In-Situ Model Validation for Continuous Processes Using In-Network Computing

Ike Kunze[*], Dominik Scheurenberg[†], Liam Tirpitz[‡], Sandra Geisler[‡], Klaus Wehrle[*]
[*]*Communication and Distributed Systems* · {kunze, wehrle}@comsys.rwth-aachen.de
[†]*Institute of Automatic Control* · d.scheurenberg@irt.rwth-aachen.de
[‡]*Data Stream Management and Analysis* · {tirpitz, geisler}@dbis.rwth-aachen.de
All authors are affiliated with *RWTH Aachen University*, Aachen, Germany

***Abstract*—The advancing industrial digitalization enables evolved process control schemes that rely on accurate models learned through data-driven approaches. While they provide high control performance and are robust to smaller deviations, a larger change in process behavior can pose significant challenges, in the worst case even leading to a damaged process plant. Hence, it is important to frequently assess the fit between the model and the actual process behavior. As the number of controlled processes and associated data volumes increase, the need for lightweight and fast reacting assessment solutions also increases. In this paper, we propose *CIVIC*, an in-network computing-based solution for *C*ontinuous *I*n-situ *V*alidation of *I*ndustrial *C*ontrol models. In short, *CIVIC* monitors relevant process variables and detects different process states through comparison with a priori knowledge about the desired process behavior. This detection can then be leveraged to, e.g., shut down the process or trigger a reconfiguration. We prototype *CIVIC* on an Intel Tofino-based switch and apply it to a lab-scale water treatment plant. Our results show that we can achieve a high detection accuracy, proving that such monitoring systems are feasible and sensible.**

I. Introduction

Ongoing initiatives, such as Industry 4.0 [1] or the Internet of Production [2], further fuel the ongoing digitalization of production industries and, e.g., enable the creation of new knowledge. One beneficiary of such data-driven approaches is industrial process control [3]. For example, model predictive control (MPC) can effectively control continuous production processes, but requires precise system models to first predict future behavior and then apply suitable actions [4].

In practice, small deviations between the model and the real process usually only (slightly) reduce the control performance of MPC. In contrast, larger differences can cause significant misbehavior of the control and ultimately lead to considerable damage to the process plant. To ensure the use of a sufficiently accurate process model, it is important to detect and handle any potentially dangerous deviations as early as possible [5].

Modern decentralized manufacturing, however, often outsources process control to powerful cloud environments as conventional field-based devices are often insufficient for advanced control and as deploying dedicated local compute resources is cost-intensive [6]. The additional latency to the cloud can become problematic for MPC as it delays potential responses to differences between the model and the process [7]. Hence, there is a strong need for a dedicated, but lightweight monitoring system on the edge that can detect such changes, e.g., by constantly monitoring the current process state and comparing it with the expected behavior to quickly trigger adequate responses.

Based on related work, we identify in-network computing (INC) as a promising solution platform. In short, INC uses the capabilities of modern networking devices to move computations to the data path, allowing earlier processing at higher processing rates compared to end-host systems. Previous work has already demonstrated the general applicability of INC to industrial scenarios [8] and to the control of robot arms [9]–[11]. While the former approach only performs single operations per packet, the latter works rely on precise models of the underlying processes, similar to MPC. Hence, they are likely also affected by changes in process behavior, but do not yet explicitly detect or handle these cases.

To address this gap, we propose *CIVIC*, a **C**ontinuous **I**n-situ **V**alidation of **I**ndustrial **C**ontrol models. *CIVIC* monitors real process communication, extracts relevant sensor and control information, and aggregates this information to describe the current process state. It then checks this state against an expected system model with the goal of identifying relevant deviations. We prototype *CIVIC* on an Intel Tofino-based switch and demonstrate its practicability by applying it to a lab-scale water treatment plant. Our experiments show that *CIVIC* can reliably detect deviations from the expected process behavior. Overall, we contribute the following:

- We identify the need for validating the system models of continuous industrial processes at run-time.
- We design *CIVIC* which monitors the communication of industrial processes to assess their system behavior.
- We demonstrate the efficacy of *CIVIC* at the example of a lab-scale water treatment plant and show that it can efficiently assess the fit of the process model.

