# CoCPN – Towards Flexible and Adaptive Cyber-Physical Systems Through Cooperation

Florian Rosenthal*, Markus Jung†, Martina Zitterbart†, and Uwe D. Hanebeck*

***Abstract*—This work is concerned with our ongoing research project CoCPN: Cooperative Cyber Physical Networking that has the goal to allow cooperation between control applications and the communication system that constitute cyber-physical systems. We describe the envisioned architecture of CoCPN and outline how it improves the flexibility of cyber-physical systems by cooperatively sharing a common network infrastructure. We also present our simulation tool CoCPN-Sim that we developed as a method to thoroughly investigate the interaction between control applications and the communication system. By providing the results of selected simulations using the well-known inverted pendulum, we identify potential aspects that can be exploited for cooperation and hence serve as starting points towards more flexible and adaptive cyber-physical systems.**

## I. Introduction

Cyber-physical systems (CPS), such as smart grid, smart factory and smart city, are becoming key components of our future, largely digitalized world [1], [2]. From an abstract technical point of view, they consist of control loops for applications in, e.g., industrial process control, robotics or surveillance [3], that sit on top of a communication system that comes with its own internal control loops (cf. Fig. 1). Typically, both types of control loops are designed independently from each other. Very commonly, the control application has strict requirements, for instance with respect to timely and failure-free delivery of control information, and demands very specific networking support with pre-reserved resources.

The vision of our research project *CoCPN: Cooperative Cyber Physical Networking* is to enable cooperation between the control application and the network-internal control loops and, thus, to i) pave the way for the usage of standard networking equipment being able work with less strict network requirements and ii) to harness potential flexibility in the control application, e.g., through adaptable event-based systems. Consequently, this should make a more dynamic and autonomic operation of future cyber-physical systems possible.

The contribution of this paper is threefold. First and foremost, we outline the goals and concepts of CoCPN. Second, we introduce *CoCPN-Sim* [4], the simulation tool we developed in the course of this project, which combines MATLAB with the event-driven network simulator OMNeT++ and allows us to investigate the cooperation between control applications and communication system in detail. Finally, we present and discuss results that show the interplay between control application and communication system.

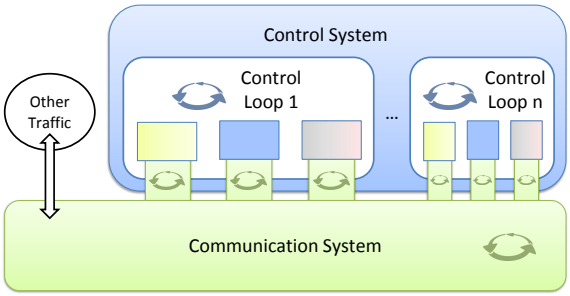


Fig. 1: Abstract view of a cyber-physical system.

*Outline:* The remainder of this paper is organized as follows. First, we review some recent research concerned with simulating and benchmarking cyber-physical systems in Section II. Then, in Section III, we outline the basic ideas and goals of CoCPN and describe its envisioned architecture. Subsequently, we introduce the associated simulation framework CoCPN-Sim in Section IV. In Section V, we provide experimental results and a discussion of these before we conclude this work in Section VI.

## II. Related Work

Since simulation-based benchmarking is an easy and repeatable method to investigate the behavior of different influencing factors in CPS, plenty of work has been carried out recently with regards to simulation and benchmarking. Simulation-based benchmarking is particularly suited when the focus is on higher-level protocols and quantities, such as end-to-end delays or jitter, are of interest [5]. Consequently, the majority of the existing approaches focuses on certain aspects of a CPS only, which renders them impractical for our holistic view of such systems.

