Towards a new task model for merging control theory and real-time scheduling problems

Ismail Hawila* †, Liliana Cucu-Grosjean*, Slim Ben Amor†, and Yves Sorel *
*Kopernic, Inria, France, Email: firstname.lastname@inria.fr
†StatInf, France, Email: firstname.lastname@statinf.fr

***Abstract*—Existing task models for cyber-physical systems have been proposed to merge the requirements associated to the stability of control tasks managing physical components, and to the schedulability of these tasks. Nevertheless, these models stay to a high level without covering concerns like the data and/or precedence constraints between different control tasks as well as non-control tasks, the impact of period variation between dependent control tasks and the impact of data constraints on the execution times of control tasks. Our current work concentrates on discussing properties of a new task model in order to cover these concerns.**

I. Motivation of our problem

Our every day life is improved by the presence of *a new generation of systems with integrated computational and physical capabilities that can interact with humans through many new modalities* [1]. These systems are called cyber-physical systems (CPSs). A defibrillator, a mobile phone, an autonomous car or an aircraft, are all examples of CPSs. Beside constraints like power consumption, security, correctness, size and weight, CPSs may have cyber components required to fulfill their functions within a limited execution time interval, often imposed by the physical environment, which is also known as real-time constraints.

Concretely, the CPS design implies the implementation of controllers as periodic programs that have real-time constraints to meet. Usually, the programs are modeled as real-time tasks and the associated CPS task model is considered more general than the classical real-time task model as defined by Liu and Layland in [2]. Recent results [3] have underlined that there is a gap between the CPS *stability* as defined within a control theory context and the schedulability as defined within the real-time scheduling community and the authors have proposed a new model to fill this gap. Nevertheless, when considering the co-design problem there are no dependencies considered between the controller tasks while they may share the same microcontroller for their execution [4]–[6]. More precisely, in the remainder of this paper, we understand by *controller*, a cyber component algorithm deciding what actions must be taken based on sensor readings or input data. The output of the controller is the input to the physical component of the system, usually called *plant*, which forces the output of the plant to follow a desired setpoint [7]. This implies that a *control task* implements the controller's algorithm, such that computing the output of the controller is done in this task. Moreover, by stability we mean that the output of the CPS is under control, following the desired setpoint.

A simple real-time constraint to be fulfilled by a control task is to have its worst-case execution time (WCET) smaller than the period of its execution. Often, the real-time community has studied the impact of the period variation on the stability of the controller, but, to the best of our knowledge, no effort has been dedicated to the impact of the variation of values for variables describing external events on the variation of execution time of a control task. This limitation may come from the fact that, usually, the real-time control problems are studied by the scheduling community and much less by the WCET community.

With respect to the proposition of a new task model merging scheduling and WCET concerns for real-time control systems, our purpose is that this new task model fulfills the following expectations:

1) the data and/or precedence constraints between different control tasks as well as non-control tasks are considered;
2) the impact of data constraints on the execution times of control tasks is considered;
3) the impact of period variation between dependent control tasks is considered.

Our contribution concerns underlining limitations of existing work and the presentation of preliminary results in favor of the proposition of a new task model. In Section II, we present a use case of an open source measurement-based benchmark that we use to support our discussion. We detail our discussion in Section III by considering existing work and its limitations. In the last section of the paper, we conclude by presenting a possible work plan towards the proposition of a new real-time control task model.

II. Considered use case: KDBench

Our discussion on a potential new task model is supported with examples based on KDBench [8][1]. This is a measurement-based benchmark which is defined as a 4-uple $(A, p, \mathcal{M}, c(A))$ composed by a program A, a microcontroller p, a measurement protocol $\mathcal{M}$ and an ordered sequence of execution times $c(A)$ of program A on the microcontroller p. For the program A, one provides the source code as well as the binary code. A measurement protocol $\mathcal{M}$ is defined by the variation of the input variables (associated to sensors) of these benchmarks. In our case, the variation of the input

[1]More details on the KDBench are available at https://team.inria.fr/kopernic/kdbench/

variables is obtained by collecting them during a simulated flight of a drone. The fourth element is an ordered set of execution times, $c(A)$, proposed to overcome the difficulty of the reproducibility of results [9] while comparing execution times measured for the same program on slightly different microcontroller configurations. Moreover, we use execution times measured at the scheduler level in order to decrease the intrusion of the measurement protocol. Such measurements are said to be obtained in a hardware-in-the-loop manner. We understand by hardware-in-the-loop that the execution of the benchmarks is done on a microcontroller while sensors and actuators of a CPS, as well as its physical environment, are simulated. Indeed, our community does not often provide numerical results for programs executed on microcontrollers because of the important effort of implementation required for such execution, or the lack of access to these microcontrollers. We believe that this limitation prevents our community from proposing realistic models describing the impact of existing microcontrollers on the execution time variation of programs and thus our community proposes results that may be not realistic w.r.t. the microcontrollers used by the real-time industry.

