoS resources are allocated periodically once in every four control instances, the settling time is improved from 400% (0.7s) to 133% (0.24s) of that obtained using high-QoS resources exclusively (0.18s).

External References

[A1] H. Yun, G. Yao, R. Pellizzoni, M. Caccamo, L. Sha, "Memory Bandwidth Management for Efficient Performance Isolation in Multi-core Platforms", in IEEE Transactions on Computers, vol. 65, no. 2, pp. 562-576, 1 Feb. 2016

[A2] H. Yun, R. Mancuso, Z.P. Wu, and R. Pellizzoni. "PALLOC: DRAM bank-aware memory allocator for performance isolation on multicore platforms". in Proceedings of 2014 IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)

[A3] T. Kloda, M. Solieri, R. Mancuso, N. Capodieci, P. Valente and M. Bertogna, "Deterministic Memory Hierarchy and Virtualization for Modern Multi-Core Embedded Systems," in Proceedings of 2019 IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)

[A4] Jailhouse hypervisor. Siemens AG on GitHub, https://github.com/siemens/

[A5] Bao hypervisor. https://github.com/bao-project/bao-hypervisor

[A6] ACRN hypervisor. The Linux Foundation Projects. https://projectacrn.org/

[A7] PikeOS hypervisor. https://www.sysgo.com

[A8] GHS Integrity. https://www.ghs.com/products/rtos/integrity_virtualization.html

[A9] ETAS RTA Hypervisor. https://www.etas.com/en/products/rta-vrte.php

[A10] Renato Mancuso. Boston University. http://www.bu.edu/cs/profiles/renato-mancuso/

[A11] HiPeRT Lab. https://hipert.unimore.it

[A12] H. Zahaf, G. Lipari, M. Bertogna and P. Boulet, "The Parallel Multi-Mode Digraph Task Model for Energy-Aware Real-Time Heterogeneous Multi-Core Systems," in IEEE Transactions on Computers, vol. 68, no. 10, pp. 1511-1524, 1 Oct. 2019

[A13] R. Pellizzoni, E. Betti, S. Bak, G. Yao, J. Criswell, M. Caccamo, R. Kegley, "A Predictable Execution Model for COTS-based Embedded Systems", in Proceedings of 2011 IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)

[B1] Kim, H., De Niz, D., Andersson, B., Klein, M., Mutlu, O., & Rajkumar, R. (2014, April). Bounding memory interference delay in COTS-based multi-core systems. In 2014 IEEE 19th Real-Time and Embedded Technology and Applications Symposium (RTAS) (pp. 145-154). IEEE.
[B2] Pellizzoni, R., Betti, E., Bak, S., Yao, G., Criswell, J., Caccamo, M., & Kegley, R. (2011, April). A predictable execution model for COTS-based embedded systems. In 2011 17th IEEE Real-Time and Embedded Technology and Applications Symposium (pp. 269-279). IEEE.
[B3] Burgio, P., Marongiu, A., Valente, P., & Bertogna, M. (2015, October). A memory-centric approach to enable timing-predictability within embedded many-core accelerators. In 2015 CSI Symposium on Real-Time and Embedded Systems and Technologies (RTEST) (pp. 1-8). IEEE.
[B4] Becker, M., Dasari, D., Nicolic, B., Akesson, B., Nélis, V., & Nolte, T. (2016, July). Contention-free execution of automotive applications on a clustered many-core platform. In 2016 28th Euromicro Conference on Real-Time Systems (ECRTS) (pp. 14-24). IEEE.
[B5] Wasly, S., & Pellizzoni, R. (2014, April). Hiding memory latency using fixed priority scheduling. In 2014 IEEE 19th Real-Time and Embedded Technology and Applications Symposium (RTAS) (pp. 75-86). IEEE.
[B6] Tabish, R., Mancuso, R., Wasly, S., Alhammad, A., Phatak, S. S., Pellizzoni, R., & Caccamo, M. (2016, April). A real-time scratchpad-centric OS for multi-core embedded systems. In 2016 IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS) (pp. 1-11). IEEE.
[B7] Schoeberl, M., Pezzarossa, L., & Sparsø, J. (2018). A multicore processor for time-critical applications. IEEE Design & Test, 35(2), 38-47.
[B8] Erika Enterprise (Evidence) - http://www.erika-enterprise.com

[C1] D. Seto, J. P. Lehoczky, L. Sha and K. G. Shin, "On task schedulability in real-time control systems," Real-Time Systems Symposium, 1996.
[C2] W. Chang, L. Zhang, D. Roy, and S. Chakraborty, “Control/architecture codesign for cyber-physical systems, ser. Handbook of Hardware/Software Codesign. Springer Netherlands, 2017.
[C3] F. Simonot-Lion, “In car embedded electronic architectures: How to ensure their safety,” IFAC International Conference on Fieldbus Systems and their Applications (FET), 2003.
[C4] P. Marti, J. M. Fuertes, G. Fohler, and K. Ramamritham, “Improving quality-of-control using flexible timing constraints: Metric and scheduling,” Real-Time Systems Symposium (RTSS), 2002.
[C5] N. Navet and F. Simonot-Lion, “Automotive Embedded Systems Hand-book,” CRC Press, 2009.
[C6] H. Faik and J. C. Kleinsorge, “Optimal static WCET-aware scratchpad allocation of program code,” Design Automation Conference (DAC), 2009.
[C7] D. Roy, W. Chang, S. K. Mitter, and S. Chakraborty, “Tighter dimensioning of heterogeneous multi-resource autonomous CPS with control performance guarantees,” Design Automation Conference (DAC), 2019.

