# Decentralized Autonomous Architecture for Resilient Cyber-Physical Production Systems

Laurin Prenzel
*Technical University of Munich, Germany*
Munich, Germany
laurin.prenzel@tum.de

Sebastian Steinhorst
*Technical University of Munich, Germany*
Munich, Germany
sebastian.steinhorst@tum.de

***Abstract*—Real-time decision-making is a key element in the transition from Reconfigurable Manufacturing Systems to Autonomous Manufacturing Systems. In Cyber-Physical Production Systems (CPPS) and Cloud Manufacturing, most decision-making algorithms are either centralized, creating vulnerabilities to failures, or decentralized, struggling to reach the performance of the centralized counterparts. In this paper, we combine the performance of centralized optimization algorithms with the resilience of a decentralized consensus. We propose a novel autonomous system architecture for CPPS featuring an automatic production plan generation, a functional validation, and a two-stage consensus algorithm, combining a majority vote on safety and optimality, and a unanimous vote on feasibility and authenticity. The architecture is implemented in a simulation framework. In a case study, we exhibit the timing behavior of the configuration procedure and subsequent reconfiguration following a device failure, showing the feasibility of a consensus-based decision-making process.**

***Index Terms*—Autonomy, Decision-Making, Resilience, Consensus, Manufacturing Systems**

## I. Introduction

Autonomy is a desirable quality in many types of systems. In the automotive sector, autonomy saves cost and time, and prevents fatal accidents. In a more general sense, autonomous systems are able to "*learn, evolve, and permanently change their functional capabilities as a result of the input of operational or contextual information*" [1]. In Cyber-Physical Production Systems (CPPS), this ability to evolve and adapt has been anticipated in the works on *self-x production (self-evolvable, self-reconfigurable, self-diagnosing, ⋯)* [2]. In Cloud Manufacturing, on the other hand, a centralized scheduling authority dynamically pairs users and providers in a service-oriented architecture on a higher abstraction level than CPPS [3], [4].

As identified by [5], the necessary real-time decision-making mechanisms with respect to production planning in manufacturing systems are difficult. Nevertheless, real-time adaptability, especially when facing failures or disruptions, is critical for any autonomous manufacturing system [6], [7]. As a result, most current approaches either focus on the optimization problem, using algorithms such as Genetic Algorithms [8], or focus on dynamic real-time production planning using multi-agent systems [9]. While centralized approaches are able to solve arbitrarily complex optimization problems, the necessary orchestrators represent single points of failure. Furthermore, with cloud computing and manufacturing, a production plan may be generated by potentially untrusted devices. On the other hand, fully-decentralized approaches, such as multi-agent systems, struggle to find the optimal production plan. This paper focuses on the problem of how centralized optimization algorithms can be used in a resilient and decentralized architecture for the autonomous generation and application of production plans. In contrast to Cloud Manufacturing, we anticipate a production plan on a lower abstraction layer in which safety- and timing related properties must be guaranteed.

The authors acknowledge the financial support by the Federal Ministry of Education and Research of Germany (BMBF) in the framework of ReMiX (project number 01IS18063B). With the support of the Technische Universität München – Institute for Advanced Study, funded by the German Excellence Initiative and the European Union Seventh Framework Programme under grant agreement n° 291763.

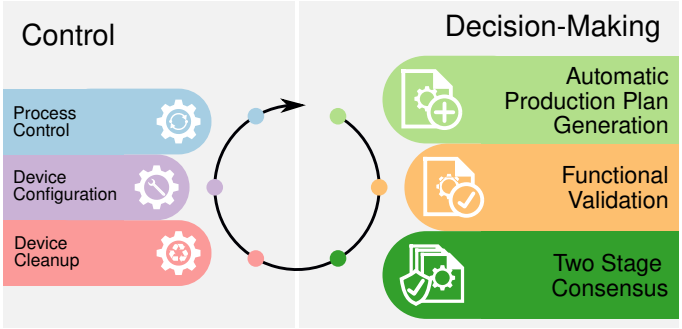



Fig. 1. Lifecycle of the autonomous system architecture, separated by decision making tasks and control tasks.