For instance, in [6] a framework is presented that captures both the processing and network nodes of a CPS, while [7], [8] are concerned with designing interfaces for co-simulation of the physical and the computational components. However, [7] focuses on the interaction between the control software and the physical plant to be controlled itself and does not explicitly take the underlying network into account. On the other hand, the work [8] employs the IEEE High Level Architecture standard and hence utilizes a relatively high degree of abstraction. In [9], the authors propose a set of performance indicators from both network and control domain that should be used

*Florian Rosenthal and Uwe D. Hanebeck are with the Institute for Anthropomatics and Robotics, Karlsruhe Institute of Technology (KIT), Germany. Email: florian.rosenthal@kit.edu, uwe.hanebeck@ieee.org

†Markus Jung and Martina Zitterbart are with the Institute of Telematics, Karlsruhe Institute of Technology (KIT), Germany. Email: markus.jung@kit.edu, zitterbart@kit.edu

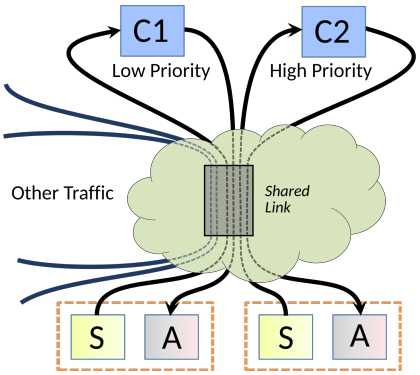


Fig. 2: Implicit coupling of two control applications, each consisting of a sensor (S), a controller (C) and an actuator (A), due to a shared link.

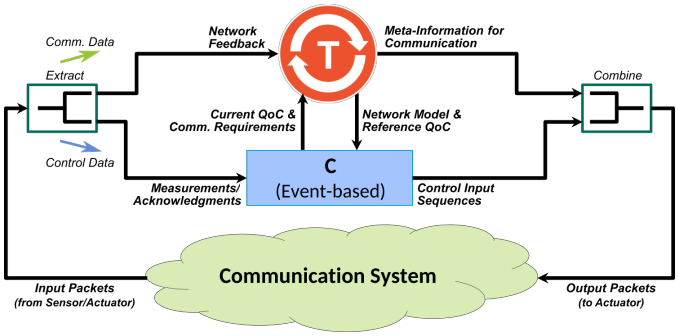


Fig. 3: Data flow of a CoCPN control application. The translator (T) as key component of CoCPN enables the cooperation between communication system and control application (C).

for benchmarking, yet without introducing a corresponding evaluation platform. For wireless CPS, such a platform that integrates simulated systems into a real-world testbed has been presented recently in [10]. A summary of other co-simulation and modeling approaches can be found in [3], [11], [12].

## III. CoCPN Architecture

The goal of our research project CoCPN is to improve the flexibility of control applications and communication systems in CPS. Thus, instead of specialized networks, we presume a communication system based on standard networking equipment as an adaptable foundation for networked applications. This approach however induces three main challenges that have to be addressed. First, applications may have very different requirements regarding latencies, data rate, reliability and priorities. Second, access to limited resources must be balanced between applications such that high performance of the overall CPS is ensured. Finally, sharing a common network infrastructure can cause unwanted side effects since applications may interfere with each other.

Consider the situation in Fig. 2, where two control applications C1 and C2 share a common link within the network. Sharing this link establishes an *implicit coupling* between the two control loops. We call control applications that are subject to the coupling effect at a shared link *adjacent applications*. Any over-utilization of the shared link, which, e.g., may occur when C1 increases its data rate, leads to larger delays or even packet losses. As this affects all adjacent applications, the *quality of control* (QoC) that can be achieved by C2 is thus degraded, despite the fact that C2 has a much higher priority within the CPS.

CoCPN seeks to address the challenges mentioned beforehand by a cooperative use of communication resources. Our approach relies on cooperation i) between control applications and communication system and ii) between applications themselves. Both domains, control and communication, strive for a common goal: a high overall QoC across the whole CPS without imbalances or displacements between adjacent applications.

Cooperation between control applications and communication system is enabled by exchanging information about their current state, individual requirements and additional meta-data. In the CoCPN architecture, the so called *translator* facilitates this exchange. It acts as a mediator between applications and the underlying communication system. Fig. 3 illustrates the data flow of a CoCPN control application (C) and its interaction with the translator (T). Control data is is processed by the control application. Accompanying monitoring feedback is collected within the communication system. The translator processes this feedback to provide network models, which describe the actual state of the communication system, to the control application and translates communication requirements to matching networking primitives.

