Benchmarking Networked Control Systems

Sebastian Gallenmüller*, Stephan Günther*, Maurice Leclaire*, Samuele Zoppi†,
Fabio Molinari‡, Richard Schöffauer§, Wolfgang Kellerer†, Georg Carle*
*Chair of Network Architectures and Services, Department of Informatics, TUM, Germany
†Chair of Communication Networks, Department of Electrical and Computer Engineering, TUM, Germany
‡Control Systems Group, Technische Universität Berlin, Germany
§Heisenberg Communications and Information Theory Group, Freie Universität Berlin, Germany
Email: {sebastian.gallenmueller, guenther, maurice.leclaire, samuele.zoppi, wolfgang.kellerer, carle}@tum.de,
molinari@tu-berlin.de, schoeffauer@zedat.fu-berlin.de

***Abstract*—Smartification of devices and ever-increasing connectivity are major factors paving the way towards the creation of networked control systems (NCS). Traditionally, both domains involved in an NCS, network and control, are considered separately. Towards a novel methodology for benchmarking such systems, we propose a series of key performance indicators (KPIs) that are based on well-established foundations of both worlds. These KPIs form our framework for the development of an NCS benchmark. The majority of KPIs, particularly those from the network domain, are applicable to any type of NCS. We are currently implementing an evaluation platform based on Lego Mindstorms to demonstrate and apply our framework. Both the benchmarking framework and our platform are designed with high affordability and easy reproducibility in mind.**

We plan to release our source code and construction manuals to motivate others to recreate our benchmark.

***Index Terms*—networked control system, benchmarking, Lego Mindstorms**

I. Introduction

In recent years, many devices such as phones, light bulbs, and other home appliances were equipped with microcontrollers transforming them into smart devices or cyber-physical systems (CPS). A CPS can roughly be split into two components: the actual physical system or plant, and a microcontroller managing this plant. If the microcontroller controls a process on the plant, both components form a control system. Traditionally, controller and plant are integrated into the same device forming a robust, self-contained control system. Coordination of multiple CPSes requires an exchange of information between these independent systems, leading to a network of CPSes or a networked control system (NCS).

An NCS requires the expertise of formerly separated domains, namely the control domain and the network domain. Both disciplines have established their own procedures and methodologies when it comes to comparing and rating the performance of their systems. In this paper, we provide a holistic approach towards the development of a benchmarking framework: we describe a common methodology for benchmarking the control as well as the network aspects of an NCS. We specify a common scenario to benchmark an NCS that defines the relevant key performance indicators (KPIs) to determine the quality of the control process and the network through repeatable experiments. In addition, we describe the application of this benchmark on an example platform. To that end, we currently develop an NCS based on the widely used Lego Mindstorms which we want to evaluate using our benchmark. We plan to release construction manuals and software as open source to establish a low-cost benchmarking framework for NCSes.

The remainder of the paper is organized as follows: we investigate relevant foundations for benchmarking in the control and network domain in Section II, and propose a unifying benchmark methodology for NCSes in Section III. In Section IV we describe our own open-source benchmarking framework. Section V concludes the paper.

II. Related Work

The problem of modeling the effects and constraints introduced by the network on a control system is well studied in literature [1]–[3]. A traditional, control-oriented approach to the problem is to model the network as a source of random delays and dropouts [1], [2]. A more detailed survey on the constraints introduced by the network on control systems is presented by Zhang et al. [3], together with practical applications arising from NCSes.

Besides modeling and summarizing the effects of the network on control systems, the problem has been studied with several experimental case studies on benchmark control systems. In particular, a variety of measurements has been conducted for inverted pendulums [4]–[6] (which, according to Boubaker [7], have been used since the 1950s as a control benchmark for its simple structure but nonlinear dynamics) and batch reactors [8]–[10]. All these implementations perform real-world measurements of standard NCSes to proof the validity of their contributions. However, they make use of specific hardware and software solutions, which, together with different and scattered measurement scenarios, introduce a major obstacle for reproducibility. For this reason, different techniques are difficult to compare in a single, repeatable NCS benchmark scenario.