Structure. In Sec. II, we introduce continuous industrial processes and discuss related work on INC. Sec. III then presents the design of *CIVIC* which we implement on an Intel Tofino-based switch and apply to a water treatment plant in Sec. IV. We evaluate *CIVIC* in Sec. V. Sec. VI concludes the paper.

II. Controlling Continuous Industrial Processes

Modern industrial ecosystems are heterogeneous and decentralized environments that enable dynamic task allocation to diverse resources. Leveraging this flexibility, the control of many continuous industrial processes, as, e.g., used in the chemical industry, is increasingly spatially decoupled and moved away from the plants to optimally utilize powerful hardware and satisfy ever growing quality demands. Yet, this spatial separation also raises challenges, e.g., by inducing latencies that delay control responses. Hence, it is still sensible to push *some* lightweight, time-sensitive operations to the network edge. In the following, we present continuous processes and associated challenges in more detail and further discuss how research addresses this field.

Continuous industrial processes. Continuous processes transform raw materials into finished products without interruption. For example, chemical plants utilize such processes for reactions or refinements while water treatment plants continuously filter or disinfect water. Fig. 1 shows a simple coupled tank system that provides water to a larger continuous water treatment pipeline [12] for which the water levels L_1-L_3 in tanks T_1-T_3 need to be controlled. In all examples, continuous and precise process control is needed to ensure operational efficiency and consistent product quality.

Controlling and monitoring continuous processes. In practice, feedback control is the most prominent control scheme: the process "output", e.g., the water levels L_1-L_3, is continuously monitored while "input" parameters, such as pump speed P_1 or the valves V_1 and V_2, are adjusted based on the difference between the desired and actual output. Proportional integral (PI) control is a simple feedback control that is easy to implement and sufficient for many processes. However, it is bad at handling nonlinearities or constraints, and lacks predictive capabilities, i.e., it only responds to the current state without considering future behavior, such that additional mechanisms may be needed to prevent system limit violations.

Model predictive control (MPC). MPC addresses several limitations of PI control and is preferred for complex and dynamic processes that require an adaptable and more precise control. In short, MPC solves an optimization problem at iterative time steps to calculate the optimal plant input given certain constraints. For this, it predicts the process behavior over a given horizon based on a process model and then considers the current state, the predicted future states, and desired objectives to determine optimal control actions with respect to a cost function [4]. However, its effectiveness depends on the accuracy of the model: if it poorly represents the actual process dynamics, the control performance can suffer. Additionally, wear of or damage to production plants often change the actual behavior. Hence, regular model validation and refinement are important for maintaining the accuracy of the process model.

Validating the process model in situ. Feedback control in general already includes monitoring the controlled variables. Validating the process model further requires monitoring other process parameters that provide additional information on the process state and whether changes occurred [13]. In particular, smaller changes might only require model updates while larger changes might lead to unsafe states or behavior as, e.g., a severe malfunction of pump P_1 might require a system shutdown. Hence, such safety-critical states need to be

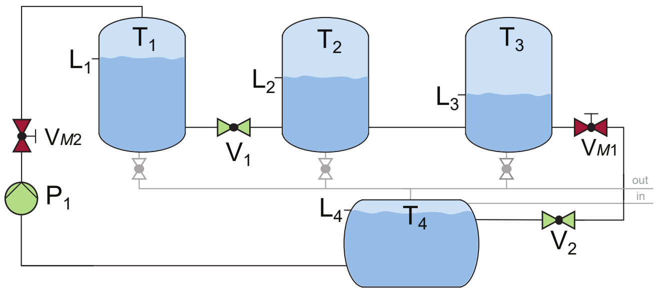

Fig. 1. The coupled tank system uses pump P_1, and valves V_1/V_2 to control the water levels L_1-L_3. The manual valves V_{M1}/V_{M2} can emulate faults.

detected as quickly as possible. However, existing approaches for plant-wide monitoring of large-scale industrial production plants require powerful hardware and perform time-intensive computations. Since these solutions rely on sensor fusion and cause large data volumes that need to be transmitted and processed [14], they delay any potential reaction, especially when spatially separated from the process [15], [16]. As a new solution space, research investigates monitoring and analyzing certain process characteristics close to the process on programmable network devices (PNDs).