For control applications, we leverage paradigms such as model predictive and sequence-based control [13], [14], and employ an event-driven information exchange. This furnishes us both the robustness to operate without hard real-time guarantees and the ability to adapt to varying communication resources and control demands. The communication system in turn adapts to application specific communication requirements. Supported by the CoCPN translator, it chooses suitable mechanisms and strategies to, for instance, maintain sufficiently low latencies.

The previously discussed mechanisms pave the way for cooperation between control applications and communication system. This improves their adaptivity and robustness, but is not yet sufficient to prevent negative side effects due to implicit coupling between control applications. Hence, CoCPN also introduces a cooperation mechanism among applications to support a cooperative use of shared communication resources. Key idea behind this mechanism is to cope with the inevitable coupling between applications by making it *explicitly* visible to applications and communication system. Supported by both communication system and translator, each control application locally adjusts its communication behavior by incorporating information about the actual state of the communication system. Cooperatively, they balance the available resources within the communication system *fairly* between adjacent control applications to arrive at an evenly distributed QoC.

Roughly speaking, the CoCPN mechanism for cooperative resource sharing works by weighting the urgency to communicate of adjacent control applications. Here, urgency

is reciprocal to the QoC of a control application. Based on monitoring feedback, an estimation of the actual state of the communication system and of the QoC of adjacent control applications is deduced. The local control application provides data about the actual QoC, which is also sent alongside outgoing control data. Thus, it is incorporated into the monitoring feedback for all adjacent control applications. Considering this information, CoCPN aims to balance the available communication resources between adjacent control applications by locally computing a suitable *reference QoC*. Provided that it is possible to vary the communication behavior of control applications by adjusting their QoC, this mechanism permits a dynamical sharing of the communication resources without over-utilization. It is the task of the translator to provide a mapping between the two domains, between QoC and communication behavior, and to actually perform suitable QoC adjustments.

Our approach of explicit coupling enables a CPS to dynamically adapt to varying configurations, requirements and available resources. The translator enables control applications to share communication resources in fair and cooperative manner by adjusting their local QoC. However, this requires the translator to have knowledge about the control application and its interplay with the communication system. As a means to investigate this interplay and, subsequently, to identify opportunities for collaborative approaches, we developed *CoCPN-Sim*, a simulation and evaluation framework that is briefly introduced in the next section.[1]

## IV. CoCPN-Sim at a Glance

CoCPN-Sim integrates the event-driven simulation framework OMNeT++ and the numerical computing platform MATLAB. While MATLAB is widely used in the control community, for instance for design and analysis of control algorithms, OMNeT++ and its accompanying model suite INET are often used for network analyses.

In CoCPN-Sim, MATLAB thus provides the mathematical toolbox for control-related tasks within the CPS and OMNeT++ and INET are used to model the network. Consequently, with CoCPN-Sim more sophisticated and realistic network scenarios can be provided so that cyber-physical systems can be analyzed on a more fine-grained level. To realize the communication between the components of control applications – for instance, sensor data must be transmitted to the controller and control inputs are sent to the plant – in the simulation, data must be exchanged between MATLAB and OMNeT++ and then translated into an equivalent representation in OMNeT++. Additionally, the communication must be integrated into the event-driven workflow of OMNeT++ because the individual components communicate in a clock-controlled manner.

Both MATLAB and OMNeT++ are interfaced by components that are located at either side of CoCPN-Sim, as illustrated in Fig. 4, where the key components are sketched out.

[1]CoCPN-Sim is open source software and available on github [15].

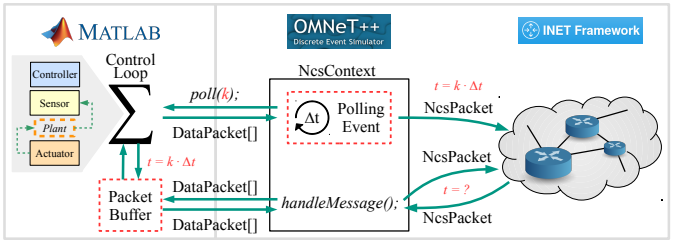


Fig. 4: Overview of the data flow between MATLAB and OMNeT++/INET within CoCPN-Sim.

