# Controller-Aware Dynamic Network Management for Industry 4.0

Efe C. Balta, Mohammad H. Mamduhi, John Lygeros, and Alisa Rupenyan
Department of Information Technology and Electrical Engineering
Automatic Control Laboratory, ETH Zürich, Zürich, Switzerland
{ebalta,mmamduhi,jlygeros,ralisa}@ethz.ch

IECON 2022 – 48th Annual Conference of the IEEE Industrial Electronics Society | 978-1-6654-8025-3/22/$31.00 ©2022 IEEE | DOI: 10.1109/IECON49645.2022.9968460

***Abstract*—In this paper, we consider a cyber-physical manufacturing system (CPMS) scenario containing physical components (robots, sensors, and actuators), operating in a digitally connected, constrained environment to perform industrial tasks. The CPMS has a centralized control plane with digital twins (DTs) of the physical resources, computational resources, and a network manager that allocates network resources. Existing approaches for the allocation of network resources are typically fixed with respect to controller-dependent run-time specifications, which may impact the performance of physical processes. We propose a dynamic network management framework, where the network resource allocation schemes are controller-aware. The information about the controllers of the physical resources is implemented at the DT level, and metrics, such as regret bounds, take the process performance measures into account. The proposed network management schemes optimize physical system performance by balancing the shared resources between the physical assets on the plant floor, and by considering their control requirements, providing a new perspective for dynamic resource allocation. A simulation study is provided to illustrate the performance of the proposed network management approaches and compare their resource allocation performance efficiencies.**

***Keywords*—Network resource allocation, factory management, IIoT enabled control, cyber-physical systems**

## I. Introduction

Industrial Cyber-Physical Systems (ICPS) generally involve a layer of coupled mechanical/physical components, and a cyber layer that supports the interactions and data exchange between the components of the physical layer and additionally with the monitoring, remote supervision, and maintenance units [1]. Modern applications of ICPS require sensitive control functionalities across the physical layer, and a reliable and agile cyber layer enabling sustainable and coordinated interactions [2], [3]. Traditionally, the development of ICPS mainly coincided with the development of control architectures, e.g., the progress in deploying distributed control systems for process industries, or advancements in programmable logic controllers (PLC) for discrete manufacturing, while the communication was merely in the form of exchanging predetermined control and actuation command signals via locally installed networks infrastructure mainly for wired channels [4]. Rapid advances in communication technology and the need for higher levels of automation for efficient production led to a huge revolution in industrial manufacturing where devices, machines, and industrial components became interconnected (networked) with the support of a communication and computation network. Efficient resource allocation has been a challenging problem generally in networked control systems [5], [6], and specifically in interconnected industries [7] where quality-of-production is coupled with the quality-of-service.

The introduction of the 5th-generation mobile Internet, known as 5G Internet, together with the fast development of advanced computational methods and in-network computations (edge and fog computing) provide a plethora of opportunities for ICPS to move into a new level of autonomy, reconfigurability, and efficiency [8]. State-of-the-art networking technology can support a wide range of industries within one physically installed network infrastructure, through its virtual and programmable features [9], [10]. Additionally, software-defined and virtual components, e.g., software-defined networks (SDN), enable demand-driven, status-aware data networks that allow for customized quality-of-service, optimized for the physical components' needs [11], [12].

In the context of manufacturing, ICPS are often referred to as Cyber-Physical Manufacturing Systems (CPMS). Efficient network resource allocation among service-critical physical components is a challenging problem in a CPMS scenario, due to the time-varying resource requirements that are sensitive to the communication and computation needs [7]. Novel networking features and capabilities have led to the development of more advanced and reconfigurable resource allocation architectures [13], [14]. Learning and AI methods, software-defined resource orchestration, and virtual sub-networks have been used for efficient resource allocation in the automation and control of CPMS [14], [15], [16]. Hierarchical resource allocation mechanisms are popular for resource provisioning in industrial plants due to their flexibility in deployment and capability of being implemented in distributed fashion [17]. Nevertheless, a deployable resource allocation mechanism that supports heterogeneous industrial operations *in a controller-aware fashion* is not addressed in the literature.