An exemplary definition of the elements needed to enable the benchmarking of different NCSes is presented by Niemueller et al. [11]. Here, the KPIs and context of an industrial CPS are identified and presented in a holistic benchmark scenario. However, this benchmark is limited to high-level multi-robot systems and does not cover the low-level interconnections of closed-loop NCSes.

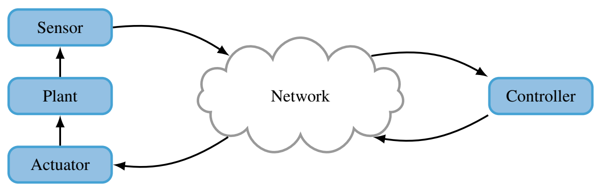

Fig. 1. Control loop of an NCS

Similarly, there exist standard benchmarks exclusively focusing on the network domain such as RFC 2544 [12] or similar benchmarks [13], [14]. Due to technical advance, the benchmarks need to be extended to reflect this progress [15]. However, these benchmarks still offer only limited insight into the behavior of applications, such as controllers which utilize those networks.

III. Framework for Reproducible NCS Benchmarking

A simple model of an NCS is shown in Figure 1. This control system involves a single controller and a single plant visualizing the typical flow of information in a control loop– additional information flows, e.g. configuration of the sensor, are neglected. Sensor and actuator of the plant are connected to the controller over an arbitrary wired or wireless network. We intentionally decided on the most basic system consisting of the minimum number of components involved, thus powerful enough to demonstrate our benchmarking framework while minimizing the chance of misconfiguration and simplify reproducing experiments.

a) Reproducibility: according to ACM [16], is the recreation of experimental results that can be achieved on three different levels. Experiments are considered *repeatable* if the same people can recreate their results, *replicable* if other persons can recreate results using the same setup, and finally *reproducible* if other researchers can recreate results using their own equipment. By defining the experiment conditions and releasing software as well as construction manuals as open-source, this paper aids the process towards reproducible experiments.

b) Benchmarking framework: Our framework defines a series of repeatable experiments. In each experiment, a set of values is measured. In the context of our benchmark, an *experiment* can be performed by measuring a real-world setup, or by obtaining values from a simulation or emulation. A whole series of different experiments might be necessary to achieve the expected behavior of the NCS. The process of identifying these conditions we call *challenge*. The results of the experiments depend on the conditions of the experiment during the time of execution. An entire set of conditions relevant to an experiment defines a *scenario*, which includes the set of parameters that are relevant for evaluation. To make the experiments and ultimately the benchmarks repeatable, replicable and reproducible, all relevant information of a scenario has to be documented in a *scenario description* specifying the

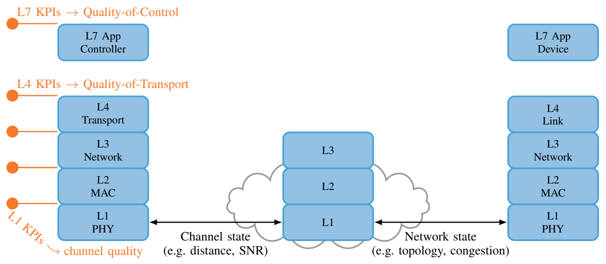
L7 KPIs → Quality-of-Control
L7 App Controller
L7 App Device
L4 KPIs → Quality-of-Transport
L4 Transport
L3 Network
L2 MAC
L1 PHY
L3
L2
L1
L4 Link
L3 Network
L2 MAC
L1 PHY
L1 KPIs ↪ channel quality
Channel state (e.g. distance, SNR)
Network state (e.g. topology, congestion)

Fig. 2. KPIs on different layers of the ISO / OSI stack

1) *application software* with a specific controller or plant,
2) *network stack* including the various protocols and technologies,
3) *network topology* describing the connectivity between the nodes of the network,
4) *physical conditions* such as distance of nodes, model and parameters of the channel model and noise floor (SNR),
5) *interference* with other nodes in the network or with external transmitters outside the network (SINR), and
6) *hardware* of the controller, the network, and plant.