Major component is the `NcsContext` that resides inside the OMNeT++ part of the simulation and represents an application control loop within OMNeT++. It offers a periodic polling event that is used by the simulation kernel of OMNeT++ to drive the clock-based models within MATLAB and hence displaces the main loop of a pure MATLAB simulation.

As shown in the figure, the `NcsContext` polls a hook function at every time step that in turn prompts MATLAB to perform all necessary computations within the control loop. As a result of this call, a number of `DataPackets`, that is, unified and serializable representations of messages to be exchanged between the components of the control loop, are handed back to OMNeT++. In a typical control application, these `DataPackets` would contain the sensor data to be transmitted to the controller and the control inputs computed by the controller that shall be sent to the plant.

After being transformed into an INET-compatible representation, the data is passed to the network model. From this point on, all further processing steps are carried out asynchronously with respect to the time domain of the control loop. Once a message arrives at a network node corresponding to a component of the application control loop, the `NcsContext` immediately forwards it to the MATLAB side. There, it is either buffered for clock-synchronous processing or directly processed. In any case, responses, such as application layer acknowledgments, can be issued and handed back to OMNeT++. When the next polling event is triggered, the next control cycle is started at the MATLAB side of the simulation and the buffered messages are processed as described above.

After having described the internal data flow of CoCPN-Sim, we outline the simulation workflow, which essentially comprises three steps, in the remainder of this section. First, MATLAB models of the components of the application control loop are implemented. A shared library containing all the required MATLAB functionality is subsequently created using the the MATLAB Compiler SDK [16]. Second, the desired network setup is then specified in OMNeT++. This can either be done by implementing custom models for, e.g., applications, hosts, or routers, or by using any of the models that are already provided by INET. Finally, the simulation can be run after it has been configured by parameterizing all components within OMNeT++ according to the desired scenario.

## V. Selected Experiments

In this section, we present and discuss some experimental results that we have gathered to date with the help of CoCPN-Sim. More precisely, we utilize the well-known task of stabilizing an inverted pendulum on a moving cart to assess the performance characteristics of an event-based control strategy w.r.t. different event trigger configurations. Knowledge of these characteristics will be necessary to enable the translator to conduct purposeful adjustments of control parameters. We use the norm of the control error to evaluate the control performance. Additionally, we evaluate the data rate required by the control strategy, or more precisely, the number of transmitted data packets per second.

In the following, we will first give a detailed description of the control task and the implemented controller in Section V-A. Then, in Section V-B and Section V-C, we describe the topology and characteristics of the underlying network, and the overall setup of the simulation, respectively. Finally, results will be presented and discussed in Section V-D.

### A. Description of the Control Task

As mentioned in the introduction of this section, the control task we are concerned with is to stabilize an inverted pendulum on a cart operating in a transient state, that is, frequent setpoint changes of the cart's position occur. To that end, consider the state of the pendulum at time step $k$ given by

$$\underline{x}_k = \left[ s_k \;\; \dot{s}_k \;\; \phi_k \;\; \dot{\phi}_k \right] ,$$

with $s_k$ denoting the position of the cart (in $\mathrm{m}$) and $\phi_k$ the angle of the pendulum (in $\mathrm{rad}$), chosen such that $\phi_k = \pi$ corresponds to the unstable upward equilibrium. The discrete-time, nonlinear dynamics of the pendulum $\underline{x}_{k+1} = \underline{f}(\underline{x}_k, u_k)$, where $u_k$ is the input force applied to the cart (in $\mathrm{N}$), is obtained by discretizing the continuous-time dynamics with a sampling rate of $t_\mathrm{a} = 0.01\,\mathrm{s}$.

To control the pendulum, we implement a sequence-based controller as in [17] based on a usual infinite-horizon, linear-quadratic regulator. For the computation of the regulator gain and the feedforward, the continuous-time pendulum dynamics is first linearized around the upward equilibrium and then discretized using $t_\mathrm{a}$. The state $\underline{x}_k$ is not measured directly, but only observations of the position and the deviation of the pendulum from the upward equilibrium are available to the controller, both of which corrupted by zero mean Gaussian noise. Hence, a state estimate is required for the computation of the control inputs. For this purpose, we utilize the estimator proposed in [18].