The main goal of this paper is to provide controller-aware and sustainable resource allocation and networking management for the state-of-the-art CPMS. We assume that the CPMS has a centralized control plane that has access to the plant floor data through a unified data interface, see, e.g., [18], [15], and a pool of digital twins (DTs) that represent the physical plant floor through models, measurement data, and algorithmic

This work is supported by NCCR Automation, funded by the Swiss National Science Foundation through Grant no. 180545.

intelligence [18], [19], [20]. The communication/computation requirements of the physical resources vary based on a number of factors, which are monitored by the responsible DTs. We propose a dynamic network management framework, where the controller-aware requirements and run-time measurements gathered through the DTs are used by the network manager for optimal network resource allocation, according to set requirements. We explore event-based reallocation of network resources, where the events are triggered based on controller performance-aware metrics [21], [22], and online (continuous) reallocation scheme [23], and compare them with existing network allocation approaches. We demonstrate the efficiency of the proposed controller-aware approaches in a numerical case study and show that they greatly outperform static network allocation methods without run-time data from the DTs.

The remainder of this paper is organized as follows. Section II describes the CPMS model and the problem formulation. We propose our controller-aware resource allocation architecture in Section III. In Section IV the simulation results are demonstrated and the paper is concluded in Section V.

## II. CPMS Model and Problem Formulation

We consider a CPMS scenario wherein multiple dynamically heterogeneous components are operating to perform given industrial tasks including sensing, monitoring, and actuation. The CPMS setting is schematically illustrated in Fig. 1 with the central controller including a DT module in the baseline setting. The physical systems make up the plant floor of the CPMS. Each physical system is equipped with various sensors, industrial internet-of-things devices, etc. A unified communication interface collects data from the heterogeneous devices from the plant floor, and makes it available to the software modules in the central controller, e.g., a southbound interface as proposed in [15], [18]. The central controller has a DT module that includes the local DTs, databases, and necessary application interfaces. The network manager utilizes the data from the DTs and the plant floor to access the computation resources and optimize the necessary allocation. The computational resources may be part of the centralized controller plane, edge devices on the plant floor, or distributed resources connected with 5G networks. Using the network allocation, the DTs compute the control inputs and send them to the physical systems on the plant floor. The DTs are tasked with monitoring the physical assets and processes against task requirements and providing control actions to the plant floor. The network requirements of physical resources and processes are evaluated by the DTs and shared with the network manager so that the network resources can be optimally allocated to maximize performance. The proposed framework builds a network resource allocation layer on top of the given baseline CPMS configuration including the DTs for process monitoring and data analytics.

Control inputs for each physical component are computed *a priori* based on the simulated operations, the task requirements, and the constraint sets, assuming that the required communication resources are provided by the network. Since

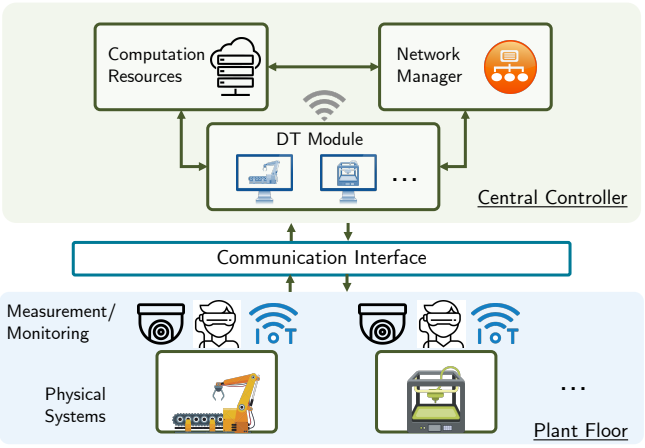


Fig. 1. Overview of the CPMS setting considered in this work.

network resources need to be shared among all physical components, there might be temporary mismatches between service requirements and network serviceability in terms of providing tailored services with exact characteristics such as latency or bandwidth during the operation time. Moreover, industrial plants might be exposed to disturbances and malfunction, resulting in potential discrepancies between the expected and real performance of each physical component. Similarly, changing operation characteristics due to interactions between the physical resources, or reconfigurations on the plant floor may result in changes in required network resources. The control action computed by the DT might be an actuation command or a reference trajectory to be executed or tracked by the physical resource or process on the plant floor.

## III. Controller-aware Resource Allocation

We start by considering the static allocation problems and move toward online and event-triggered controller-aware resource allocation schemes; in the process we outline the role of DTs in our proposed framework.

