Towards Quality-of-Control-Aware Scheduling of Industrial Applications on Fog Computing Platforms

Mohammadreza Barzegaran
mohba@dtu.dk
DTU Compute
Technical University of Denmark
Kongens Lyngby, Denmark

Anton Cervin
anton@control.lth.se
Department of Automatic Control
Lund University
Lund, Sweden

Paul Pop
paupo@dtu.dk
DTU Compute
Technical University of Denmark
Kongens Lyngby, Denmark

ABSTRACT

In this paper we address Industrial IoT control applications which are safety-critical and real-time, and which have very low latency and jitter requirements. These control applications are virtualized as software tasks running on a Fog Computing Platform that brings computing and deterministic communication closer to the edge of the network, where the industrial "things" are located. Due to the demanding dependability and timing requirements, we consider that the tasks are scheduled with a static-cyclic scheduling policy. We are interested to determine the mapping of the tasks and a schedule table of their activation, such that we maximize the quality-of-control for the control tasks and meet the timing requirements for all tasks, including non-critical real-time tasks. We have proposed a Simulated Annealing-based metaheuristic to solve this problem, and we have evaluated the solution on several test cases.

CCS CONCEPTS

• **Theory of computation** → **Scheduling algorithms**; **Simulated annealing**; • **Computer systems organization** → *Real-time system specification*; • **Hardware** → Safety critical systems.

KEYWORDS

scheduling, simulated annealing, quality-of-control, fog platform

ACM Reference Format:
Mohammadreza Barzegaran, Anton Cervin, and Paul Pop. 2019. Towards Quality-of-Control-Aware Scheduling of Industrial Applications on Fog Computing Platforms. In *Workshop on Fog Computing and the IoT (IoT-Fog '19), April 15–18, 2019, Montreal, QC, Canada.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3313150.3313217

1 INTRODUCTION

We are at the beginning of a new industrial revolution, i.e., Industry 4.0, which is underpinned by a digital transformation that will affect all industries. Industry 4.0 will only become a reality through the convergence of Operational and Information Technologies (OT & IT), which use different computation and communication technologies. OT consists of cyber-physical systems that monitor and control physical processes. These application areas are typically safety-critical and real-time, requiring guaranteed extra-functional properties, such as, real-time behavior, reliability, availability, industry-specific safety standards, and security.

A new paradigm, called *Fog Computing*, is envisioned as an architectural means to realize the IT/OT convergence [27]. According to the OpenFog consortium, Fog Computing is a "system-level architecture that distributes resources and services of computing, storage, control and networking anywhere along the continuum from Cloud to Things". The main component of a Fog Computing platform is the *fog node* (FN). In many applications, including industrial automation and robotics, several layers of fog nodes with differing computation, communication and storage capabilities will evolve, from powerful high-end fog nodes to low-end fog nodes with limited resources. The main benefits of Fog Computing for Industrial IoT is that it virtualizes and integrates several separate equipment, such as gateways, Programmable Logic Controllers (PLCs), Industrial PCs. This will lead to 50% reduction in hardware costs and will bring the same ease of deployment and configuration as the one taken for granted in Cloud Computing [29].

An FN is equipped with computational resources that allows the execution of applications and it is connected to a larger data processing facility like a Cloud environment. Regarding computation, we assume that the control tasks are running in an Real-Time Operating System using real-time scheduling policies (we consider static-cyclic scheduling in this paper), and the control applications are separated in different "partitions" enforced using hardware-supported virtualization. We assume that the FN is connected to the machines through a deterministic communication solution, such as IEEE 802.1 Time-Sensitive Networking (TSN) [14]. Such Fog Computing Platforms (FCPs) are envisioned, for example, in the FORA EU project[1], and in the "Nerve" products from TTTech [29].

The FCP allows to increase the spatial distance between the physical process and the FN that controls it, allowing the control functions to be executed remotely on the FN. However, the way the FCP is configured has an impact on the control performance of the control applications. In this paper, we are interested in the configuration of the FCP, such that the extra-functional properties of the control tasks are guaranteed. In particular, we address the problem of mapping and scheduling of the control tasks on the FNs; we do not address the issue of partition optimization [28] and the scheduling of messages on TSN [9], leaving these aspects for future work.