In-network computing (INC) and related work. INC refers to deploying functionality on PNDs that can process large data volumes at high rates but are often limited in their processing capabilities as, e.g., multiplications are typically only possible with a fixed second factor and involving powers of two [17]. Yet, their privileged position on the data path and close to the data source makes PNDs an attractive target for processing massive industrial data streams with strict latency requirements. Approaches on protocol level, e.g., utilize the structure of industrial protocols to filter out messages and reduce traffic [18], but ignore the actual payload. Payload-focused proposals inspect individual packets to perform simple computations, such as coordinate transformations [8] or data stream processing tasks [19]. PNDs can also be used to react to data in individual packets, e.g., by issuing alerts (if monitored sensor readings violate thresholds [20]) or even stopping machines [9]. While computations and reactions on the level of individual sensor readings already demonstrate the usefulness of PNDs for industrial data processing in general and the detection of error cases specifically, industrial processes are often characterized by long-term dependencies, across series of readings. Laki et al. [10] use a PND to plan and continuously control the trajectory of a robot by implementing PID-like control functionality. Kunze et al. [21] monitor cyclic manufacturing processes to detect instabilities by taking advantage of the repeating signal pattern of this class of processes. However, this approach is not feasible for continuous processes, which we consider in this paper.

In summary, there is a strong need for dynamic approaches capable of validating models of continuous industrial processes at run-time. While INC is a potential fit, no work has yet addressed this topic. Hence, in this paper, we propose *CIVIC*,

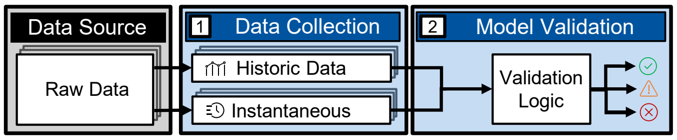

Fig. 2. *CIVIC* consists of two components: its $\boxed{1}$ data collection unit collects instantaneous and long-term process information which the $\boxed{2}$ model validation unit uses to assess the fit of the process to the given model.

a **C**ontinuous **I**n-situ **V**alidation of **I**ndustrial **C**ontrol models which compares process behavior to process models using INC to enable model refinement and emergency reactions.

III. Continuous In-situ Validation of Industrial Control models (*CIVIC*)

A continuous in-situ validation of process models can help to ensure a high process control quality. In particular, control schemes heavily relying on these models, such as MPC, can significantly benefit while validation results can also be used for triggering warnings or emergency shutdowns if there is a significant system misbehavior. *CIVIC* uses INC to provide such an in-situ validation for continuous production processes.

Design overview. In essence, *CIVIC* continuously monitors periodic process communication, extracts relevant information for monitored components, and validates the process model, e.g., via known dependencies. Conceptually, *CIVIC* consists of two components as visualized in Fig. 2: $\boxed{1}$ a data collection unit, and $\boxed{2}$ a model validation unit. In the following, we present the two components in more detail.

A. Data Collection

The data collection unit monitors periodic process communication to collect information on all process components of interest, potentially for multiple processes at the same time. Given that production plants grow in size and often feature many processes, each consisting of numerous sensors and components, one particular challenge for any data collection unit is the processing of increasingly large data volumes [14]. Addressing this challenge, we envision to deploy the data collection unit on PNDs as they are capable of handling traffic at speeds of Tbps and are specifically designed for extracting specific (bit-level) information on a per-packet basis [22].

What information to extract? The scope and form of the extracted information depend on the monitored process and the desired validation steps. In particular, one might be able to validate some aspects of a process using instantaneous data, e.g., values from a single sensor reading, while other aspects might require longer term dependencies. For this purpose, *CIVIC* uses registers to provide both capabilities, i.e., *instantaneous* information as well as a *history* of up to n previous readings where n can differ for different information sources or even for the same information source when used in different processes. Based on the collected information, *CIVIC* then validates the underlying system model.

B. Model Validation

The model validation unit checks the monitored information for known properties to detect relevant deviations. Accounting for the diversity of continuous processes, we do not propose a single one-fits-all solution, but instead aim to provide different validation modules that are applicable to different scenarios. Hence, we next non-exhaustively illustrate the possible range of mechanisms with *some* possible forms of validation.