Cyber-Physical Systems in Production Engineering

Prof. Dr. Marco Caccamo

Contact
www.mw.tum.de/cps
mcaccamo@tum.de
Tel: +49.89.289.55170

Management
Prof. Dr. Marco Caccamo, Director

Administrative Staff
Anke Harisch, Secretary

Research Scientists
Tomasz Kloda, Dr.
Ohchul Kwon, Dr.
Andrea Bastoni, Dr.
Debayan Roy, Dr.
Mirco Theile, M.Sc.
Bingzhuo Zhong, M.Sc.
Gero Schwäricke, M.Sc.
Or Dantsker, M.Sc.
Denis Hoornaert, M.Sc.
Jiyang Chen, M.Sc.

Hongpeng Cao, M.Eng.
Jorge Luis Martinez Garcia, M.Sc.
Richard Nai (MS student)
Andrei Goncharov (MS student)

Research Focus
Safety-critical cyber-physical systems
Real-time systems
Scheduling and schedulability analysis
Secure and safe integration of machine learning with CPS
Reinforcement Learning for CPS

Competence
System-level programming
Embedded system software design
Hardware and software integration on FPGAs
Real-time operating systems
Real-time reachability analysis
Reinforcement Learning for CPS

Infrastructure
3 DOF helicopter
Embedded and FPGA multi-core development platforms
High-performance server
Linear inverted pendulum
Fused filament fabrication, dual-head 3D printer
F1/10 autonomous cars

Collaborations
University of Illinois at Urbana-Champaign, USA
Boston University, USA
University of Colorado Boulder, USA
University of Waterloo, Canada
University of Modena, Italy
Federal University of Santa Catarina, Brazil
EURECOM, Sophia Antipolis, France

Lectures

- Advanced Seminar on Safe Cyber-Physical Systems
- PhD-Seminar on Real-Time Cyber-Physical Systems
- Concepts and Software Design for Cyber-Physical Systems
- Tutorial Concepts and Software Design for Cyber-Physical Systems
- Simplex: Fault-Tolerant Control Strategy for Real-Time Cyber-Physical Systems - Laboratory
- Cyber-Physical Systems Lab: Autonomous Applications

Humboldt Sponsored Research:

Selected Publications 2020

[1] M. Theile, H. Bayerlein, R. Nai, D. Gesbert, and M. Caccamo, “UAV coverage path planning under varying power constraints using deep reinforcement learning,” in IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2020.

[2] H. Bayerlein, M. Theile, M. Caccamo, and D. Gesbert, “UAV path planning for wireless data harvesting: A deep reinforcement learning approach,” in IEEE Global Communications Conference (GLOBECOM), 2020.

[3] M. Theile, H. Bayerlein, R. Nai, D. Gesbert, and M. Caccamo, “UAV path planning using global and local map information with deep reinforcement learning,” arXiv:2010.06917 [cs.RO], 2020.

[4] H. Bayerlein, M. Theile, M. Caccamo, and D. Gesbert, “Multi-UAV Path Planning for Wireless Data Harvesting with Deep Reinforcement Learning,” arXiv:2010.12461 [cs.MA]), 2020.

[5] G. Schwäricke, T. Kloda, G. Gracioli, M. Bertogna, and M. Caccamo, "Fixed-Priority Memory-Centric Scheduler for COTS-Based Multiprocessors", in Proceedings of 2020 Euromicro Conference on Real-Time Systems (ECRTS 2020)

[6] D. Roy, S. Ghosh, Q. Zhu, M. Caccamo, and S. Chakraborty, "GoodSpread: Criticality-Aware Static Scheduling of CPS with Multi-QoS Resources", in Proceedings of the IEEE Real-Time Systems Symposium, December 2020.

[7] M. Verucchi, M. Theile, M. Caccamo, and M. Bertogna, "Latency-Aware Generation of Single-Rate DAGs from Multi-Rate Task Sets", in Proceedings of 2020 IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)

[8] L. M. dos Santos, G. Gracioli, T. Kloda, and M. Caccamo, "On the Design and Implementation of Real-Time Resource Access Protocols",Proceedings of the X Brazilian Symposium on Computing Systems Engineering, November 2020.

[9] A. Bansal, J. Singh, Yifan Hao, J. Wen, R. Mancuso, and M. Caccamo, "Reconciling Predictability and Coherent Caching", Proceedings of IEEE 9th Mediterranean Conference on Embedded Computing (MECO), June 2020.

[10] R. Tabish, JY. Wen, R. Pellizzoni, R. Mancuso, H. Yun, M. Caccamo, L. Sha, "SCE-Comm: A Real-Time Inter-Core Communication Framework for Strictly Partitioned Multi-core Processors", Proceedings of IEEE 9th Mediterranean Conference on Embedded Computing (MECO), June 2020.

Selected Publications 2019

[11] M. R. Soliman, G. Gracioli, R. Tabish, R. Pellizzoni, and M. Caccamo, "Segment Streaming for the Three-Phase Execution Model: Design and Implementation", in Proceedings of 2019 IEEE Real-Time Systems Symposium

Produced Software

[S1] Jailhouse Cache-coloring. Jailhouse ML. https://groups.google.com/g/jailhouse-dev/c/K4rqZxpxa0U
[S2] uavSim - Reinforcement learning for path planning of UAVs https://github.com/theilem/uavSim.git
[S3] cpsFanucInterface - Interface to command FANUC robots over HTTP or socket messaging to a Django command server https://github.com/seedship/cpsFanucInterface