IoT-Fog '19, April 15–18, 2019, Montreal, QC, Canada

ACM ISBN 978-1-4503-6698-4/19/04…$15.00
https://doi.org/10.1145/3313150.3313217

[1] http://fora-etn.eu

Related work. There is already a lot of work on various topics related to Fog Computing, see surveys such as [18, 21, 31]. However, the work so far addresses quality-of-service applications that are not safety-critical and real-time. None of the approaches for resource management [1] (that configure the FCP resources) are applicable to our context. However, there have been several related works in the area of co-design of control and real-time [3, 4, 19, 23, 26, 30, 33] which have tackled the design of controllers and scheduling of the control tasks with respect to the control performance. The co-design procedure involves designing of control applications such that the controller is robust against degradation due to scheduling of the tasks. Related work has also investigated the impact of the virtualization of control applications (in Cloud Computing) [10, 12, 13]. The virtualization of control applications involves resource management and scheduling of control applications for guaranteed control performance. The performance of a *feedback control system* (FCS) is evaluated by measuring various parameters such as settling time, rise time, overshoot, offset error, etc. Various computational methods have been introduced to evaluate the performance of an FCS [2, 11, 25].

Researchers [7, 8, 24, 32] have addressed the problem of scheduling control tasks for improving the control performance. In the context of static scheduling, researchers have not considered preemption, i.e., allowing tasks to be preempted by other tasks when constructing the schedule tables. The control performance is not only affected by the scheduling of tasks but also affected by the scheduling of messages in networks. Only a few of the works concerning scheduling of control tasks consider Deterministic Ethernet, such as TSN [17].

In this paper, we propose a Simulated Annealing (SA)-based metaheuristic strategy for the mapping and scheduling of mixed-criticality tasks. The tasks are scheduled such that the quality-of-control (QoC) for all critical control applications is maximized and deadlines are met for all tasks, including non-critical real-time tasks. Within the SA, the static-cyclic schedule table is synthesized based on an offline simulation of an *Earliest Deadline First* (EDF) scheduling policy, controlled by SA-generated parameters such as offsets and relative deadlines. Compared to related work, we take into account non-control real-time tasks, and we allow preemption in the static schedules (decided at design time), increasing thus the solution space.

2 CONTROL THEORY

The mathematical relation between the input, output and state variables of a dynamical system can be modelled as a linear differential equation and denoted by a state-space representation captured by (1) [22]:

$$\begin{aligned} \dot{x}(t) &= Ax(t) + Bu(t) \\ y(t) &= Cx(t) + Du(t), \end{aligned} \tag{1}$$

where $x(t)$, $u(t)$ and $y(t)$ denote the state, input and output respectively. The state space representation of a dynamical system can be replaced by the transfer functions [22]. A controller can drive the output of the dynamical system to the desired state by applying an appropriate control signal $u(t)$. Various methods can be utilized to calculate suitable control signals for a dynamical system [22]. An FCS aims to minimize the deviation of the current output from the desired output $r(t)$ by sampling the deviation $e(t) = r(t) - y(t)$. In this paper, we generically assume $r(t) = 0$.

A control application usually has three tasks, (i) sensor, (ii) control and (iii) actuation. Data from sensors is captured and processed by the sensor task. The control signals, based on the implemented control method, are calculated in the control task, which implements the control law. The actuation command is produced based on the control signals in the actuation task. A control loop is widely dependent on time, meaning that the time variations in each task, both in terms of execution and input-output delivery, affect the control performance. For example, *release jitter* is the worst-case delay between the arrival of a task and its release. It is possible to neutralize the effects of fixed communication latency in links between sensors to processing elements and processing elements to actuators by utilizing proper compensators in the control method [15]. Given that most of the sensors and actuators are mechanical parts, the time dependencies and the related variations for the internal dynamics of the actuators and sensors can be either neglected or compensated by implementing proper compensators in the control law. However, the scheduling of tasks has an impact on the control performance, which we address in this paper.