Therefore, we propose a novel decentralized system architecture for CPPS, combining an optimization algorithm with a decentralized validation and consensus framework. The phases of the lifecycle are depicted in Figure 1. The production plan, which details the mapping of devices and tasks, is generated automatically. We apply a functional validation of the plan and a two-stage consensus algorithm to provide autonomy, resilience, and real-time decision-making capabilities to a decentralized architecture. The architecture is implemented in a custom simulation framework that enables a detailed, quantitative timing analysis and provides further opportunities for research in optimization, decentralized decision-making, and validation. A case study exhibits the configuration and reconfiguration behavior in response to a device failure, showing the feasibility of our architecture and the two-stage consensus. The contributions of this paper are:

- We propose a novel system architecture for CPPS in which we define an autonomous system operation featuring an automatic production plan generation, functional validation, and a custom consensus algorithm.
- Specifically, we develop a two-stage consensus algorithm with a majority-based check on optimality and safety, and a unanimous consensus on authenticity and feasibility.
- We present a flexible simulation framework allowing a detailed timing analysis, and we implement a case study exhibiting the timing behavior of a configuration and reconfiguration cycle following a device failure (Section III).

Section IV expands on the interfaces of this architecture with the existing literature. Section V points out future research directions.

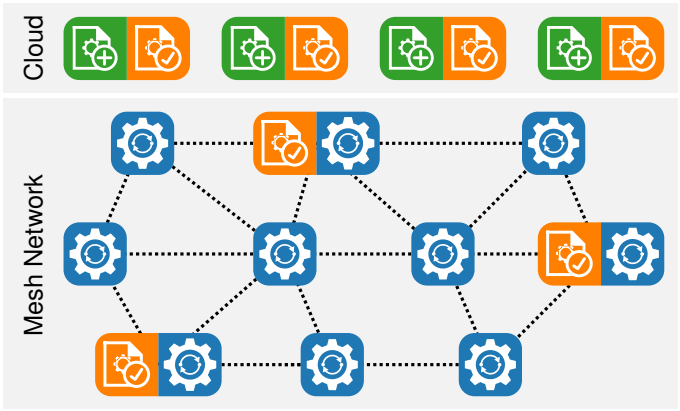



Fig. 2. Hardware architecture featuring three types of heterogeneous devices. Devices with additional computational resources can participate in the validation, whereas the production plan generation is outsourced to the cloud.

## II. Methods

We propose a novel system architecture for CPPS featuring three distinct phases:

1) **Automatic Production Plan Generation:** We automatically generate the production plan based on a system description and formalized specifications.
2) **Production Plan Validation:** The timing, safety, and functional correctness of the generated production plan are validated.
3) **Decentralized Two-Stage Consensus:** The decision-making process is built on a two-stage consensus algorithm, featuring a majority agreement on the safety and optimality and a unanimous agreement of all executing devices on the feasibility and authenticity of the plan.

The hardware design of the architecture is visualized in Figure 2. We consider a mesh network of heterogeneous devices with varying functionalities and computational capabilities. The generation of the production plans is outsourced to cloud devices, whereas the functional validation can be performed on the computationally stronger devices in the mesh network as well.

### A. Automatic Production Plan Generation

The automatic production plan generation is performed by an optimization algorithm. The inputs to the optimization are a formalized system description and one or multiple product specifications. The goal of the optimization is to find a production plan that uses the devices in the system description to fulfill (ideally) all product specifications. This problem can be formalized as a version of the Job Shop Scheduling problem, to which a great number of solutions already exists [10]. In our system, we use a greedy algorithm to find a suitable plan, although different algorithms can be plugged in.