Validation logic. A simple validation is to check specific system thresholds or combinations thereof, such as minimum or maximum fill levels of water tanks, pump speeds, or valve positions (cf. Fig. 1). More advanced validation schemes could combine the simple checks with longer term assessments, e.g., concerning the rate of water increase or decrease in the tanks, to allow for a broader understanding of the system. For example, if P_1 is turned off and V_1 is closed, we would not expect any changes to the fill level L_1 while opening V_1 should induce a known water discharge profile. Overall, concrete validation steps are highly process-dependent.

Validation complexity. Validation logic can also significantly differ in complexity. The mentioned threshold checks, e.g., only require simple comparisons while checking for change *rates*, e.g., requires computing slopes. At this point, the computational capabilities available on PNDs influence which forms of validation are possible. Hence, when aiming to deploy *CIVIC* for a specific process, it is important to consider which forms of validation are desired and which PNDs are available. However, even if some operations do not map to the fast data plane, more expressive operations can be executed on the slower control plane at the cost of higher delays [21].

Severity levels. Based on the validation, *CIVIC* needs to take appropriate action. Minor deviations from the process model typically only slightly impact the system such that immediate action is beneficial but not critically important. Hence, raising warnings is often enough, e.g., to trigger a reconfiguration. In contrast, larger deviations can hint at imminent critical damage and the system should be immediately brought to a safe state. Overall, we identify at least three validity levels that *CIVIC* needs to distinguish: (i) normal operation, (ii) small deviations ("warning"), and (iii) large deviations ("error").

In the following, we demonstrate the potential of our considerations by applying *CIVIC* to a lab-scale water treatment plant using a real PND as our deployment platform.

IV. Monitoring a Water Treatment Plant

We implement *CIVIC* in P4 on an Intel Tofino switch [22] and use it to monitor a lab-scale water treatment plant used for research [12] and teaching purposes. In the following, we present the plant and discuss how we deploy *CIVIC* using the data collection and model validation modules shown in Fig. 3.

A. Water Treatment Plant

Our water treatment plant comprises several stations, each with dedicated equipment to implement a specific continuous operation, such as water level control, heating, or pH adjustment. In this paper, we focus on the first station of the plant:

a coupled tank system as depicted schematically in Fig. 1. It consists of four distinct tanks (T_1-T_4) and the primary control aims to achieve and sustain specific water levels (L_1-L_3) within the upper tanks. For this, one pump (P_1) and two valves (V_1,V_2) are controlled automatically via an MPC while additional valves V_{M1} and V_{M2} can be adjusted manually.

System behavior. The upper tanks T_1-T_3 are positioned at the same height. Since water not only flows into the succeeding tank but also affects the preceding tank, e.g., via backflow due to sweeping movements, T_1-T_3 are interdependent. We control the water flow into T_1 via the speed of pump P_1 and the water flows T_1 to T_2 and T_3 to T_4 via pneumatic ball valves V_1 and V_2, each equipped with position controllers. Overall, the behavior of the plant is characterized by the (nonlinear) interaction of water levels, valve states, and pump activity.

System model. The properties of the coupled tank system have been approximated via empirical investigations for which we have performed a dead-time measurement with regard to P_1 and the impact of V_1 and V_2, yielding nonlinear characteristic maps of the components and resulting in a higher-order nonlinear system. However, it is possible to create a linear system model by linearization at suitable operating points, which can describe a large part of the operating range sufficiently well.

B. Data Collection

CIVIC extracts the current status of each component of the water treatment plant from periodic process control messages. In particular, we track the current water levels (L_1-L_4), the valve states (V_1,V_2), and the pump speed of P_1 (cf. Fig. 3).

Data collection on Tofino. We extract individual values from control messages via Tofino's parser, treating the entire message as a packet header. These instantaneous values are directly available for model validation. For the main control variables L_1-L_4, we keep a longer history via dedicated ring buffers R_i for each variable. They have different sizes and are each implemented using two registers: one holding a pointer to the current index, the other storing the actual data (cf. [23]).