Additionally, the controller does not transmit control inputs every time step, but only when a certain event is triggered. This is achieved by implementing a strategy similar to what has been proposed in [19]. More precisely, new control inputs are only transmitted if the controller detects a significant increase of its underlying cost function, computed based on its current state estimate. Here, the maximum allowed increase is a tuning parameter to be provided by the designer in order to trade off the number of data transmissions against the achievable control performance. Inspired by the notion in [19], we refer to this parameter as *deadband*, denoted by $\delta$, in the following.

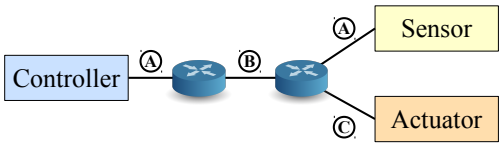


Fig. 5: Network topology used for the experiments.

TABLE I: Link parameters used for the network topology in Fig. 5.

| Link type | Bitrate | Delay |
|---|---|---|
| **A** | $100\,\mathrm{Mbit/s}$ | $50\,\mathrm{ns}$ |
| **B** | $10\,\mathrm{Mbit/s}$ | $50\,\mathrm{ns}$ |
| **C** | $100\,\mathrm{Mbit/s}$ | $50\,\mathrm{ns}\ldots 50\,\mathrm{ms}$ |

### B. Description of the Network Parameters

To investigate the interplay between communication system and control system, we model a simplified network as illustrated in Fig. 5. Sensor and actuator are physically co-located and thus connected to the same router. The controller is connected to this network by a second router. Three different link types **A**, **B**, and **C** are configured using the parameters listed in Table I. Each link is assumed to be ideal, that is, without bit errors. Note that the delay of link type **C** can be varied in order to assess the impact of latencies within the network on the control task. Henceforth, this parameter will be denoted as *controller-actuator delay* and abbreviated by $t_\mathrm{CA}$.

All components of the control loop utilize an UDP/IP network stack so that the reception of data packets is generally not acknowledged by the receiver. However, application-level acknowledgments are sent back from the actuator to the controller upon reception of applicable control inputs. As detailed in [18], the reception of these acknowledgments enables the controller to infer control inputs that where applied in the past and to thus improve the quality of the state estimate.

### C. Simulation Setup

We configured the previously described scenario within CoCPN-Sim to conduct two experiments. The first experiment focuses on the effect of the deadband parameter on control performance and communication behavior. The second experiment briefly studies the interplay between both the deadband and controller-actuator delay. The parameters are chosen to cover a reasonable range w.r.t. control performance and realistic network states. More precisely, in the first experiment the deadband increases exponentially from $\delta = 0.1$ to $\delta = 204.8$ and the controller-actuator delay is set to $t_\mathrm{CA} = 50\,\mathrm{ns}$. In the second experiment, we use the same deadband values but also vary $t_\mathrm{CA}$ within the range given in Table I. For comparison, we also include the case $\delta = 0.0$ in both experiments, which corresponds to a controller that is not event-based but always transmits control inputs.

To alleviate the influence of the random number generators in CoCPN-Sim, we carried out several simulation runs per parameter configuration in each experiment, namely 25 in the

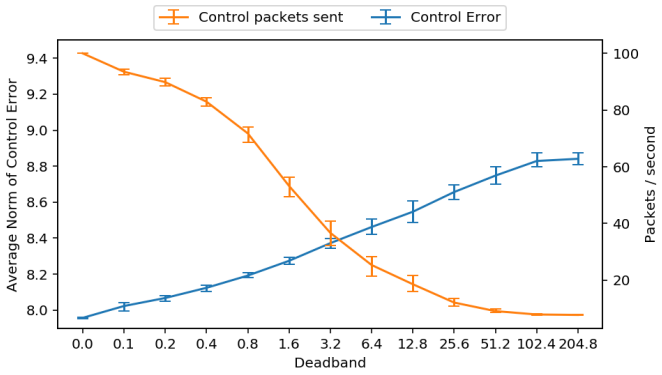


Fig. 6: Average norm of control error and number of packets sent by the controller for each employed deadband $\delta$ and $t_{\mathrm{CA}} = 50\,\mathrm{ns}$.