To enable not just configuration but reconfiguration, a procedure similar to the one proposed by [11] can be used, in that the production plan generation can be coupled with the calculation of feasible migration routes. These migration routes handle leftover workpieces and lead to a stateful reconfiguration, in contrast to resetting the system to an initial state before applying the new production plan.

### B. Production Plan Validation

The new proposed production plan must be validated. Given that it may be generated remotely, this plan is not immediately trustworthy. Thus, the validation can be used to generate this trust [12].

The validation may encompass the verification of various timing- and safety-related properties. Since the inputs of the generation (system description, specifications) are openly available, they can be used in the validation procedure as well, leading to a more meaningful validation. Additional complexity is introduced by considering a stateful reconfiguration, in which the system state is (partially) preserved during a modification of the production plan. Depending on the expressiveness of the system description and specifications, a formal verification may be feasible, e.g. to verify safety-critical time constraints. The validation can be performed in a distributed manner, with devices validating a hierarchical subcomponent of the overall production plan.

### C. Decentralized Two-Stage Consensus

The two-stage consensus starts when a new production plan is proposed. The first stage synchronizes the results of the functional validation between the involved devices. During this stage, unsafe or functionally incorrect plans are filtered out, and a new plan may be selected based on its optimality. The second stage empowers all devices affected by the new plan to veto the change.

*1) Majority Consensus:* The first consensus uses majority voting to select the most suitable new plan after performing an in-depth functional validation, as detailed in the previous subsection. Given the computational complexity of this validation, this consensus is formed between the computationally stronger devices. A majority vote is chosen over a unanimous vote, since the functional validation leads to the same result on all devices.

*2) Unanimous Consensus:* The second consensus requires all affected devices to unanimously agree to this new plan. Every device must perform a quick feasibility analysis and verify the authenticity. Only if every device is able to fulfill this new production plan, it can be applied to all devices.

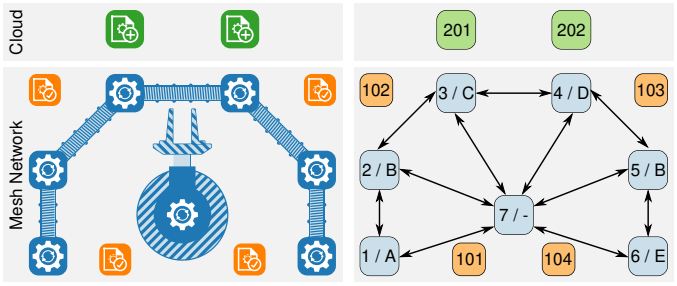
In case a consensus can not be formed in the second stage, the production plan may be changed incrementally to exclude problematic devices. Thus, a malicious device in the second stage would not be able to block the consensus indefinitely.

### D. Simulation Framework

The architecture is implemented in a custom discrete event simulation using the Python library `simpy` [13]. We can simulate arbitrary mesh networks, which are defined in a `YAML` file and can be visualized in a graphical user interface. The optimization is currently performed using a greedy algorithm on a graph representation of the system. The simulation may be executed in real-time or simulation time, and can automatically output the timestamped interactions between all system components (see for example Figure 4).

## III. Case Study

To highlight the features of the simulation and to demonstrate the feasibility of the architecture, a case study is implemented. Figure 3a depicts a conveyor belt system with an additional gripper robot that may serve as a backup to reroute workpieces in a factory. An abstract system view is visualized in Figure 3b, detailing the specific functions of the stations. Each station (`1`–`6`) has a distinct function, where station `2` and `5` are identical. Station `7` is the robot, which has no specific function




(a) Example System (b) Abstract Depiction

Fig. 3. Example system of a flexible manufacturing system with conveyor belts and a backup robot. A workpiece must follow the specification `A-B-C-D-B-E`. Functionality B is redundant in stations 2 and 5, every other function is only available once.

apart from transportation. The specification defines that every workpiece must pass all functions in the order `A-B-C-D-B-E`. Every station features a device, and there are four additional *gateway devices* (`101 - 104`) with additional computational power and no functional capabilities, and two *cloud devices* (`201 - 202`). In this case study, three contributions will be illustrated:

**System Configuration:** The system is able to autonomously generate and apply a production plan.

**System Reconfiguration:** When facing a failure, the system can reconfigure itself to circumvent the failed device and continue operation.