The outcome of an experiment is a set of measured values and parameters. Utilizing these results we derive KPIs describing the quality of the whole system or of subcomponents of the system in a repeatable way. As the CPS and the network require their own KPIs, we decided to split these KPIs along the layers defined by the ISO / OSI model with the control system KPIs on Layer 7 (see Section III-B) and the network KPIs on Layers 4 to 1 (see Section III-A). Figure 2 depicts the vantage points of our measurements on the left side. The layers are investigated separately due to different network behavior. We consider our benchmark to be network-agnostic, i. e., we want to be able to apply it using different network technologies such as wireless LAN (IEEE 802.11), low power wireless networks (IEEE 802.15.4), Ethernet (IEEE 802.3) and others. Therefore, we specify only the interface to the highest layer we want to investigate. For instance, we specify using a UDP socket when investigating the network starting at the transport layer.

c) Challenge: The challenge is central to our benchmarking framework to identify scenarios where the NCS can operate successfully. When designing an NCS one usually expects a certain service level from the whole system, i. e., the quality of control. This quality of control can be expressed using Layer 7 KPIs. Our benchmarking framework can now help to find scenarios, i. e., a set of conditions that is able to fulfill these requirements. Based on certain desired KPIs, it may be possible to directly infer the required quality of the network KPIs, e. g. the maximum allowable delay. This reduces the set of possible scenarios for testing. In other cases, the network KPIs may not be directly inferable from the Layer 7 KPIs. For these cases, scenarios can only be determined experimentally by gradually modifying the conditions of the scenarios.

In the following, Subsection III-A describes KPIs to characterize network links, such as packet rate, loss rate, delay,

TABLE I
TABLE OF KEY PERFORMANCE INDICATORS

Sym	Description
r	packet or message rate
R	data rate [bit/s]
ϵ	packet loss rate
δ	one-way delay from source to destination
δ_s	serialization delay
δ_p	propagation delay
τ_n	inter-packet time between n-th packet and its predecessor
I_n	jitter of n-th packet and predecessor
P	bandwidth-delay product expressed in number of packets

inter-packet time, jitter, and bandwidth delay product. Subsection III-A focuses on the control system, presenting two different sets of KPIs: a set of generic KPIs applicable to a wide range of different control systems and set of specific KPIs adapted to our specific device under test–an inverted pendulum.

A. Network Domain

1) Packet rate: The packet rate r denotes the number of packets being transmitted or received. Since the maximum packet rate is limited by per-packet overheads such as processing time and medium access, it is often more important than the actual data rate, which in addition depends on the packet size. For scenario specification and analysis, the following issues are relevant:

First, the average rate at which packets are transmitted at the source node may differ from the rate packets are received due to losses. When referring to the rate at the destination, the term *goodput* may be used describing the gross data rate minus actual losses in the network.

Second, the rate may differ depending on the layer of the ISO / OSI model that is being considered: multiple messages transmitted by a controller may be aggregated at the transport layer and sent as a single frame on the link layer. Conversely, a large message may be split into multiple packets at the network layer and transmitted as multiple individual frames while being reassembled at the destination's network layer. It is therefore important to clearly state which layer one is referring to.

There exist naming conventions for the protocol data unit of different layers, e. g. *frame* on the link layer and *packet* on the network layer. On the transport layer, there is no general term as it depends on the specific implementation, e. g. datagram-oriented protocols such as UDP versus stream-oriented protocols such as TCP. Data units at the application layer can be referred to as *messages*. Unfortunately, there is no common term on the physical layer except *physical layer protocol data unit (PPDU)*, which refers to transmission on the physical layer including any signaling overhead.

These terms may be used to implicitly state the layer under consideration, but it is not always practical to do so. However, we encourage to clearly state the layer under consideration.