C. Model Validation

CIVIC uses instantaneous and long-term data provided by the data collection unit for validating the system behavior. As the water treatment plant's model specifically describes the dependencies between different valve states, water levels, and pump speeds, we focus on these aspects for our validation.

Inhibited water flow. Most misbehavior in our plant affects the water flow between different tanks. In particular, we typically expect specific water movement depending on the overall system state, i.e., the fill levels of the tanks, the state of the valves, and the speed of the pump. Hence, we set up *CIVIC* to monitor the water movement between the tanks over longer timespans while we use instantaneous information on the control parameters to assess the actuator state (cf. Fig. 3).

Monitoring water flows. There are different possible strategies for monitoring the water movement. We opt for tracking the water levels in each tank over operating windows of n control messages after which we determine volume changes by performing a linear regression on the entire window to obtain a smoothed average volume change. In our initial empirical system assessment, we have already derived three baseline slopes, one for each of the validity levels defined in Sec. III-B. Hence, we can assess the adherence to the model by comparing the slope derived via linear regression with the three validity slopes where we choose the closest one as the validity class.

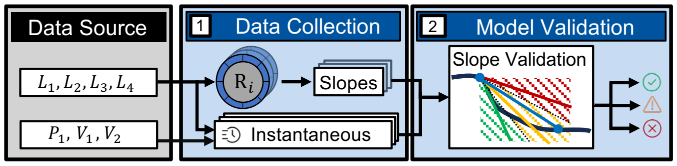

Fig. 3. Our prototype $\boxed{1}$ tracks instantaneous control data (P_1,V_1,V_2) and long-term information on L_1-L_4. $\boxed{2}$ We compare the long-term slopes against given references when the instantaneous data matches certain thresholds.

Model validation on Tofino. While our assessment logic is straightforward, linear regressions over longer windows are usually infeasible on PNDs. Hence, we adapt our methodology as follows: instead of a full regression, we approximate the slope by subtracting the latest value in the window from the oldest one. We then determine the validity class by comparing the resulting slope with baseline values using a dedicated P4 table with range matches configured with specific ranges between the baseline slopes (shared areas in Fig. 3).

Decision Logic. For this paper, we configure *CIVIC* to only label outgoing packets with the current detection state. In the future, we envision to feed this information back into the process control, e.g., to switch between different available system models depending on the system behavior.

In the following, we evaluate the performance of *CIVIC*.

V. Evaluation

CIVIC is designed to validate that the behavior of our water treatment plant follows expected behavior. For evaluating its efficacy, we control the plant assuming nominal operation, deliberately introduce errors in the system, and then evaluate how well and fast *CIVIC* can detect the corresponding deviations in system behavior. In the following, we present our evaluation methodology in more detail before presenting our results.

A. Methodology

We evaluate our prototype based on real data from our water treatment plant collected during several experiments.

Process control. In the experiments, we control the plant with a linear MPC scheme that we have extensively evaluated in previous work [12], allowing for a responsive and precise control of the system. In short, we linearize the system model (cf. Sec. IV-A) at a specific operating point that is partially characterized by the fill levels of the upper tanks (L_1-L_3), and then discretize the model for a sampling rate of 10 Hz.

Communication setup. The coupled tank system is equipped with dedicated sensors and actuators that are connected to an S7 programmable logic controller (PLC) serving as a pure data gateway. The actual control runs on a dedicated computer that

TABLE I
CLASSIFICATION SCORES OF *CIVIC* IN THE CLOGGED PIPE SETTING WITH ADJUSTED START CONDITION.

Class	Precision	Recall	F1	Accuracy
Normal	1.0	0.99	1.0	1.0
Warning	1.0	0.95	0.98	0.98
Error	0.97	1.0	0.98	0.98

is connected via Profinet and communicates via UDP. We place *CIVIC* between the PLC and the control computer such that it can operate on actual real-world control messages.

Datasets. Our experiments cover different fault states as well as nominal behavior of the water treatment plant. In particular, we emulate faults of pump P_1 and of the pipe draining T_3 by partially closing the manual valves V_{M1} and V_{M2}. During the experiments, we capture the periodic control traffic from our gateway PLC to our control computer which includes all sensor values and the current configuration of the control parameters. We capture dedicated traces for "training" (five runs) and testing (ten runs) in each of the considered scenarios.