### A. Static to Dynamic Allocation

In the *static allocation* setting, allocation is done offline, based on the expected requirements and each resource gets the predetermined allocation in run-time. A generalized static allocation problem can be formulated as follows:

$$\min_{a\in\mathcal{A}} \{J(a,r)|\sum a_i \leq b\}, \tag{1}$$

where $a \in \mathbb{R}^{n_a}$ is the allocation of each asset in the network for all times; $\mathcal{A}$ denotes constraints such as non-negativity, additional lower-bounds, or limits; and $b$ is the total available resource in the network, e.g., total bandwidth, or computation capacity. $J(a,r)$ is a well-defined cost function designed to ensure optimal allocation configurations in the network, which often penalizes deviations from target allocation values. Note that the inequality constraint in (1) may be exchanged with a simplex constraint, whenever applicable.

Resource allocation according to scheme introduced in (1) relies on the expected requirements of the physical assets,

denoted by $r \in \mathbb{R}^{n_a}$, which is not robust to disturbances and run-time changes in the requirements, and does not allow for easy reconfiguration. Therefore, while static allocation may be adequate in certain settings, an interconnected Industry 4.0 system with dynamically changing plant floor environment requires higher levels of robustness and flexibility.

Dynamic resource allocation schemes utilize run-time information about the plant floor and the physical assets to allocate the network resources based on the current-time requirements. Such applications are enabled through efficient communication protocols and increased edge intelligence on devices that allow them to have better cognition. The dynamic resource allocation problem can be generally stated as

$$\min_{\boldsymbol{a}} \ \{J_N(\boldsymbol{a}, \boldsymbol{r}) | H\boldsymbol{a} \leq \boldsymbol{b}\}, \tag{2}$$

where $\boldsymbol{a} = \{a_{t_0}, \ldots, a_{t_{N-1}}\}$ is a horizon of allocation for the assets, $J_N(\boldsymbol{a}, \boldsymbol{r})$ denotes a cost function over the horizon of $N$, $\boldsymbol{r} = \{r_{t_0}, \ldots, r_{t_N}\}$ represents the forecast of requirements over the horizon, and $H\boldsymbol{a} \leq \boldsymbol{b}$ denotes input constraints and network capacity constraints over the horizon of $N$, including the set $\mathcal{A}$ in (1). Note that (2) is a multi time-step generalization of (1), and the planning horizon $N$ is application-specific. The corresponding optimal allocation, found by solving problem (2), may be implemented until the end of a horizon, which may require a re-optimization. Alternatively, the problem can be solved in receding horizon fashion, as we propose next.

### B. Online Dynamic Allocation

To take the controller requirements and performance into account, we define additional constraints and cost terms denoting the controller performance under a given network allocation. In this setting, the cost function $J_N$ may utilize an online forecast $\boldsymbol{r}$ that is updated via information collected from the plant floor in run-time. We assume here that this is a deterministic forecast, though uncertain forecasts can also be considered through robust [24] and stochastic [25] formulations. The controller-aware resource allocation problem is defined as

$$\min_{\boldsymbol{a}} \ \mathcal{I}(\boldsymbol{\gamma}) + \sum_{t=t_0}^{t_0+N-1} \ell(\xi_t, r_t, a_t) + \ell(\xi_{t_0+N}, r_{t_0+N}), \tag{3a}$$

$$\text{s.t.: } H(\boldsymbol{r})a \leq \boldsymbol{b}(\boldsymbol{r}) + \boldsymbol{\gamma}, \quad \xi_{t_0} = \hat{\xi} \tag{3b}$$

$$\xi_{t+1} = A\xi_t + Ba_t, \ t = t_0, \ldots, t_0 + N - 1, \tag{3c}$$

where $\boldsymbol{\xi} = \{\xi_{t_0}, \ldots, \xi_{t_N}\}$ is the predicted network state evolution with the dynamics given in (3c), $\hat{\xi}$ is the current network state, and $\ell$ is a well-defined loss function on the difference between the reference and the state. $\boldsymbol{\gamma}$ is a slack variable for satisfying the constraints represented by $H(\boldsymbol{r})$ and $\boldsymbol{b}(\boldsymbol{r})$, with $\mathcal{I}(\boldsymbol{\gamma})$ representing the corresponding penalty function.

The constraints represent allocation input constraints, as well as additional constraints based on the reference $\boldsymbol{r}$, such as maximum allowable deviation, or minimum allowable allocation for a given physical resource. Additionally, the vector $\xi(t)$ may represent state variables that are not only the allocated network, etc., hence the formulation of (3) allows us to consider a general class of online network allocation problems.