2) Loss rate: The loss rate denotes the fraction of packets transmitted but not received. Given the number of packets p that were successfully received and the number q of missed packets, the loss rate is given as $\epsilon = {}^{q}/_{(p+q)}$. Besides the mere rate, the pattern in which the losses occur should also be described. The Gilbert-Elliot model [17] is able to characterize such lossy wireless channels.

In practice, the loss rate can most easily be determined by the receiver using sequence numbers in each transmitted packet. In case of unreliable physical links such as wireless networks, those sequence numbers are part of the MAC header. Sequence numbers are inserted by the transmitter. When a receiver detects a gap in the sequence number of subsequent packets, it can determine the exact number of missed packets in between. Note that this demands that no re-ordering occurs after sequence numbers where chosen.

3) Delay: The term *delay* commonly refers to the total time needed to transmit (medium access plus serialization) a packet and forward it to the destination (propagation delay plus processing and buffering delays at intermediate nodes). It is often referred to as *one-way delay* in order to avoid confusion with the *round trip time (RTT)*. The latter also includes the delay for an answer from the destination back to the source, which may take a different path and thus potentially differs significantly from the delay from source to destination.

The time for medium access significantly differs depending on the actual implementation of the medium access and physical layer: while it is in the single-digit microsecond range and rather constant for switched, full-duplex Gigabit Ethernet networks [18], it is orders of magnitude larger in wireless networks due the complex medium access strategy and the shared nature of the medium [19]. If multiple transmitters contend for medium access, both the delay and its standard deviation may well be in the range of milliseconds for individual nodes.

Timestamps can be acquired from the hardware of network interfaces for transmitted or received packets. To timestamp events in software, clock counters of CPUs can be used, such as the TSC on x86-CPUs. For synchronizing timestamps across different devices, clocks need to be synchronized for instance by utilizing protocols such as PTP. These protocols offer higher accuracy if the synchronization is done via a wired connection. Therefore, an additional wired connection beside a wireless connection is beneficial during the execution of the benchmark.

The serialization delay can be approximated[1] by the frame size L and the bit rate R once control over the medium is gained, i. e., $\delta_s = L/R$. For high-speed connections such as Gigabit Ethernet, it may be neglected.

The propagation delay depends on the distance d a signal has to travel, the speed of light c, and a medium-specific constant ν: $\delta_p = {}^{d}/_{\nu c}$. The constant ν is approximately one for wireless transmissions in air and vacuum, roughly $2/3$ for copper cables, and slightly larger in fibers. For transmission in local networks, the propagation delay can obviously be

[1]Wireless networks, for instance, prepend signaling information on the physical layer at a different data rate during the physical layer convergence procedure. Therefore, the relation between frame length and bit rate is only an approximation.

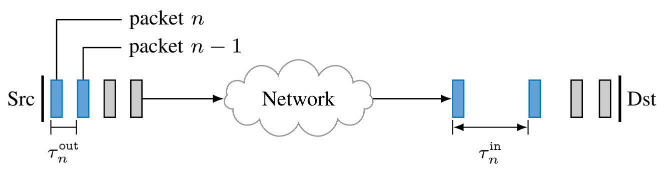

Fig. 3. Illustration of inter-packet times and jitter. Packets are transmitted by a source in regular intervals $\tau_n^{\text{out}} = \tau_{n-1}^{\text{out}}$. Variable delays in the network result in a different inter-packet time τ_n^{in} at the destination.

neglected, but may be the dominating part for long range or satellite transmissions.

Finally, processing and buffering delays differ between forwarding nodes along the path from source to destination and on the current load of those nodes. In general, those delays are difficult to quantize and particularly hard to measure without direct access to the respective node.

Whether or not individual summands of the delay may be neglected depends on the demands of the NCS. Even for our example of an inverted pendulum the delay requirements heavily depend on the used control algorithm.