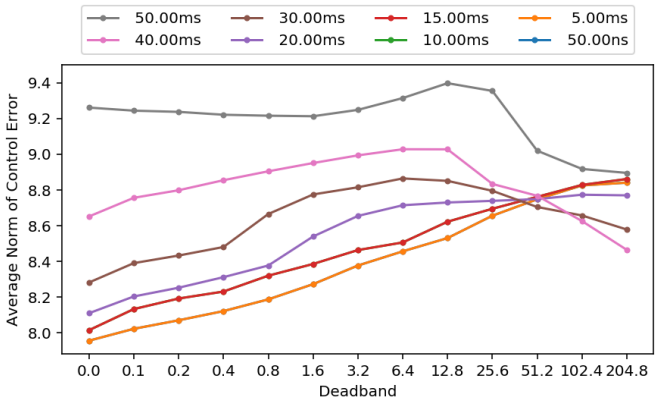


Fig. 7: Average norm of control error for each employed deadband $\delta$ and different values of the controller-actuator delay $t_{\mathrm{CA}}$.

first one and 5 in the second one. In each run, the simulation time was set to $t_{\max} = 30\,\mathrm{s}$. To avoid disruptions due to the initialization of the network, the control task, and in particular the movement of the cart, does not start until $t_1 = 3\,\mathrm{s}$. The true benchmark phase then commences at $t_2 = 12\,\mathrm{s}$ and ends at $t_{\max}$.

Among other metrics, CoCPN-Sim records the actual control error at each sampling instant. Likewise, the number of data packets sent out by the components of the control loop is recorded. To assess the control performance in each experiment, we compute the average norm of the control error for every parameter configuration, where we average over the complete benchmark phase ($t_2 \leq t \leq t_{\max}$) and all simulation runs. Similarly, we compute the average data rate employed by the controller, expressed in terms of the number of transmitted packets per second, for both experiments to evaluate the communication behavior.

### D. Results

The results of the first experiment are visualized in Fig. 6, where the average norm of the control error (orange curve, primary y-axis on the left) and the controller's packet rate (blue curve, secondary y-axis on the right) are plotted for each value of $\delta$. Both metrics were computed as described above and we also plotted error bars to indicate minimum and maximum values.

We get from the curves that the control error grows with increasing values of $\delta$, or, to put it another way, the achieved QoC decreases the more $\delta$ increases. This is an expected behavior because large deadband values at the same time reduce the rate at which data packets are sent to the actuator. Moreover, the results exhibit that the control error also varies to a greater extent for larger values of $\delta$. For the packet rate, the biggest variations are observed for mid-range deadband values.

Two conclusions can be drawn from this experiment. First, the event-based control approach provides a manageable way to influence the control performance. The deadband values we used in the simulations correspond to angle deviations of the pendulum growing from approx. $\pm 0.2°$ to $\pm 3°$. A higher variance of the control performance at high deadband values is most likely the result of a delayed control response, which makes the control loop more susceptible to noise and other disturbances. This motivates further research towards more sophisticated predictive event triggers. Second, there appears to be a connection between $\delta$, the achievable QoC and the packet rate actually employed by the controller. Hence, it should be possible to derive a model of the control task that is suitable to support the computations of the CoCPN translator component.

The results of the second experiment, depicted in Fig. 7, show that delays induced by the network also affect the achievable control performance. In essence, they reveal that increasing delays also lead to larger control errors, which correspond to angle deviations of up to $\pm 15°$. It is worth to mention that the curves for $t_{\mathrm{CA}} = 50\,\mathrm{ns}$ and $t_{\mathrm{CA}} = 5\,\mathrm{ms}$ are equal. This is reasonable since the sampling rate of the control system is $t_{\mathrm{a}} = 10\,\mathrm{ms}$ so that these small delays cannot be perceived by the application. Similarly, the curves for $t_{\mathrm{CA}} = 10\,\mathrm{ms}$ and $t_{\mathrm{CA}} = 15\,\mathrm{ms}$ are equal because in both cases the data packets are processed by the application at the same sampling instant.