**Decentralized Consensus:** Using the two-stage consensus algorithm, the devices can decentrally agree on the course of action.

The simulation architecture uses a distributed ledger to share the production plans and synchronize the system state. The system starts with all devices available, configures itself, and starts producing. At a predefined point in time, `Device 2` fails, causing a reconfiguration. Because `Device 5` is still available, the system remains functional and continues production at a slower rate by using `Device 5` twice and involving the robot arm. To better visualize the timing behavior, the durations of computational tasks and communication are exaggerated.

### A. Results

The timing behavior of the simulated case study is displayed in Figure 4. Particularly interesting events are marked with numbers. At Event 0, the system description and specifications are initialized, which leads to the generation of new production plans. At Event 1, the first plan is created, and the first stage of the consensus begins, in which the gateway and cloud devices validate the plan. After a majority of devices agrees (Event 2), the second stage of the consensus takes place, which ends at Event 3. As a result, the production plan is set to valid and the devices are configured. Consequently, the second production plan, which is now outdated, is dismissed in the second phase of the consensus (Event 4). When `Device 2` fails (Event 5), this is recognized as a system change (Event 6) and leads to the generation of new plans. Eventually, a new plan passes both stages of the consensus and replaces the old plan (Event 7). Since the last plan is already outdated when it is ready, it is discarded in the first stage of the consensus (Event 8).

### B. Discussion

The case study highlights the features of the simulation, and demonstrates the feasibility of the architecture and the two-stage consensus algorithm. The system is able to automatically generate a production plan that enables the control devices to produce. The greedy graph-search algorithm is able to find a valid production plan and may in the future be exchanged for a more elaborate algorithm that can consider a multitude of factors. The control devices configure themselves when the production plan is selected and begin the process control autonomously.

After the failure of `Device 2`, new production plans are generated automatically. Following a cleanup phase, in which the state of the previous production is discarded, the involved devices configure themselves again and begin producing according to the new plan. In the future, this cleanup phase may be replaced by a stateful reconfiguration, where leftover workpieces are considered. An approach similar to the calculation of migration routes proposed by [11] may be utilized.

The feasibility of the two-stage consensus algorithm is demonstrated. The first stage, featuring a functional validation, can take place without interrupting the potentially safety- and timing-critical process control. Only the second stage requires the synchronization between all involved devices. As a result, the devices are able to decentrally configure and reconfigure themselves, showing resilience against failures. The custom simulation framework is able to perform a meaningful quantitative timing analysis and provides an extensible skeleton for future implementations.

## IV. Related Works

In the previous section we demonstrated the functionalities of the architecture in a simulation. In the following, we summarize how this architecture relates to current research.

Cloud Manufacturing promises the service-orientied pairing of manufacturing demand and supply through a cloud architecture [3]. As such, the abstraction level is much higher and the production plan does not consider safety- or timing related properties. Within Cloud Manufacturing, the issue of trust and security has been touched and may be solved by blockchain technology, but since the abstraction level is higher, safety validation is not commonly applied [4].

In CPPS, flexibility, reconfigurability, and adaptability have been considered for decades [5]. With the introduction of the *self-x* characteristics, such as *self-evolvable*, *self-reconfigurable*, *self-diagnosing*, the systems themselves are involved *in* the process of evolution, reconfiguration, and diagnosis, whereas before, it was performed *on* them [2].

Most commonly, these concepts of *self-organisation* are implemented as multi-agent systems [14]. A network of (more or less) autonomous agents is empowered to organize itself with respect to a common goal. One example, in which a multi-agent system is used for production planning, is given by [9].