3 SYSTEM ARCHITECTURE AND APPLICATION MODELS

Architecture model. The FCP architecture is modelled as a set of nodes denoted with Υ. Each node, $v_i \in \Upsilon$ can be a processing element es_i, network switch, sensor or actuator. In this architecture, each processing element is an FN. Each FN es_i has a set of cores CS_i. The tasks are scheduled using static cyclic scheduling. The set of all schedule tables in the FCP is denoted with $\mathcal{S}$, and will be determined by our optimization approach in Sect. 4. For each core $cs_j \in CS_i$, its schedule table is denoted with $s_j \in \mathcal{S}$. The static schedule table captures the start time and finishing time of the tasks which are assigned to the respective core. The static schedule table repeats with a hyperperiod, which is the least common multiple of all the assigned task periods, see the application model below. An example architecture is presented in Fig. 1.

Application model. The set of all applications in the system is denoted with Γ. An application is modelled as a directed acyclic graph (DAG), $A_i(H_i, E_i) \in \Gamma$. Each node $h_j(F_j, C_j) \in H_i$ of the graph represents one task. Each edge $\epsilon_{mn} \in E_i$ captures the data flow between tasks, which means that the output of h_m is the input

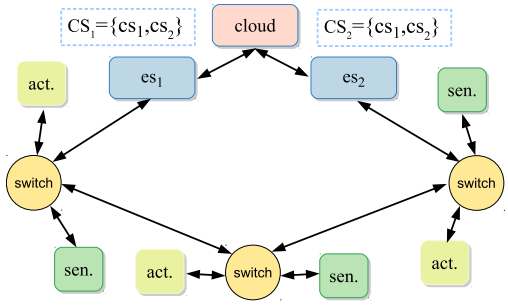


Figure 1: Example architecture model

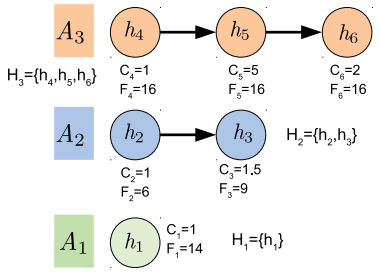


Figure 2: Example application model

to h_n. A task produces its output when it terminates. A task will be ready to execute when all its inputs have arrived. For each task h_j we know its worst-case execution time (WCET) C_j, its period F_j and its deadline D_j. The WCET of a task is specified for each core where the task is considered for mapping. The tasks are mapped to cores by utilizing the function $M : H_i \longrightarrow N$, where N is the set of all cores in the FNs of the FCP. The mapping is decided by our optimization approach.

An example application model composed of three applications is presented in Fig. 2. The WCETs and periods of the tasks (in milliseconds) are given in the figure; in this example, the deadlines are equal to the periods. The applications A_1 and A_2 represent Real-Time Applications (RTA) that do not implement a FCS (they are non-control tasks that have hard deadlines). A_3 represents a Single Input Single Output (SISO) Control Application (CA). In our example, A_3 implements a controller that controls the speed of a DC motor. The task $h_4 \in H_3$ filters the sensor data that is the current speed of the rotor. The task $h_5 \in H_3$ represents a PID controller that gets the speed of rotor from $h_4 \in H_3$. The task $h_6 \in H_3$ represents an actuator that is an electric voltage regulator; it takes the control signal of $h_5 \in H_3$ as input and calculates the voltage command for the actuator. A unit function is used for both sensor and actuator tasks, h_4 and h_6. A discrete PID controller with wide stability margin is designed and used in task h_5. The transfer function of the system plant is shown in (2) and the parameters for the transfer function are shown in Table 1.

$$P(s) = \frac{\dot{\Theta}(s)}{V(s)} = \frac{(K_t + K_e)/2}{(Js + b)(Ls + R) + ((K_t + K_e)/2)^2}. \quad (2)$$

Table 1: Parameters for the function (2)

Symbol	Definition	Unit
$\dot{\Theta}$	angular velocity of the rotor	rad/s
V	electric voltage of the stator	V
J	moment of inertia of the rotor	kg·m^2
b	motor viscous friction	N·m·s
K_e	electromotive force constants	N/rad/s
K_t	motor torque constants	N·m/A
R	electric resistance	Ω
L	electric inductance	H

4 PROBLEM FORMULATION AND SOLUTION

Given (1) a set Γ of applications and (2) the architecture of the system modelled by nodes Υ, we want to determine a mapping M of the tasks to cores and the schedule tables $\mathcal{S}$, such that (1) maximum control performance is achieved for the CAs, (2) the deviation between these performances is minimized for the CAs, and (3) the deadlines for all tasks (in both CAs and RTAs) are met.