Evaluation. To enable reproducibility, we replay the recorded traffic from a dedicated machine M_1 to one Tofino-based switch running *CIVIC*. The switch performs the validation, labels the traffic accordingly, and then forwards it to machine M_2 where it is captured for analysis. We assess *CIVIC*'s performance by comparing its labeling with the expected results of the scenario, i.e., if we do not emulate a fault, *CIVIC* should identify normal system operation. As *CIVIC* yields unambiguous results, we replay each setting once.

B. Identifying Clogged Pipes

A frequent fault in our plant is a clogging of pipes due to deposits which inhibits the possible water flow and worsens process control. Our first evaluation case covers this scenario.

Fault emulation. We emulate a clogged pipe by a partial closure of manual valve V_{M1}, inhibiting the water flow from T_3 to T_4. Based on the severity levels defined in Sec. III-B, we distinguish three cases: (i) *normal* operation where V_{M1} is completely open, (ii) a *warning* scenario where we slightly close V_{M1} to $30\,\%$, and (iii) an *error* scenario where we close V_{M1} to $55\,\%$, significantly changing the system state.

Validation logic. For detecting the clogged pipe, we trigger a water discharge from T_3 via V_2 and analyze the corresponding volume changes over an operating window of 20 sensor intervals. As a discharge requires a sufficient initial volume in T_3 and a sufficiently open valve, we only apply this logic if the volume in T_3 (L_3) is larger than $3\,\mathrm{L}$ and if V_2 is at least $80\,\%$ open. We realize this logic by combining the instantaneous threshold and long-term slope validation primitives of *CIVIC*.

Baseline results. In a first step, we train *CIVIC*'s slope validation primitive on a training set of five runs collected for an initial volume in T_3 of $7.5\,\mathrm{L}$. For this, we manually optimize *CIVIC*'s range matches such that we achieve perfect results on the training set. We then evaluate the corresponding performance on a test set of ten runs. We find that *CIVIC* still achieves a perfect assessment in these runs, i.e., all cases are classified correctly without any false or missed classifications as indicated by classification scores of 1. However, the system can have different initial configurations and we have only picked a single one. Hence, to study *CIVIC*'s broader applicability, we decide to also test a different initial state.

Changing the start condition. We repeat the experiment but change the initial volume in T_3 to $6.5\,\mathrm{L}$ which reduces the slope of the discharge profile due to smaller water pressure. In this setting, we only collect a test set of ten runs. Table I shows the classification scores when using *CIVIC* with its configuration from above. As can be seen, we still achieve a high performance and correctly identify all error cases (indicated by a recall of 1). While we misclassify some normal and warning cases as the next higher severity, we argue that being a bit overcautious is desirable from a process safety standpoint. Hence, overall, *CIVIC* performs well and its configuration can also cope with smaller system setup changes.

TABLE II
CLASSIFICATION SCORES OF *CIVIC* IN THE FAILING PUMP SETTING.

Class	Precision	Recall	F1	Accuracy
Normal	1.0	1.0	1.0	1.0
Warning	0.98	1.0	0.99	1.0
Error	1.0	1.0	1.0	1.0

C. Detecting a Failing Pump

Another common fault is a failing pump, e.g., moving less volume than normal, which we use as our second case.

Fault emulation. Similar to the clogged pipe, we emulate a fault of P_1 by partially closing V_{M2}, inhibiting the water flow from T_4 to T_1. We again use three severity levels: (i) *normal* operation with V_{M2} open, (ii) *warning* where V_{M2} is closed to $20\,\%$, and (iii) *error* where V_{M2} is closed to $40\,\%$.

Validation logic. Depending on P_1's speed, we expect a corresponding volume increase in T_1. Hence, we validate P_1 by tracking the slope of L_1 when filling an almost empty T_1. As this change is slower than the water discharge, we use a longer operating window of 90 sensor intervals and only perform the validation if P_1 has a high speed ($P_1 > 99\,\%$) and if the water level in T_1 is in a certain corridor ($5\,\mathrm{L} < L_1 < 8\,\mathrm{L}$).