By contrast, our approach combines more traditional optimization algorithms with a decentralized consensus algorithm. The ability to generate a production plan is reserved for devices with larger computational capabilities or the cloud, whereas validation can be performed more liberally, and the feasibility can be checked on every device. This allows us to open our architecture to production plans from arbitrary providers, may

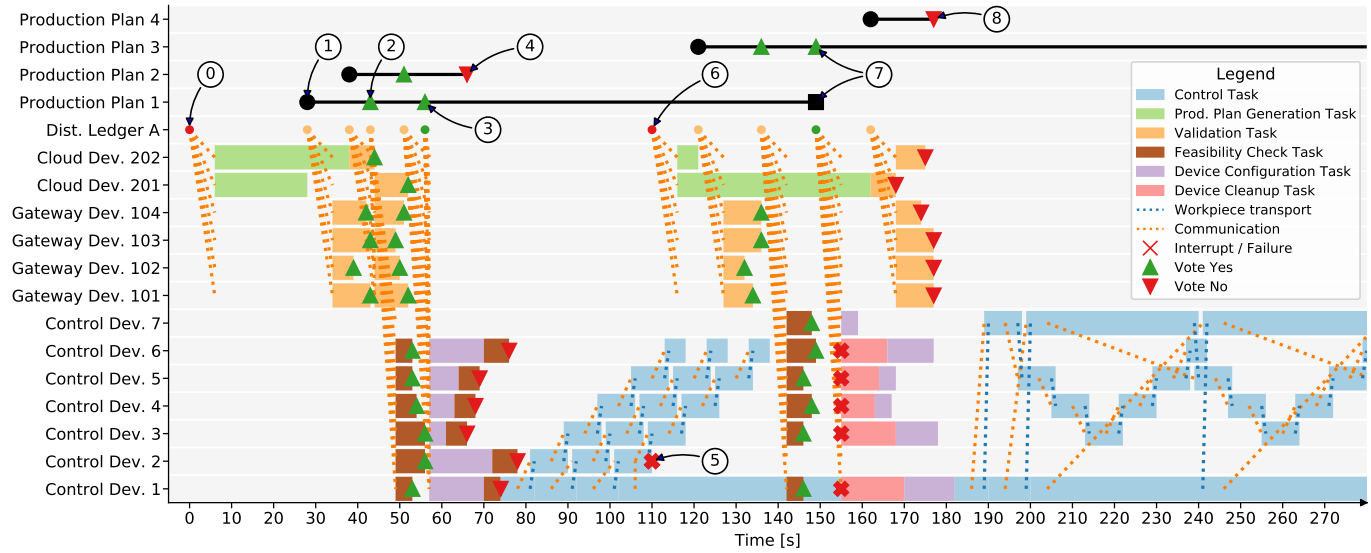



Fig. 4. Time diagram representing the result from the discrete event simulation. After system initialization (0), a production plan is generated (1). From the point when all devices agree (3), the new plan runs until a device failure appears (5), at which point new production plans are generated. Eventually, a new plan is found and applied (7). Timing is exaggerated for better visualization.

they be generated by genetic algorithms, artificial intelligence, or multi-agent systems.

With respect to service composition, [12] proposes a trust evaluation of new services by using model-based testing. The trust level is then used in the selection of services during the service composition. This enables the composition of services from untrusted origins. Similarly, our validation and two-stage consensus generate trust in a production plan from potentially unknown or untrusted origins.

Finally, with the introduction of blockchain technology, the use of distributed ledgers for software updates [15], task allocation [16], and distributed decision-making [17] could be investigated. The two-stage consensus of our work may be implemented with a distributed ledger, but is not bound to it. The formal validation as part of our consensus extends beyond the level of validation that is performed in most blockchain applications to date.

## V. Conclusion & Future Works

This paper presents a novel system architecture for resilient Cyber-Physical Production Systems. We define an autonomous system operation consisting of three phases: Automatic production plan generation, functional validation, and two-stage consensus algorithm. The architecture is implemented in a custom simulation framework that enables quantitative timing analyses. Specifically, we demonstrate the feasibility of our architecture and the two-stage consensus in a case study, exhibiting the configuration and reconfiguration behavior following a device failure.