For $0.0 \leq \delta \leq 12.8$, all runs except $t_{\mathrm{CA}} = 50\,\mathrm{ms}$ exhibit a very similar behavior with a steadily increasing control error. For lager deadband values, i.e., $\delta \geq 25.6$, the graphs can be differentiated into two groups, depending on the values of $t_{\mathrm{CA}}$. While the control error further increases for $t_{\mathrm{CA}} \leq 15\,\mathrm{ms}$, it either remains constant or even decreases for $t_{\mathrm{CA}} \geq 20\,\mathrm{ms}$. Interestingly, the curves also exhibit that in case of large delays, that is, $t_{\mathrm{CA}} \geq 40\,\mathrm{ms}$, the control performance achieved with large deadband values was superior to the control performance with $\delta = 0.0$.

Our findings from this experiment are threefold. First, the results suggest that presence of relatively large delays, large deadband values and, consequently low data rates, do not necessarily lead to reduced control performance. Second, our simulations also indicate that delays within the network matter. However, it should be possible incorporate the influence of

them into translator models by parameterizing them appropriately. The same applies to the outgoing packet rate (not shown here), which did not differ much across all parameter configurations. Third, the experiment allows to draw conclusions on possible trade-offs to be leveraged for cooperation in CoCPN. Consider the results for $t_{\mathrm{CA}} = 50\,\mathrm{ns}$ and $t_{\mathrm{CA}} = 40\,\mathrm{ms}$. They indicate that a similar control error is obtained for $\delta = 25.6$ and the base case $\delta = 0.0$. However, for the former the required packet rate is about $90\,\%$ lower.

Hence, we can conclude that, in particular in the presence of high delays due to, e.g., an over-utilized link, adapting the desired QoC of the application by the CoCPN translator component is a viable method to drastically reduce the communication rate. This allows for lower delays within the communication system, which in turn enables the control application to reach a higher QoC than before.

## VI. Summary and Outlook

In this work, we introduced our ongoing research project CoCPN: Cooperative Cyber Physical Networking that aims at enabling cooperation between applications and the underlying network in cyber-physical systems. Likewise, we presented the simulation tool CoCPN-Sim, which combines MATLAB and OMNeT++, that we developed to investigate the interaction between the control loops in the applications and the network-internal control loops in detail. Moreover, we provided the results of an experiment that we conducted to study the effect of different transmission rates employed by the event-based controller on both control performance and communication behavior. In a second experiment, we also investigated the interplay between the employed transmission rates and different delays in the network.

The results demonstrated a significant interdependence between the achievable quality of control and network-induced delays. They also highlighted possible trade-offs that can be leveraged in future research and consequently be used as starting points towards more flexible and adaptive cyber-physical systems. In particular they point out that aiming at a lower than optimal QoC can lead to a higher actual control performance as this reduces the utilization of the communication system. Thus, they strongly indicate the feasibility of our cooperative approach.

Additional prospective research will aim at deriving models for the communication behavior of control applications that can be utilized by the cooperation mechanisms of CoCPN. To that end, the impact of further parameters, such as varying delays in the communication of sensor data to the controller, latencies, jitter, or packet losses due to cross-traffic, has to be investigated.

## Acknowledgment

This work is supported by the German Science Foundation (DFG) within the Priority Programme 1914 “Cyber-Physical Networking”.

## References

[1] S. Karnouskos, “Cyber-Physical Systems in the SmartGrid,” in *2011 9th IEEE International Conference on Industrial Informatics*, July 2011, pp. 20–23.

[2] L. Ribeiro, “Cyber-physical Production Systems’ Design Challenges,” in *2017 IEEE 26th International Symposium on Industrial Electronics (ISIE)*, June 2017, pp. 1189–1194.

[3] S. K. Khaitan and J. D. McCalley, “Design Techniques and Applications of Cyberphysical Systems: A Survey,” *IEEE Systems Journal*, vol. 9, no. 2, pp. 350–365, June 2015.

[4] M. Jung, F. Rosenthal, and M. Zitterbart, “Poster Abstract: CoCPN-Sim: An Integrated Simulation Environment for Cyber-Physical Systems,” in *2018 IEEE/ACM Third International Conference on Internet-of-Things Design and Implementation (IoTDI)*, April 2018, pp. 281–282.