We interpret the minimization problem in (3) as a receding horizon control problem that accounts for the network dynamics through $A$ and $B$. For nonzero matrix $A$, we have inertia of delays in the system, meaning that the network allocation for some of the resources cannot be arbitrarily changed between consecutive time-steps. Matrix $B$ accounts for input dynamics for the allocation system, e.g., if there are proportional losses in the allocated network states (services), or if there is efficiency involved in the allocation.

An online dynamic network allocation scheme obtains $\hat{\xi}_{t_0}$, the estimate of the state at time $t_0$, solves (3), allocates the network resources as $a_{t_0} = a^*_{t_0}$, at time $t_0$, where $\boldsymbol{a}^*$ is the optimal solution, and repeats the process at the next time step. It is also possible to provide an ancillary network controller, or a baseline allocation scheme, based on either (1) or (2). In this case, the dynamic allocation scheme only accounts for online adjustments on the allocation in each time-step, with the knowledge of the baseline allocation scheme. By solving the problem (3), the dynamic allocation scheme allocates network resources according to the requirements given by the DTs in terms of desired network allocation to provide the quality-of-service required to achieve the desired quality-of-control.

### C. Event-Triggered Dynamic Allocation

Solving (3) in an online receding horizon fashion and updating the allocation of network resources may not be favorable in practical scenarios, due to computational or communication overhead, or due to complications with frequent allocation changes in the network. We propose an event-triggered dynamic allocation scheme that is more suitable in practical applications. Here, the network manager computes a re-allocation only when an event is triggered in the plant floor.

Consider the plant floor and the physical systems in Fig. 1. Let the physical systems be indexed by $i \in \{1, \ldots, n\}$. The DTs continuously monitor the output and performance of each physical component, and can identify discrepancies due to either disturbances in the physical system or a mismatch between the required and allocated network resources. We model such performance discrepancy with a *regret* function associated with each physical system, i.e.,

$$R_i(T|\tau) = \sum_{t=\tau}^{\tau+T} \left(p_i(t) - \hat{p}_i(t)\right), \tag{4}$$

where $\hat{p}_i(t)$ denotes the performance of system/component $i$ under the DT-requested resource allocation, and $p_i(t)$ is the actual measured performance of the system/component $i$ at time $t$. The metric $R_i(T|\tau)$ computes the cumulative performance discrepancy of the physical system/component $i$ starting from a time $\tau$, denoting the time of the last reallocation event, and $T$, denoting the number of time-steps after $\tau$.

The regret functions can thus track real-time performance error of each component, and if the error exceeds some thresholds, adaptation or reconfiguration may be performed. The performance is considered as "satisfactory" as long as

$$|R_i(T|\tau)| \leq \epsilon_i(\tau + T). \tag{5}$$

The thresholds $\epsilon_i(\tau + T)$ are determined *a priori* depending on the controlled performance requirements and on the characteristics of the physical system $i$. Hence they may change from one physical system to the other and between different control tasks. Additionally, note that since (4) grows with the number of time-steps, the bound $\epsilon_i(\tau + T)$ may also be set to increase as a function of time index to consider this growth, depending on the application setting. The incremental difference $(p_i(t) - \hat{p}_i(t))$ in (4) may be negative if more than required network resources are allocated to a system. The threshold in (5) penalizes such excessive allocation as the resources may be given to other systems in the network instead. The regret function (4) and the threshold (5) may be adjusted based on the specific application of interest to reflect the satisfactory allocation scenarios. The proposed scheme utilizes the regret bound (5) to trigger a dynamic allocation event, which solves the following optimization problem:

$$\min_{a} \ \mathcal{I}(\boldsymbol{\gamma}) + \sum_{t=t_0}^{t_0+N_e} \ell(\xi_t, r_t, a), \tag{6a}$$

$$\text{s.t.: } H(\boldsymbol{r})a \leq \boldsymbol{b}(\boldsymbol{r}) + \boldsymbol{\gamma}, \quad \xi_{t_0} = \hat{\xi} \tag{6b}$$

$$\xi_{t+1} = A\xi_t + Ba, \ t = t_0, \ldots, t_0 + N_e - 1, \tag{6c}$$

where the decision variable $a$ now represents the fixed network allocation for the prediction horizon of $N_e$, since the fixed allocation is only updated after an event is triggered, i.e., $a_t = a$ for $t = t_0, \ldots, t_0 + N_e$. Similar to the problem (3), forecast of resource requirements $\boldsymbol{r}$ and corresponding parameters are utilized as part of the constraints in $H(\boldsymbol{r})$ and $\boldsymbol{b}(\boldsymbol{r})$.