The architecture and the simulation framework will be extended in future projects, allowing research in multiple directions, some of which are:

**Optimization:** Multi-objective production planning for the purpose of configuration and stateful reconfiguration.

**Consensus:** Decentralized decision-making algorithms in manufacturing with the goal of real-time adaption.

**Validation:** Hierarchical, formal verification of configurations and reconfigurations with respect to safety and timing.

## References


[1] P. A. Hancock, "Imposing limits on autonomous systems," *Ergonomics*, 2017.

[2] M. Onori, D. Semere, and B. Lindberg, "Evolvable systems: an approach to self-x production," *Int J. of Computer Integrated Manufacturing*, 2011.

[3] X. Xu, "From cloud computing to cloud manufacturing," *Robotics and computer-integrated manufacturing*, vol. 28, no. 1, 2012.

[4] X. Zhu, J. Shi, S. Huang, and B. Zhang, "Consensus-oriented cloud manufacturing based on blockchain technology: An exploratory study," *Pervasive and mobile computing*, vol. 62, 2020.

[5] Y. Koren, X. Gu, and W. Guo, "Reconfigurable manufacturing systems: Principles, design, and future trends," *Frontiers of Mechanical Engineering in China*, 2018.

[6] S. Jeschke, C. Brecher, T. Meisen, D. Özdemir, and T. Eschert, "Industrial internet of things and cyber manufacturing systems," in *Industrial Internet of Things: Cybermanufacturing Systems*, S. Jeschke, C. Brecher, H. Song, and D. B. Rawat, Eds. Cham: Springer, 2017.

[7] P. Skobelev and D. Trentesaux, "Disruptions are the norm: Cyber-Physical multi-agent systems for autonomous Real-Time resource management," in *Service Orientation in Holonic and Multi-Agent Manufacturing*. Springer, 2017.

[8] R. Ramezanian, D. Rahmani, and F. Barzinpour, "An aggregate production planning model for two phase production systems: Solving with genetic algorithm and tabu search," *Expert systems with applications*, 2012.

[9] N. He, D. Z. Zhang, and Q. Li, "Agent-based hierarchical production planning and scheduling in make-to-order manufacturing system," *International Journal of Production Economics*, 2014.

[10] J. Zhang, G. Ding, Y. Zou, S. Qin, and J. Fu, "Review of job shop scheduling research and its new perspectives under industry 4.0," *Journal of intelligent manufacturing*, 2019.

[11] B. Pourmohseni, S. Wildermann, M. Glaß, and J. Teich, "Hard real-time application mapping reconfiguration for NoC-based many-core systems," *Real-Time Systems*, 2019.

[12] B. Shala, U. Trick, A. Lehmann, B. Shala, B. Ghita, and S. Shiaeles, "Trust-Based composition of M2M application services," in *International Conference on Ubiquitous and Future Networks*. IEEE, 2018.

[13] K. Muller and T. Vignaux, "Simpy: Simulating systems in python," *ONLamp. com Python Devcenter*, vol. 650, 2003.

[14] D. Ye, M. Zhang, and A. V. Vasilakos, "A survey of Self-Organization mechanisms in multiagent systems," *IEEE Transactions on Systems, Man, and Cybernetics*, 2017.

[15] B. Lee and J.-H. Lee, "Blockchain-based secure firmware update for embedded devices in an internet of things environment," *The Journal of supercomputing*, 2017.

[16] T. L. Basegio, R. A. Michelin, A. F. Zorzo, and R. H. Bordini, "A decentralised approach to task allocation using blockchain," in *Engineering Multi-Agent Systems*. Springer, 2018.

[17] E. Castelló Ferrer, "The blockchain: A new framework for robotic swarm systems," in *Proc of the Future Technologies Conf (FTC)*. Springer, 2019.