Optimization Strategy. Our problem is NP-complete. We propose a *Simulated Annealing* (SA)-based metaheuristic approach extended from [20] in order to solve the problem. SA is an optimization heuristic that tries to find the global optimum by performing design transformations (also called moves) to generate new "neighboring solutions" of the current solution [5]. Each visited solution is evaluated in terms of the Cost of Control function from eq. (3). A solution is invalid if the deadlines are not satisfied. Note that SA, being a metaheuristic, is not guaranteed to find the optimal solution.

To decide the mapping, the SA "moves" change the mapping of tasks to cores (swapping two randomly selected tasks). Given a mapping, the schedule table is synthesized based on "simulating" an *Earliest Deadline First* (EDF) scheduling policy at design time. EDF is a well-known scheduling algorithm [6] that decides at runtime which of the ready tasks gets to run on the processor. EDF uses dynamic task priorities based on their relative deadlines d_i: The task with the earliest deadline, considering the current time moment, is given the highest priority and may interrupt lower priority tasks. The schedule is influenced by the offsets of the tasks (their earliest start times) and the relative deadlines. Considering that each task runs for its WCET, and given a mapping, a set of offsets and deadlines, we simulate how the EDF scheduling policy would perform the scheduling on each core. We consider the outcome of such a simulation to be the schedule tables $\mathcal{S}$. In our approach, SA uses moves to change the task offsets and relative deadlines d_i (smaller or equal to the respective task deadline D_i), before using the EDF simulation. By controlling the offsets and relative deadlines, we generate different schedules, exploring thus the solution space.

Cost of Control. Several functions have been proposed for measuring the quality-of-control of an FCS [25]. In this paper, we use a performance index that evaluates the behavior of the controller over time and call it the *Cost of Control* (CoC) function. The control performance is improved if the CoC decreases. The CoC is a quadratic function of the tracking error and the control signal:

$$\text{CoC} = \lim_{t \to \infty} \int_0^T (e^2(t) + u^2(t))dt. \quad (3)$$

We determine the CoC for a given application A_i, mapping M and schedule S using the Matlab toolbox Jitterbug [16]. The jitter for the application A_i is extracted from the schedule tables S. For the known application A_i, Jitterbug calculates the performance index based on the system model and the respective jitter. Jitterbug assumes that there are stationary stochastic disturbances that drive the plant away from the desired state. It further assumes that the timing variation of the controller tasks in each period can be described using a set of independent random delays. The delay distributions are specified as discrete-time probability density functions with a time-grain δ.

Internally, Jitterbug samples the cost function, the plant dynamics and the stochastic disturbances with the time step δ. Based on the random delay descriptions, it then formulates the closed-loop system as a jump linear system and calculates the covariance of the plant state x in each Markov node. Finally, it evaluates the CoC based on the stationary probability of being in each Markov node and the penalty assigned to different plant states and control signals.

5 RESULTS

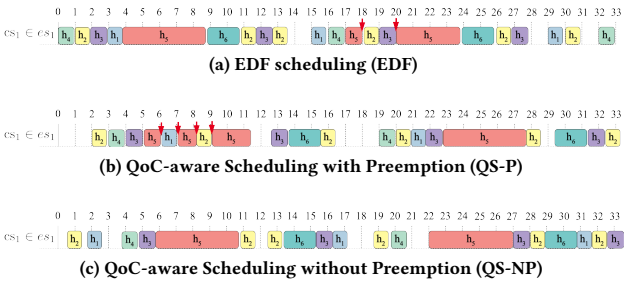
We have used several test cases to evaluate our proposed solution. The solution was implemented in C# and all the experiments were run on a laptop with an i7 CPU at 2.8 GHz and 16 GB of RAM, with a time limit of 10 to 20 minutes, depending on the size of the test case.

Let us first discuss the results obtained on the example applications from Fig. 2, considering an FCP consisting of one single-core FN. We have generated the schedule table using our proposed strategy from Sec. 4. The resulting schedules are depicted in Fig. 3b and 3c. The schedule in Fig. 3b allows preemption (called QS-P), whereas in Fig. 3c we do not allow preemption (QS-NP). For comparison purposes, we have also generated a schedule as it would be produced if we applied the EDF algorithm (EDF) shown in Fig. 3a, but without optimizing the schedule for quality-of-control. The Table 2 compares these alternative solutions in terms of the performance of the control application (CA) and the impact on the non-control applications (RTAs). The values in the Jitter and CoC columns are sums over the respective applications and the values in the "Missed deadlines" column are the total number of deadline misses for RTAs (CAs meet all their deadlines) in a hyperperiod. (Smaller means better for all values in the table.)