Given synchronized clocks at source and destination, the times t_n^{out} when the n-th packet is transmitted and t_n^{in} when it is received are known. The one-way delay is then given as $\delta_n = t_n^{\text{in}} - t_n^{\text{out}}$. However, the transmit and receive times are still subject to noise due to processing delays on both sides. Depending on the required time resolution, it can be a challenging task to determine the delay even if both source and destination are fully controllable.

4) Inter-packet time: The *inter-packet time (IPT)* at the source between the n-th packet and its predecessor is given as $\tau_n^{\text{out}} = t_n^{\text{out}} - t_{n-1}^{\text{out}}$, assuming that messages are transmitted as individual packets and neglecting processing delays and medium access; that value should directly correlate to the time between two individual messages sent by an application. The IPT at the destination, however, is influenced by any variable effects contributing to the total delay. This effect is shown in Figure 3.

5) Jitter: Jitter is defined as the difference between the IPTs at the source and destination for a specific pair of packets. For the n-th packet and its predecessor we therefore obtain

$$I_n = \tau_n^{\text{in}} - \tau_n^{\text{out}} = t_n^{\text{in}} - t_{n-1}^{\text{in}} - (t_n^{\text{out}} - t_{n-1}^{\text{out}}). \tag{1}$$

If the jitter is subject to significant short-time changes, it may interfere with connection-oriented transport protocols such as TCP [20].

Note that jitter may also be negative: if packets are buffered in the network and then released as burst with smaller inter-packet time than at the source, we have that $\tau_n^{\text{in}} < \tau_n^{\text{out}}$ yielding $I_n < 0$. We further note that–in the absence of losses–the long-term averages of the IPT at source and destination must be the same. If this would not hold, the network would buffer an ever-increasing amount of packets over time.

6) Bandwidth-delay product: The *bandwidth-delay product* commonly expresses the amount of data in flight between source and destination, and is thus expressed in bit. However, it may also be used to quantify the number of packets currently in flight between source and destination. For the scope of an NCS, the latter is probably the more interesting metric as packets containing sensor data or control feedback are expected to be small and of more or less constant size. Furthermore, sensor data that arrives too late at the controller is commonly considered as loss, which is why aggregation of multiple sensor values in single packets is expected to be of little help.

To that end, the bandwidth-delay product expressed as packets concurrently in flight from source to destination is given by $P = \delta r$. We expect that simple metric to be one of the most important from a controller's perspective.

B. Control Domain

Analyzing the interplay between system robustness and performance is a key concern for control designers. Each controller is designed according to a model of the system. A model can be more or less accurate, e. g. in the case of the inverted pendulum, many control strategies are designed according to a linearized model around the equilibrium position. Thus, variation of parameters can directly affect the stability of the system. Robustness is then defined as the ability of the closed-loop system to be insensitive to component variations [21]. A quantitative measure of the system robustness is essential.

Besides guaranteeing the stability of the controlled system, a designed control strategy is also evaluated based on some performance indicators measuring the controller effectiveness [22], [23]. The system's ability to reject disturbances, to keep the control variables bounded within some thresholds, to get faster to the equilibrium, or to guarantee a small error after the transient phase are some examples of performance measures. The reader can easily deduce that robustness and performance indicators are closely related.

To the best of our knowledge, most prior suggestions for benchmarking CPSes take a high-level approach. Indeed this does seem appropriate for more complex systems where performance cannot easily be tied to specific system states (time-dependent variables). On the other hand, this limits their usability for more traditional CPSes where performance is objectively definable. We try to solve this conflict by taking a two-stage approach defining the standard quality of control techniques. We define high-level or generic KPIs applicable to a variety of CPSes. Additionally, we describe KPIs specific to our use case–the inverted pendulum.

1) Generic KPIs: We assume there exists an identifiable (analytic) system state which has to be steered according to some reference. Let this reference be equal to zero and let the system be inline with it. Then we can list the following KPIs:

- The maximal disturbance that may occur such that the state can still be steered back to the reference.
- The time it takes to steer the state back to reference after applying a specific disturbance. Here we define an environment $\pm\varepsilon$ around the reference. The reference is

considered reached when we can be sure to stay within that environment. Different definitions are possible.
- The energy needed to steer the system back to reference after a specific disturbance. Here, energy refers to the integral of the state trajectory (or some related trajectory) over time.