The horizon $N_e$ is an estimate for the time of the next event, which can be obtained through historical data. Let $\tau_k$ denote the time for the $k^{\text{th}}$ event in the past. Then the average time interval between the past $m$ events is an approximation for expected $N_e$, so that we have $N_e = \lfloor 1/m \sum_{k=1}^{m-1} \tau_{k+1} - \tau_k \rfloor$, where $\lfloor \cdot \rfloor$ denotes the floor operator.

The dynamic allocation event may be triggered by including an appropriate triggering function that monitors the regret bound (5), in the DTs. For example, a trigger may be set when the mean regret, or maximum regret of all assets on the plant floor is above a preset amount. Using the measure of regret, the proposed allocation scheme is able to make controller-aware decisions that would satisfy requirements to ensure a desired quality-of-control.

### D. The role of DTs

The dynamic allocation methods require the knowledge of $\boldsymbol{r}$, as well as, $\hat{\xi}$ in a scalable and efficient fashion in runtime. Additionally, the centralized controller requires a runtime representation of the controlled physical resources and their network requirements, which is provided by the DTs [19]. While DT information is often utilized in a monitoring capacity in the existing literature, here we proposed to utilize the DTs as a distributed network of online observers, and to exploit the DT monitoring data as inputs to the centralized controller for online network allocation.

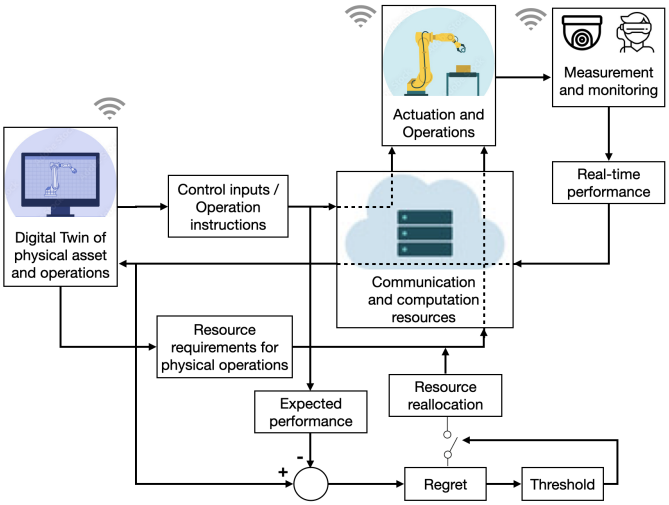


Fig. 2. Schematic diagram of a CPMS scenario (from a single system's perspective) with a network that supports the required communication and computation resources and with DTs with feedback to measure the regret and trigger resource reallocation, if necessary.

Fig. 2 illustrates the proposed dynamic allocation concept from a single physical system's perspective. The measurement and monitoring data collected from the physical systems are communicated to the central controller, through the unified communication interface (see Fig. 1). DTs of the corresponding physical resources process this data and compute the control action via the computational resource allocated by the network manager, which is then applied on the physical system. Therefore, DTs provide a distributed network of computational intelligence that use common computational resources based on the allocation determined by the network manager. The performance regrets of the processes are also continuously monitored by the DTs, and in the case of event-triggered allocation, new allocations are computed, when the regret metric is above a certain tolerable threshold.

## IV. Simulation Study

We demonstrate the proposed network controller on a simulated plant floor with interconnected resources. The plant consists of $n_r$ physical resources, all of which have their own DTs. In this study we consider a job shop with several CNCs, 3D printers, and transport agents to move products between machines (see Fig. 3). Similar settings are given in [14], [19]. The DTs monitor the process outputs of the physical resources and apply control inputs using the system dynamics and control models of the resources. The control task for a CNC might require computation of optimal cutting parameters, 3D printers require layer-to-layer monitoring and parameter updates, while a transport agent, e.g., a robot or an AGV, might compute optimal paths between source and target locations. Note that we provide the schematic setting in Fig. 3 for the conceptual case study, and the proposed allocation framework can be used in any baseline CPMS setting as outlined in Section II. The control inputs are computed using optimization-based control methods, where the objective is to