Table 2: Three different solutions for the applications in Fig. 2

Solutions	CA Jitter	RTAs Jitter	CoC	Missed deadlines
EDF	2.874 μs	13.006 μs	3.867	3
QS-P	0.918 μs	6.689 μs	2.399	0
QS-NP	1.034 μs	15.693 μs	1.961	2

As we can see, the EDF solution, which is not aware of Quality-of-control (QoC), leads to poor quality solutions: large CoC values, large CA and RTA jitters (large jitters are detrimental also for RTA), and missed deadlines for RTA tasks. The QoC-aware solution found by QS-NP obtains the best control performance, at the expense of missing the deadlines for RTA (hence the solution is not valid) and increased jitters. Only by allowing preemption in the schedules (the solution called QS-P) we are able to meet all the deadlines, and also obtain good control performance.



(a) EDF scheduling (EDF)

(b) QoC-aware Scheduling with Preemption (QS-P)

(c) QoC-aware Scheduling without Preemption (QS-NP)

Figure 3: Three different schedules for the applications in Fig. 2

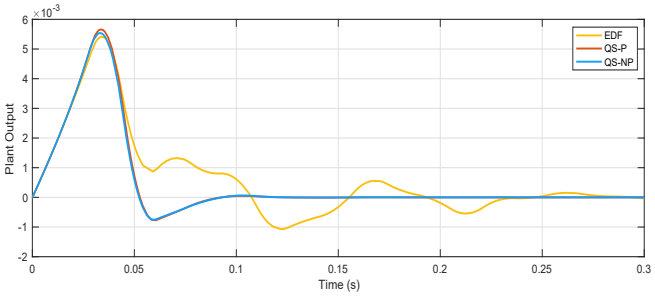


Figure 4: Plant output depicting QoC of the three solutions

The control performance is visually depicted in Fig. 4 for each of the three solutions in terms of FCS plant output over time. The FCS behavior of EDF has a long settling time, with residual errors. Comparing the QS-P and QS-NP solutions, we can see that the behavior is almost the same. The QS-P has a slightly larger overshoot. The conclusion is that by allowing preemption we can still obtain good quality of control, at the same time meeting the deadlines for all applications.

We have also evaluated our proposed solutions on six additional test cases, see Table 3, where column 2 shows the number of cores in the FCP. Several different CAs and RTAs are used, see columns 3 and 4 for their respective number. The values in the CoC columns are sums and the values in the "Misses" columns are the total number of deadline misses for RTAs (CAs meet their deadlines in all cases). The results confirm the conclusion from our first experiment: QS-P obtains the best results (good control performance and no missed deadlines). Although there are cases where QS-NP obtains slightly better control performance, these solutions are not valid because of the missed deadlines for RTA.

6 CONCLUSIONS AND FUTURE WORK

In this paper we have addressed the problem of mapping and scheduling mixed-criticality applications (both CA and non-control RTA) on a FCP. We have proposed an SA-based metaheuristic strategy that uses an EDF simulation to generate the schedules. We have evaluated this strategy on several test cases. As the results show, considering the QoC of the control applications during the optimization (hence, being control-aware) we can obtain solutions that have a good control performance. However, this improved control performance may come at the expense of missing the deadlines for non-control applications. We have shown that allowing preemption in the schedules we can find solutions that can also meet these deadlines.

The successful virtualization of control, achieving the same control performance (and dependability) as the one taken for granted in OT, is a crucial step towards the adoption of Fog Computing

Table 3: Evaluation results for six test cases

#	Cores	No. of CA	No. of RTA	Total No. of Tasks	CoC for QS-P	Misses for QS-P	CoC for QS-NP	Misses for QS-NP
1	4	4	8	20	2.718	0	2.481	1
2	4	4	10	22	3.802	0	3.264	2
3	4	4	16	28	2.586	0	2.719	0
4	5	5	9	24	3.432	0	3.374	1
5	5	6	7	25	4.280	0	4.215	1
6	5	5	15	30	3.594	0	3.640	1

in the industrial area. In our future work, we will consider period selection for control applications; we will take into account the effect of spatial and temporal separation (partitioning) on the control performance and we will also consider the effect of the communication.