While these KPIs are defined with respect to disturbances, one can also use KPIs that describe the ability of the system to follow a change in reference. Here, we can list:

- The difference that remains between state and reference when the system is exposed to a specific reference, e. g. linearly growing as most systems exhibit an integrating behavior and thus do not show any difference when only exposed to a jump in the reference.
- The relative maximum value that is reached by the state trajectory when trying to follow a specific jump in reference.
- The time needed to steer the system to a new reference after a jump to that new reference.
- The energy needed to steer the system to a new reference after a jump to that new reference. Again we refer to some kind of integral over time.

2) Specific KPIs: With respect to the example of the inverted pendulum control problem, we can define the following KPIs specific to the inverted pendulum or similar systems:

- θ_m^0: the maximum initial angle of the pendulum (with zero initial speed) from which the controlled system can get to asymptotic stability.
- v_m^0: the maximum initial longitudinal speed (with initial null angle) that allows the controlled system to reach asymptotic stability.

Control performance can be evaluated by starting from different initial conditions chosen from a given set $\mathbb{X}_0$. Accordingly, $\forall x_i^0 \in \mathbb{X}_0$, the following indicators are stated:

- $\theta_{max}(x_i^0)$: starting from x_i^0, this indicator measures the maximum value assumed by the pendulum angle.
- $u_{max}(x_i^0)$: starting from x_i^0, this indicator measures the maximum value assumed by the control action, e. g. DC motor injected current.
- $T_{max}(x_i^0)$: starting from x_i^0, this indicator measures the oscillation period, defined as the maximum time difference between two consecutive oscillation peaks of θ.

Finally, a robustness indicator which measures the ability of the controlled system to react to some disturbances is presented here:

- $d_{u,max}$: maximum allowed impulse disturbance on the control variable.

All these KPIs do theoretically depend on the communication delay between controller and plant. However, due to the control loop, it is likely that in most cases the dependencies are non-linear, leaving us only with the option of simulation or experimentation.

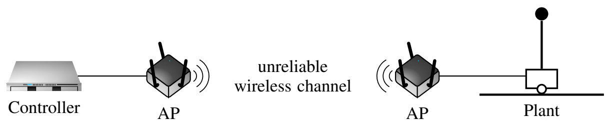

Fig. 4. Test setup for a cyber-physical network

C. Reproducibility vs. Non-Determinism

Certain KPIs might behave non-deterministically, such as the delay or packet loss. Reporting only an aggregated number, such as average or median, as a KPI does not suffice to enable repeatable experiments. Therefore, we propose to additionally report more expressive data such as entire logs or histograms allowing to model repeatable behavior. Another important feature to explain the CPS KPIs are disturbances or interferences. These should also be quantified and reported accordingly to aid the process of understanding and repeating the observed behavior of the CPS.

IV. Evaluation Platform

Our goal for the evaluation platform is a simple, low-cost platform which can easily be extended to allow others to replicate our setup and repeat our experiments. We opted for a well-known NCS setup: an inverted pendulum as shown in Figure 4. The inverted pendulum is built from Lego Mindstorms which is widely available, reasonably priced, and easily extensible either in software or in hardware.