[5] C. A. Boano, S. Duquennoy, A. Förster, O. Gnawali, R. Jacob, H. Kim, O. Landsiedel, R. Marfievici, L. Mottola, G. P. Picco, X. Vilajosana, T. Watteyne, and M. Zimmerling, “IoTBench: Towards a Benchmark for Low-power Wireless Networking,” in *2018 IEEE Workshop on Benchmarking Cyber-Physical Networks and Systems (CPSBench)*, April 2018, pp. 36–41.

[6] A. Brokalakis, N. Tampouratzis, A. Nikitakis, S. Andrianakis, I. Papaefstathiou, and A. Dollas, “An Open-Source Extendable, Highly-Accurate and Security Aware CPS Simulator,” in *2017 13th International Conference on Distributed Computing in Sensor Systems (DCOSS)*, June 2017, pp. 81–88.

[7] Y. Zhang, Y. Dong, W. Feng, and M. Huang, “A Co-Simulation Interface for Cyber-Physical Systems,” in *2016 13th International Conference on Embedded Software and Systems (ICESS)*, Aug 2016, pp. 176–181.

[8] T. Roth and M. Burns, “A Gateway to Easily Integrate Simulation Platforms for Co-Simulation of Cyber-Physical Systems,” in *2018 Workshop on Modeling and Simulation of Cyber-Physical Energy Systems (MSCPES)*, April 2018, pp. 1–6.

[9] S. Gallenmüller, S. Günther, M. Leclaire, S. Zoppi, F. Molinari, R. Schöffauer, W. Kellerer, and G. Carle, “Benchmarking Networked Control Systems,” in *2018 IEEE Workshop on Benchmarking Cyber-Physical Networks and Systems (CPSBench)*, April 2018, pp. 7–12.

[10] D. Baumann, F. Mager, H. Singh, M. Zimmerling, and S. Trimpe, “Evaluating Low-Power Wireless Cyber-Physical Systems,” in *2018 IEEE Workshop on Benchmarking Cyber-Physical Networks and Systems (CPSBench)*, April 2018, pp. 13–18.

[11] A. T. Al-Hammouri, M. S. Branicky, and V. Liberatore, “Co-simulation Tools for Networked Control Systems,” in *Hybrid Systems: Computation and Control*, M. Egerstedt and B. Mishra, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 16–29.

[12] W. Li, X. Zhang, and H. Li, “Co-simulation platforms for co-design of networked control systems: An overview,” *Control Engineering Practice*, vol. 23, pp. 44 – 56, 2014.

[13] A. Bemporad, “Predictive Control of Teleoperated Constrained Systems with Unbounded Communication Delays,” in *Proceedings of the 37th IEEE Conference on Decision and Control*, vol. 2, Dec. 1998, pp. 2133–2138.

[14] J. Fischer, A. Hekler, M. Dolgov, and U. D. Hanebeck, “Optimal Sequence-Based LQG Control over TCP-like Networks Subject to Random Transmission Delays and Packet Losses,” in *Proceedings of the 2013 American Control Conference (ACC 2013)*, Washington D.C., USA, Jun. 2013.

[15] M. Jung and F. Rosenthal, “CoCPN-Sim,” 2018. [Online]. Available: https://github.com/spp1914-cocpn/cocpn-sim

[16] The MathWorks, Inc, “MATLAB Compiler SDK.” [Online]. Available: https://www.mathworks.com/products/matlab-compiler-sdk.html

[17] G. Liu, “Predictive Controller Design of Networked Systems With Communication Delays and Data Loss,” *IEEE Transactions on Circuits and Systems—Part II: Express Briefs*, vol. 57, no. 6, pp. 481–485, 2010.

[18] F. Rosenthal, B. Noack, and U. D. Hanebeck, “State Estimation in Networked Control Systems with Delayed and Lossy Acknowledgments,” in *Multisensor Fusion and Integration in the Wake of Big Data, Deep Learning and Cyber Physical System*, S. Lee, H. Ko, and S. Oh, Eds. Cham: Springer International Publishing, 2018, pp. 22–38.

[19] Y. Zhao, G. Liu, and D. Rees, “Packet-Based Deadband Control for Internet-Based Networked Control Systems,” *IEEE Transactions on Control Systems Technology*, vol. 18, no. 5, pp. 1057–1067, Sept 2010.