ACKNOWLEDGMENTS

The research leading to these results has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No. 764785, FORA—Fog Computing for Robotics and Industrial Automation.

REFERENCES

[1] Swati Agarwal, Shashank Yadav, and Arun Kumar Yadav. 2015. An architecture for elastic resource allocation in Fog Computing. *International Journal of Computer Science & Communication* 6, 2 (2015), 201–207.

[2] Padraig Basquel, Ronan Burke, and Paul Curran. 2017. Optimal closed-loop transfer functions for non-standard performance indices. In *Proceedings of the Signals and Systems Conference*, Vol. 28. IEEE, Killarney, Ireland, 1–6.

[3] Moris Behnam and Damir Isovic. 2007. Real-time control and scheduling co-design for efficient jitter handling. In *Proceedings of the 13th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications*. IEEE, Daegu, South Korea, 516–521.

[4] Michael S Branicky, Stephen M Phillips, and Wei Zhang. 2002. Scheduling and feedback co-design for networked control systems. In *Proceedings of the 41st IEEE Conference on Decision and Control*, Vol. 2. IEEE, Las Vegas, NV, USA, 1211–1217.

[5] Edmund K. Burke and Graham Kendall. 2013. *Search Methodologies: Introductory Tutorials in Optimization and Decision Support Techniques* (2nd ed.). Springer Publishing Company, Inc.

[6] Giorgio C. Buttazzo. 2011. *Hard Real-Time Computing Systems: Predictable Scheduling Algorithms and Applications* (3rd ed.). Springer Publishing Company, Inc.

[7] Anton Cervin. 1999. Improved scheduling of control tasks. In *Proceedings of the 11th Euromicro Conference on Real-Time Systems*. IEEE, York, UK, 4–10.

[8] Anton Cervin. 2003. *Integrated Control and Real-Time Scheduling*. Ph.D. Dissertation. Lund University.

[9] Silviu S. Craciunas, Ramon Serna Oliver, Martin Chmelík, and Wilfried Steiner. 2016. Scheduling Real-Time Communication in IEEE 802.1Qbv Time Sensitive Networks. In *Proceedings of the 24th International Conference on Real-Time Networks and Systems*. ACM, Brest, France, 183–192.

[10] T. Cruz, P. SimÃţes, and E. Monteiro. 2016. Virtualizing Programmable Logic Controllers: Toward a Convergent Approach. *IEEE Embedded Systems Letters* 8, 4 (Dec 2016), 69–72.

[11] FJ Ellert. 1966. Performance indices for linear systems based on standard forms. In *Proceedings of the 5th Symposium on Adaptive Processes*, Vol. 5. IEEE, USA, 643–648.

[12] Omid Givehchi, Jahanzaib Imtiaz, Henning Trsek, and Juergen Jasperneite. 2014. Control-as-a-service from the cloud: A case study for using virtualized PLCs. In *Proceedings of the 10th IEEE Workshop on Factory Communication Systems*. IEEE, Toulouse, France, 1–4.

[13] Thomas Goldschmidt, Mahesh Kumar Murugaiah, Christian Sonntag, Bastian Schlich, Sebastian Biallas, and Peter Weber. 2015. Cloud-based control: A multi-tenant, horizontally scalable soft-PLC. In *Proceedings of the IEEE International Conference on Cloud Computing*. IEEE, New York, NY, USA, 909–916.

[14] IEEE. 2016 (accessed December. 12, 2018). *Official Website of the 802.1 Time-Sensitive Networking Task Group*. http://www.ieee802.org/1/pages/tsn.html

[15] Miroslav Krstic. 2009. *Systems & Control: Foundations and Applications Delay Compensation for Nonlinear, Adaptive, and PDE Systems*. Birkhäuser Boston.

[16] Bo Lincoln and Anton Cervin. 2002. Jitterbug: A tool for analysis of real-time control performance. In *Proceedings of the 41st IEEE Conference on Decision and Control*, Vol. 2. IEEE, Las Vegas, NV, USA, 1319–1324.