A scenario description (see Section III) presents our setup: The OS of the inverted pendulum is the open-source Debian-based ev3dev[2]. Both *applications* running on the inverted pendulum and the controller are written in Python, an easy-to-learn scripting language. Additionally, the controller application is portable, i. e., the controller supports multiple OSes. The *network stacks* involved are the network stacks of Linux / ev3dev for the inverted pendulum and the network stack of the OS on the controller. Our applications use UDP sockets for communication. The inverted pendulum utilizes Mindstorm's USB interface for network access. From a hardware perspective, any network is supported as long as there are USB adapters available. Depending on the hardware equipment, arbitrary *network topologies* are supported. The *physical conditions* should be simple enough to be replicated by interested parties. Therefore, we use a distance of up to a few meters between controller and plant. As *interference* is hard to recreate especially for wireless networks without specialized equipment, we suggest to test in a quiet environment in a closed room without any devices utilizing the same frequencies as the NCS in the same room. In addition, we propose to measure and document the properties such as attenuation and interference. Even though that does not solve the problem of expreriment recreation, it can at least offer an explanation if repeated results differ from the original results. Additionally, the documentation would allow others capable of recreating a physical environment to replicate the experiment

[2]http://www.ev3dev.org/

results. We aim for a large degree of freedom for the *hardware* by supporting multiple OSes for the controller, utilizing USB network devices, and an easily adaptable and extensible Lego Mindstorms hardware. We even plan to support alternative microcontrollers for the inverted pendulum based on a Raspberry Pi.

In order to define the relevant information and parameters of the scenario mentioned above, different types of challenges can be applied to the system. For instance, the system can be challenged by setting different initial positions or applying different external forces to the balancing problem of the inverted pendulum's controller. Similarly, challenges can be introduced by the network considering, for instance, network cross-traffic or external Wi-Fi interference.

V. Conclusion

We presented a novel benchmark methodology based on benchmarks from both disciplines joint in NCS, namely the control domain and the network domain. This allows us to create a new benchmark tailored to the specific needs of an NCS. Therefore, we identify the relevant parameters to recreate test conditions when performing the benchmark. The result of our benchmark is reported as KPIs. We define two sets of indicators that are either used to describe the quality of control of a CPS or the quality of transport of the underlying network.

Furthermore, we present our own NCS evaluation framework based on the Lego Mindstorms platform. The platform was built with cost efficiency, flexibility, and ease-of-use in mind, leading to a framework which can easily be recreated by others fostering the reproduction of our results. Up to now we only published the outline of an NCS benchmarking framework defining the KPIs of both domains without any measurements. We intend to release an evaluation with test results for a specific NCS, the inverted pendulum, in future work. To further promote our benchmark, we plan to release our software and instruction manuals in the near future on GitHub.

VI. Acknowledgments

This work was supported by the DFG Priority Programme 1914 Cyber-Physical Networking and by the German-French Academy for the Industry of the Future.

References

[1] L. Xiao, A. Hassibi, and J. P. How, "Control with Random Communication Delays via a Discrete-Time Jump System Approach," in *Proceedings of the 2000 American Control Conference. ACC (IEEE Cat. No.00CH36334)*, vol. 3, Chicago, USA, Jun. 2000, pp. 2199–2204.

[2] L. Zhang, Y. Shi, T. Chen, and B. Huang, "A New Method for Stabilization of Networked Control Systems With Random Delays," *IEEE Transactions on Automatic Control*, vol. 50, no. 8, pp. 1177–1181, Aug. 2005.

[3] L. Zhang, H. Gao, and O. Kaynak, "Network-Induced Constraints in Networked Control Systems – A Survey," *IEEE Transactions on Industrial Informatics*, vol. 9, no. 1, pp. 403–416, Feb. 2013.

[4] A. Chamaken and L. Litz, "Joint Design of Control and Communication in Wireless Networked Control Systems: A Case Study," in *Proceedings of the 2010 American Control Conference*, Baltimore, USA, Jun. 2010, pp. 1835–1840.

[5] N. J. Ploplys, P. A. Kawka, and A. G. Alleyne, "Closed-loop Control Over Wireless Networks," *IEEE Control Systems*, vol. 24, no. 3, pp. 58–71, Jun. 2004.

[6] C. Peng, D. Yue, and M. R. Fei, "A Higher Energy-Efficient Sampling Scheme for Networked Control Systems over IEEE 802.15.4 Wireless Networks," *IEEE Transactions on Industrial Informatics*, vol. 12, no. 5, pp. 1766–1774, Oct. 2016.