Results. Similar to Sec. V-B, we first train *CIVIC*'s slope validation primitive on a training set of five runs and then evaluate its performance on a test set of ten runs. Table II shows the classification performance results. As can be seen, *CIVIC* achieves high classification quality for detecting a failing pump, too. However, similar to the clogged pipe setting, we again observe a few warning cases that are misclassified as errors, i.e., *CIVIC* once more shows overcautious behavior.

D. Discussion

CIVIC is designed to validate the behavior of continuous processes and our evaluation demonstrates its general capabilities and that it can effectively track a real process. In the following, we shortly discuss possible extensions of *CIVIC* and how it can be directly embedded into the process control.

Embedding *CIVIC* into process control. *CIVIC* can detect if processes no longer comply with their expected behavior.

While we currently only label packets, there are different options for integrating *CIVIC* into the process control. For example, we could analyze different system states upfront and create unique models for each of them. Based on *CIVIC*'s assessment, we could then quickly change between the models, always choosing the best match. Similarly, there could be different control parameterizations, e.g., conservative and more aggressive ones, and *CIVIC*'s assessment could be used to choose among them. Finally, more complex embedding could directly leverage the output of *CIVIC* for the process control.

Severity levels. In this work, we experiment with three basic fault severity levels. In practice, however, there could be a larger range of such levels, each having a different implication on the overall process health. Additionally, more levels would also enable more configuration options when embedding *CIVIC* into the process control. Hence, in future work, we aim to dive deeper into possible fault states.

Application scenario. In this paper, we apply *CIVIC* to a lab-scale water treatment plant which we can monitor using the slope validation module. This allows us to verify that our concepts work as intended and can operate on real process communication. A wider selection of validation modules for *CIVIC* would extend support to a broader range of processes.

VI. Conclusion

Modern industrial control approaches increasingly rely on complex models of the controlled systems. While such data-driven concepts enable an efficient and highly performant control as long as the model accurately reflects the system behavior, deviations can impair the control quality or even lead to substantial damage of the controlled system. Hence, there is a strong need for a continuous validation of the system behavior during operation to enable quick reactions to any system changes. However, the increasing data volumes in industrial systems pose large challenges for any such validation as the data needs to be transmitted and processed.

Addressing this pain point, we propose *CIVIC*, a **C**ontinuous **I**n-situ **V**alidation of **I**ndustrial **C**ontrol models using in-network computing (INC). *CIVIC* validates the system behavior based on instantaneous and long-term process information which it checks for known dependencies using dedicated validation logic modules. We demonstrate *CIVIC*'s efficacy by applying it to a lab-scale water treatment plant and show in our evaluation that *CIVIC* is capable of accurately detecting different failure states which can then be used to raise alerts or trigger a plant reconfiguration. Overall, *CIVIC* shows that INC can be sensibly deployed for supporting process control.

Acknowledgment

Funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy – EXC-2023 Internet of Production – 390621612.

References

[1] H. Lasi et al., "Industry 4.0," *Bus. Inf. Syst. Eng.*, vol. 6, no. 4, 2014. [Online]. Available: https://doi.org/10.1007/s12599-014-0334-4

[2] J. Pennekamp et al., "Towards an Infrastructure Enabling the Internet of Production," in *IEEE ICPS*, 2019. [Online]. Available: https://doi.org/10.1109/ICPHYS.2019.8780276

[3] A. Rüppel et al., "Model-Based Controlling Approaches for Manufacturing Processes," in *Internet of Production: Fundamentals, Applications and Proceedings*. Springer, 2023, pp. 1–26. [Online]. Available: https://doi.org/10.1007/978-3-030-98062-7_7-1

[4] M. Morari and J. Lee, "Model predictive control: Past, present and future," *Comput Chem Eng*, vol. 23, no. 4, 1999. [Online]. Available: https://doi.org/10.1016/S0098-1354(98)00301-9

[5] A. Badwe et al., "Quantifying the impact of model-plant mismatch on controller performance," *Journal of Process Control*, vol. 20, no. 4, 2010. [Online]. Available: https://doi.org/10.1016/j.jprocont.2009.12.006

[6] K. Cao et al., "A Survey on Edge and Edge-Cloud Computing Assisted Cyber-Physical Systems," *IEEE TII*, vol. 17, no. 11, 2021. [Online]. Available: https://doi.org/10.1109/TII.2021.3073066