In this paper, we put the basis towards a new task model and discussing our proposal with respect to a real execution platform composed of a microcontroller, sensors and actuators.

III. Existing work and discussion on a new task model

In this paper we consider the programs or real-time tasks to be scheduled on a single-core microcontroller with no hardware accelerators. The scheduling algorithm is a preemptive fixed-priority algorithm where the priorities are given, they cannot be modified during the execution of the system.

In this section, we propose a discussion on potential means to describe a real-time control task set and one typical design choice is the use of Directed Acyclic Graphs (DAGs) to represent data and/or precedence constraints between tasks.

We understand by a precedence constraint that data is provided by a task for its successor and that the successor task cannot be executed before the predecessor task, and this is ensured in the KDBench programs by the choice of periods and priorities. We understand by data constraints that the successor task requires some data from the predecessor task, but this does not require the successor to wait for new data instead it can execute with the latest available data. In data constraints a task can be faster than its predecessor, while in precedence constraints case the task set is not schedulable. Both types of constraints are depicted by edges of the DAG.

A. Using DAG-based model

Within the KDBench benchmark, the set of programs are those of the open-source drone autopilot *PX4-RT*, where three programs are control programs (or tasks) called: *Position*, *Attitude* and *Rate ctrl*. In Figure 1, we provide the KDBench programs and their two types of constraints.

In Figure 2, we provide a possible DAG representation of data and precedence constraints as described in Figure 1,

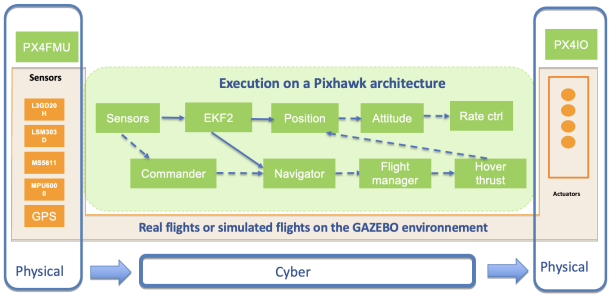

Fig. 1. A DAG describing constraints between KDBench programs

where programs that are not control ones are represented in green. Indeed, we represent by continuous lines edges that depict precedence constraints between tasks and by dotted lines edges that depict the data constraints.

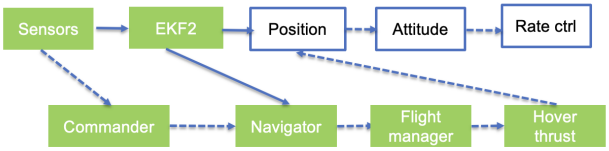

Fig. 2. Data and precedence constraints represented by a DAG-based task model for KDBench programs

Existing work has included such DAG-based models but they do not consider several dependent control tasks.

1) In [5] a set of independent control applications each corresponding to a different plant are scheduled on a single-core microcontroller, where each control application is represented by a DAG model. Whereas in our case we want to model the constraints between tasks, that are scheduled on a single-core microcontroller, and that contribute to the control of a unique plant.
2) Different works as in [6], [10] targeted the control scheduling co-design, where the assignment of periods and priorities of multiple control applications running on the same microcontroller is done considering some control performance metrics. However, these control applications are considered independent from each other.

B. The need for a model with different periods for different tasks

KDBench programs have different periods for different control tasks. For instance, *Rate ctrl* has a smaller period than *Attitude*, while *Attitude* has a smaller or equal period than *Position*. The reason is that attitude dynamics is "faster" than the translational position dynamics, meaning attitude and angular rates should be faster to ensure vehicle stability. One possible solution is to allow different periods for tasks having precedence or data constraints. For instance, in Figure 3 we provide one example of programs periods, where simulated flights are stable.

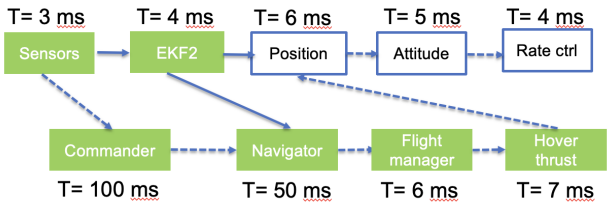

Fig. 3. Different periods for a DAG-based task model of KDBench programs

Existing work includes such DAG-based models but, to the best of our knowledge these results are not provided for (control or not) tasks with different periods.