[17] Rouhollah Mahfouzi, Amir Aminifar, Soheil Samii, Ahmed Rezine, Petru Eles, and Zebo Peng. 2018. Stability-aware integrated routing and scheduling for control applications in Ethernet networks. In *Proceedings of the Design, Automation & Test in Europe Conference & Exhibition*. IEEE, Dresden, Germany, 682–687.

[18] Redowan Mahmud, Ramamohanarao Kotagiri, and Rajkumar Buyya. 2018. *Fog Computing: A Taxonomy, Survey and Future Directions*. Springer Singapore, Singapore, 103–130.

[19] Pau Martí, José Yépez, Manel Velasco, Ricard Villà, and Josep M Fuertes. 2004. Managing quality-of-control in network-based control systems by controller and message scheduling co-design. *IEEE transactions on Industrial Electronics* 51, 6 (2004), 1159–1167.

[20] Shane D. McLean, Paul Pop, and Silviu S. Craciunas. 2019. *Mapping and scheduling of real-time tasks on multi-core autonomous driving platforms*. Technical Report. Technical University of Denmark.

[21] Carla Mouradian, Diala Naboulsi, Sami Yangui, Roch H. Glitho, Monique J. Morrow, and Paul A. Polakos. 2018. A Comprehensive Survey on Fog Computing: State-of-the-Art and Research Challenges. *IEEE Communications Surveys and Tutorials* 20, 1 (2018), 416–464.

[22] Katsuhiko Ogata and Yanjuan Yang. 2002. *Modern control engineering*. Vol. 4. Prentice Hall India.

[23] Kyung-Joon Park, Man-Ki Yoon, Kyungtae Kang, and Chang-Gun Lee. 2011. Scheduling and control co-design under end-to-end response time constraints in cyber-physical systems. In *Proceedings of the IEEE Conference on Computer Communications Workshops*. IEEE, Shanghai, China, 762–767.

[24] Reinhard Schneider, Dip Goswami, Alejandro Masrur, and Samarjit Chakraborty. 2012. QoC-oriented efficient schedule synthesis for mixed-criticality cyber-physical systems. In *Proceedings of the Forum on Specification and Design Languages*. IEEE, Vienna, Austria, 60–67.

[25] W.C Schultz and V.C Rideout. 1943. Control System Performance Measures: Past, Present and Future. *IRE transactions on automatic control* 10, 2 (1943), 22–35.

[26] Daniel Simon, Alexandre Seuret, and Olivier Sename. 2017. Real-time control systems: feedback, scheduling and robustness. *International Journal of Systems Science* 48, 11 (2017), 2368–2378.

[27] Wilfried Steiner and Stefan Poledna. 2016. Fog computing as enabler for the Industrial Internet of Things. *Elektrotechnik Und Informationstechnik* 133, 7 (2016), 1–5.

[28] Domitian Tamas-Selicean and Paul Pop. 2015. Design Optimization of Mixed-Criticality Real-Time Systems. *ACM Transaction on Embedded Computing* 14, 3 (May 2015), 50–78.

[29] TTTech-Computertechnik-AG. 2018 (accessed December 20, 2018). *Fog Computing White Paper*. https://www.tttech.com/wp-content/uploads/TTTech_Fog-Computing_White-paper.pdf

[30] Yang Xu, Karl-Erik Årzén, Enrico Bini, and Anton Cervin. 2017. LQG-based control and scheduling co-design. *IFAC-PapersOnLine* 50, 1 (2017), 5895–5900.

[31] Wei Yu, Fan Liang, Xiaofei He, William Grant Hatcher, Chao Lu, Jie Lin, and Xinyu Yang. 2018. A survey on the edge computing for the Internet of Things. *IEEE Access* 6 (2018), 6900–6919.

[32] Fumin Zhang, Klementyna Szwaykowska, Wayne Wolf, and Vincent Mooney. 2008. Task scheduling for control oriented requirements for cyber-physical systems. In *Proceedings of the Real-Time Systems Symposium*. IEEE, Barcelona, Spain, 47–56.

[33] Yun-Bo Zhao, Hui Dong, and Hongjie Ni. 2017. Scheduling and Control Co-Design for Control Systems under Computational Constraints. *IFAC-PapersOnLine* 50, 1 (2017), 5881–5886.