[7] J. Rüth et al., "Towards In-Network Industrial Feedback Control," in *NetCompute*, 2018. [Online]. Available: https://doi.org/10.1145/3229591.3229592

[8] I. Kunze et al., "Investigating the Applicability of In-Network Computing to Industrial Scenarios," in *IEEE ICPS*, 2021. [Online]. Available: https://doi.org/10.1109/ICPS49255.2021.9468247

[9] F. Cesen et al., "Towards Low Latency Industrial Robot Control in Programmable Data Planes," in *IEEE NetSoft*, 2020. [Online]. Available: https://doi.org/10.1109/NetSoft48620.2020.9165531

[10] S. Laki et al., "In-Network Velocity Control of Industrial Robot Arms," in *USENIX NSDI*, 2022. [Online]. Available: https://www.usenix.org/conference/nsdi22/presentation/laki

[11] Z. Wang et al., "Industrial Knee-jerk: In-Network Simultaneous Planning and Control on a TSN Switch," in *ACM MobiSys*, 2023. [Online]. Available: https://doi.org/10.1145/3581791.3596836

[12] D. Scheurenberg et al., "Data Enhanced Model Predictive Control of a Coupled Tank System," in *IEEE/ASME AIM*, 2022. [Online]. Available: https://doi.org/10.1109/AIM52237.2022.9863287

[13] S. Yin et al., "Real-Time Monitoring and Control of Industrial Cyberphysical Systems: With Integrated Plant-Wide Monitoring and Control Framework," *IEEE IEM*, vol. 13, no. 4, 2019. [Online]. Available: https://doi.org/10.1109/MIE.2019.2938025

[14] R. Glebke et al., "A Case for Integrated Data Processing in Large-Scale Cyber-Physical Systems," in *HICSS*, 2019. [Online]. Available: http://doi.org/10.24251/HICSS.2019.871

[15] O. Beyca et al., "Heterogeneous Sensor Data Fusion Approach for Real-time Monitoring in Ultraprecision Machining (UPM) Process Using Non-Parametric Bayesian Clustering and Evidence Theory," *IEEE T-ASE*, vol. 13, no. 2, 2016. [Online]. Available: https://doi.org/10.1109/TASE.2015.2447454

[16] Z. Gao et al., "A Survey of Fault Diagnosis and Fault-Tolerant Techniques—Part I: Fault Diagnosis With Model-Based and Signal-Based Approaches," *IEEE TIE*, vol. 62, no. 6, 2015. [Online]. Available: https://doi.org/10.1109/TIE.2015.2417501

[17] I. Kunze et al., "Tofino + P4: A Strong Compound for AQM on High-Speed Networks?" in *IFIP/IEEE IM*, 2021. [Online]. Available: https://ieeexplore.ieee.org/document/9463943/

[18] C. Györgyi et al., "In-network Solution for Network Traffic Reduction in Industrial Data Communication," in *IEEE NetSoft*, 2021. [Online]. Available: https://doi.org/10.1109/NetSoft51509.2021.9492564

[19] G. Sankaran et al., "Leveraging In-Network Computing and Programmable Switches for Streaming Analysis of Scientific Data," in *IEEE NetSoft*, 2021. [Online]. Available: https://doi.org/10.1109/NetSoft51509.2021.9492726

[20] A. Atutxa et al., "Achieving Low Latency Communications in Smart Industrial Networks with Programmable Data Planes," *Sensors*, vol. 21, no. 15, 2021. [Online]. Available: https://doi.org/10.3390/s21155199

[21] I. Kunze et al., "Detecting Out-Of-Control Sensor Signals in Sheet Metal Forming using In-Network Computing," in *IEEE ISIE*, 2021. [Online]. Available: https://doi.org/10.1109/ISIE45552.2021.9576221

[22] Intel, "Intel® Tofino™," 2023. [Online]. Available: https://www.intel.com/content/www/us/en/products/details/network-io/intelligent-fabric-processors/tofino.html

[23] I. Kunze et al., "Tracking the QUIC Spin Bit on Tofino," in *EPIQ*, 2021. [Online]. Available: https://doi.org/10.1145/3488660.3493804