[7] O. Boubaker, "The Inverted Pendulum Benchmark in Nonlinear Control Theory: a Survey," *International Journal of Advanced Robotic Systems*, vol. 10, no. 5, p. 233, Jan. 2013.

[8] M. S. Mahmoud and M. Sabih, "Experimental Investigations for Distributed Networked Control Systems," *IEEE Systems Journal*, vol. 8, no. 3, pp. 717–725, Sep. 2014.

[9] C. Lu, A. Saifullah, B. Li, M. Sha, H. Gonzalez, D. Gunatilaka, C. Wu, L. Nie, and Y. Chen, "Real-Time Wireless Sensor-Actuator Networks for Industrial Cyber-Physical Systems," *Proceedings of the IEEE*, vol. 104, no. 5, pp. 1013–1024, May 2016.

[10] J. Arajo, M. Mazo, A. Anta, P. Tabuada, and K. H. Johansson, "System Architectures, Protocols and Algorithms for Aperiodic Wireless Control Systems," *IEEE Transactions on Industrial Informatics*, vol. 10, no. 1, pp. 175–184, Feb. 2014.

[11] T. Niemueller, G. Lakemeyer, S. Reuter, S. Jeschke, and A. Ferrein, "Benchmarking of Cyber-Physical Systems in Industrial Robotics: The RoboCup Logistics League as a CPS Benchmark Blueprint," pp. 193–207, Sep. 2016.

[12] S. Bradner and J. McQuaid, "RFC 2544: Benchmarking Methodology for Network Interconnect Devices," Mar. 1999.

[13] ITU-T, "Y.1564: Ethernet service activation test methodology Recommendation," Feb. 2016.

[14] Metro Ethernet Forum, "MEF 14: Abstract Test Suite for Traffic Management Phase 1," Nov. 2005.

[15] D. Raumer, S. Gallenmüller, F. Wohlfart, P. Emmerich, P. Werneck, and G. Carle, "Revisiting Benchmarking Methodology for Interconnect Devices," in *The Applied Networking Research Workshop 2016 (ANRW '16)*, Berlin, Germany, Jul. 2016.

[16] ACM, "Artifact Review and Badging," http://www.acm.org/publications/policies/artifact-review-badging, last accessed: Mar. 23 2018.

[17] P. Ferre, D. Agrafiotis, T. K. Chiew, A. Doufexi, A. Nix, and D. Bull, "Packet Loss Modelling for H.264 Video Transmission over IEEE 802.11g Wireless LANs," in *IEEE International Workshop on Image Analysis for Multimedia Interactive Services, WIAMIS*, Montreux, Switzerland, Apr. 2005.

[18] S. Larsen, P. Sarangam, R. Huggahalli, and S. Kulkarni, "Architectural Breakdown of End-to-End Latency in a TCP/IP Network," *International journal of parallel programming*, vol. 37, no. 6, pp. 556–571, Jun. 2009.

[19] S. M. Günther, M. Leclaire, J. Michaelis, and G. Carle, "Analysis of Injection Capabilities and Media Access of IEEE[802.11] Hardware in Monitor Mode," in *IEEE Symposium on Network Operations and Management (NOMS)*, Krakw, Poland, May 2014.

[20] M. E. Jobst, S. M. Günther, M. J. Riemensberger, G. Carle, and W. Utschick, "Adaptive Suppression of inter-packet Delay Variations in Coded Packet Networks," in *IEEE International Symposium on Network Coding (NetCod)*, Sydney, Australia, Jun. 2015.

[21] K. J. Aström and R. M. Murray, *Feedback Systems: an Introduction for Scientists and Engineers*. Princeton University Press, 2010.

[22] S. Skogestad and I. Postlethwaite, *Multivariable Feedback Control: Analysis and Design*. Wiley New York, 2007, vol. 2.

[23] P. Bolzern, R. Scattolini, and N. Schiavoni, *Fondamenti di Controlli Automatici*. McGraw-Hill Italia, 1998.