1) the most common DAG-based model associates to the entire set of tasks a unique DAG to describe constraints between tasks [11]–[13];
2) recent results have been provided for several DAGs but each DAG describes a task and no constraints between tasks is considered [5], [14].

C. The impact of data constraints on the execution times of control tasks is considered

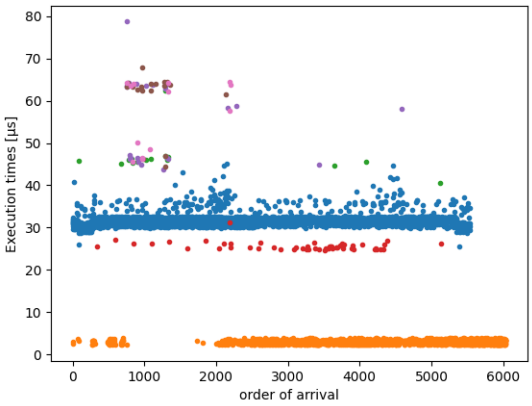

Fig. 4. Execution time of rate control tasks clustered by paths

The data received by a task at the beginning of each job may have an impact on the path executed within the program. In Figure 4 we present a sequence of 6000 execution times for the *Rate ctrl* task. These execution times have been recorded during a flight of 60 seconds. We have identified by the same colour execution times obtained by executing almost the same path within the control flow graph (CFG) of this task. We understand by a CFG of a program or task a directed graph representing all paths that may be taken during the execution of that program. A node within a CFG represents a basic block and a directed edge represents a jump transition from one "basic block" to another. A CFG may be built from the source code of a program or from its binary code and it is, often, used for the WCET estimation of a program on a microcontroller. We understand by a basic block a maximal sequential set of instructions within a program.

In order to illustrate the impact of received data on the executed path and execution time of tasks, let us consider 2 tasks (τ_1, τ_2) with precedence constraints, where τ_2 consumes data x produced by τ_1. Assuming also that the CFG of Figure 5 corresponds to τ_2, then depending on the value of x received by τ_2, one may decide to execute the block B or the block C. If the execution time of blocks B and C are significantly different this has an impact on the total execution time of the concerned job of τ_2. This is somehow what is happening for the *Rate ctrl* task, so depending on the data which is the setpoint received from *Atitude* task the path taken in the CFG is affected leading to the variation of execution times we see in Figure 4.

In order to illustrate the notion of *similar paths*, let us reconsider the two paths within the CFG of the task illustrated in Figure 5. In our case, we say that these two paths are almost the same if they have a majority of common instructions, e.g., if less than 15% of instructions are different. For instance, for the paths ABD and ACD in Figure 5 we say that they are almost the same if at least 85% of instructions between the two paths are the same, while the order of instructions is considered.

Thus, in Figure 4 we identify, at least, 5 different paths that are executed during the flight. These different paths are executed due to the variation of some input variables during the flight. We may notice that, for instance, the paths in blue (middle part of the figure) and the paths in orange (lower part of the figure), that have considerable number of instructions difference, have different execution times. To the best of our knowledge, there is no existing task model considering the relation between the path variation as a mean to adapt the period or the scheduler choice in order to improve the schedulability and the stability of the entire CPS.

In the literature, two paths are considered as identical if they contain the same ordered sequence of instructions. Otherwise, they are considered to be different. A program may have an extremely important number of paths (more than 4000 paths for the Sensors program). For this reason we propose this notion of *similar paths*.

For example, when generating a figure such as Figure 4, we have faced the difficulty of using an important number of colour to distinguish among paths. By introducing *similar paths*, we have decreased this number and make the figure more readable. We may notice that the definition seems promising as execution times of similar paths are grouped within the same horizontal clusters of execution times.

IV. Conclusion

In this paper we discuss new task models for cyber-physical systems merging the requirements associated to the stability of control tasks managing physical components, and to the schedulability of these tasks. Our discussion includes data and/or precedence constraints between different control tasks as well as non-control tasks, the impact of period variation

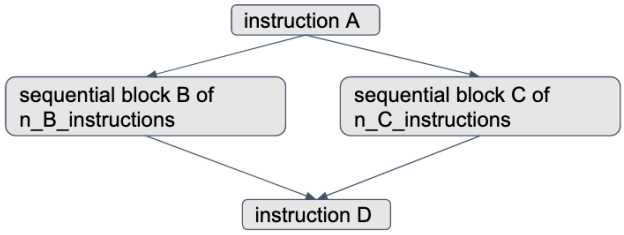

Fig. 5. Example of a CFG with two different paths

between dependent control tasks and the impact of data constraints on execution times of control tasks. Moreover, we underline limitations of existing work and we list mandatory properties that a new task model should cover. Our discussion is based on an open source measurement-based benchmark, KDBench.

Recent existing work has concentrated on the sensitivity analysis of periods for control tasks in order to improve the schedulability and the stability of the entire CPS. The main difficulty comes from merging information on periods (important for the stability problem) and execution times (important for the schedulability problem). Our future work will consider probabilistic distributions describing periods and execution times as a way to combine control tasks and non-control tasks. While probabilities are naturally associated to execution times by WCET statistical estimators, we identify the difficulty of associating probabilities to periods, because periods are imposed by the sensors design and their variations could be necessary when physical environment has important changes.

REFERENCES

[1] Radhakisan Baheti and Helen Gill. *Cyber-physical systems*. IEEE, 2011.

[2] Chung Laung Liu and James W. Layland. Scheduling algorithms for multiprogramming in a hard-real-time environment. *J. ACM*, 20(1):46–61, 1973.

[3] Hoon Sung Chwa, Kang G. Shin, and Jinkyu Lee. Closing the gap between stability and schedulability: A new task model for cyber-physical systems. In *IEEE Real-Time and Embedded Technology and Applications Symposium, RTAS*, pages 327–337, 2018.

[4] Paolo Pazzaglia, Claudio Mandrioli, Martina Maggio, and Anton Cervin. DMAC: deadline-miss-aware control. In *31st Euromicro Conference on Real-Time Systems, ECRTS 2019*, volume 133 of *LIPIcs*, pages 1:1–1:24. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019.

[5] Amir Aminifar, Soheil Samii, Petru Eles, Zebo Peng, and Anton Cervin. Designing high-quality embedded control systems with guaranteed stability. In *Proceedings of the 33rd IEEE Real-Time Systems Symposium, RTSS 2012*, pages 283–292. IEEE Computer Society, 2012.

[6] Yang Xu, Anton Cervin, and Karl-Erik Årzén. Jitter-robust lqg control and real-time scheduling co-design. In *2018 annual American control conference (ACC)*, pages 3189–3196. IEEE, 2018.

[7] John C Doyle, Bruce A Francis, and Allen R Tannenbaum. *Feedback control theory*. Courier Corporation, 2013.

[8] Marwan Wehaiba el Khazen, Kevin Zagalo, Hadrien Clarke, Mehdi Mezouak, Yasmina Abdeddaïm, Avner Bar-Hen, Slim Ben-Amor, Rihab Bennour, Adriana Gogonel, Kossivi Kougblenou, Yves Sorel, and Liliana Cucu-Grosjean. Work in progress: Kdbench - towards open source benchmarks for measurement-based multicore WCET estimators. In *28th IEEE Real-Time and Embedded Technology and Applications Symposium, RTAS 2022, Milano, Italy, May 4-6, 2022*, pages 309–312. IEEE, 2022.

[9] Cristian Maxim, Adriana Gogonel, Irina Mariuca Asavoae, Mihail Asavoae, and Liliana Cucu-Grosjean. Reproducibility and representativity: mandatory properties for the compositionality of measurement-based WCET estimation approaches. *SIGBED Rev.*, 14(3):24–31, 2017.

[10] Yang Xu, Karl-Erik Årzén, Anton Cervin, Enrico Bini, and Bogdan Tanasa. Exploiting job response-time information in the co-design of real-time control systems. In *2015 IEEE 21st International Conference on Embedded and Real-Time Computing Systems and Applications*, pages 247–256, 2015.

[11] José Carlos Fonseca, Geoffrey Nelissen, Vincent Nélis, and Luís Miguel Pinho. Response time analysis of sporadic DAG tasks under partitioned scheduling. In *11th IEEE Symposium on Industrial Embedded Systems (SIES)*, 2016.

[12] Shuangshuang Chang, Jinghao Sun, Zhixiong Hao, Qingxu Deng, and Nan Guan. Computing exact wcrt for typed DAG tasks on heterogeneous multi-core processors. *Journal of Systems Architecture*, 124(102385), 2022.

[13] Slim Ben-Amor and Liliana Cucu-Grosjean. Graph reductions and partitioning heuristics for multicore DAG scheduling. *J. Syst. Archit.*, 124:102359, 2022.

[14] Shuai Zhao, Xiaotian Dai, and Iain Bate. DAG scheduling and analysis on multi-core systems by modelling parallelism and dependency. *IEEE Transactions on Parallel and Distributed Systems*, 33(12), 2022.