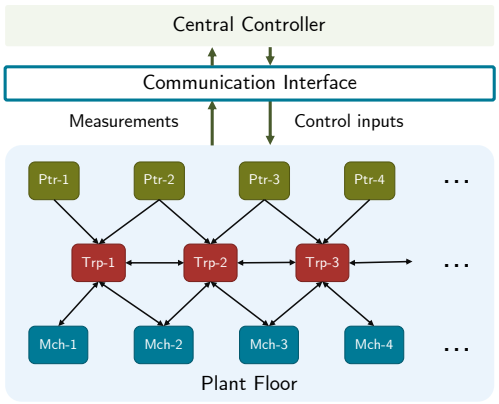


Fig. 3. CPMS setup in the case study. Ptr: Printer, Trp: Transport (Robot/AGV), Mch: CNC Machines.

minimize a control cost, given a model and a set of constraints of the system.

### A. Setting

For the case study, we implement a proximal gradient algorithm (PGA) to solve at each simulation time $t$ for each DT. The PGA is used for computing the control action for each physical system. Given an initial and feasible $x_k = x_0$, the algorithm performs the following iterates

$$x_{k+1} = \Pi_{\mathcal{X}} \left( x_k - \alpha \nabla f(x_k) \right), \tag{7}$$

where $\mathcal{X}$ is a convex input constraint set with the projection operator $\Pi$, $\alpha$ is the step size, and $\nabla f(x_k)$ is the gradient of the objective function $f$ at point $x_k$. Here, the input constraints may represent actuator constraints, and the objective function may be a function on the model-based tracking error. Under suitable conditions on the step-size, convexity of $f$, and the optimal solution, the PGA converges to the unique minimizer $x^*$ [26]. Then, the control law to actuate the physical asset at time $\tau$ can be defined as $u_\tau = \sigma(x_f)$, where $\sigma$ defines a static control policy on the final iterate of the PGA $x_f$. If the PGA is performed until convergence, we have $x_f = x^*$. Suppose the function $f$ is $L$-smooth and convex. Then, choosing a constant step-size of $\alpha \in (0, 1/L]$ gives the suboptimality bound

$$F(x_k) - F(x^*) \leq \frac{||x_0 - x^*||^2}{2\alpha k}, \tag{8}$$

where $F(x) = f(x) + \iota_{\mathcal{X}}(x)$, with $\iota_{\mathcal{X}}$ being the indicator function for the convex set $\mathcal{X}$ ([26], Theorem 4.9). In practice, the algorithm is run until a $\delta$ convergence criteria is reached or for a predefined fixed number of iterations due to computation time or capacity limits. Given the diameter of $\mathcal{X}$ and $\alpha$, it is possible to bound the right side of (8) to a ball of $\delta$ as a function of $k$. Note that while this may not be a tight bound, it nevertheless provides a computable formal upper bound on how many iterations are needed to get the desired convergence ball. Let $k'$ denote the required number of iterations to achieve the $\delta$ ball based on (8). Note that since different resources may have different controllers and requirements, $f$, $\mathcal{X}$, and therefore, $k'$ might differ between resources and DTs.

In our setting, the computations of each DT are done over a networked resource which is available to the extent allowed by a network manager. To run the PGA algorithm until convergence (e.g., for $k'$ steps), each DT requires a certain computational resource. The DTs share the values of $k'$ and a minimum acceptable lower-bound $k_\ell$ is shared with the network manager in each time step.

To demonstrate the performance of the proposed network allocation schemes, we compare four scenarios, increasing in complexity, and provide results in the following section.

($C1$) Network resources allocated equally
($C2$) Network resources allocated according to policy (1) with expected network requirements
($C3$) Event-triggered allocation scheme
($C4$) Network resources allocated according to the dynamic allocation, solving problem (3) in receding horizon.

The simulation is implemented in Matlab. All DTs provide the requirement prediction vectors to the network manager and the network manager provides the network allocation through a publish-subscribe channel in the simulation. There are 20 physical resources in the network and the simulation takes 100 time steps. We consider quadratic cost functions $f$ with uniform input dimensions for all resources for simplicity. The network requirement for all agents is stationary for the first 10 time steps, and updated with bounded random integers in future times. The maximum allowable deviation from the requested network allocation is set at 10 for all resources, e.g., $k' - k_\ell = 10$. We implement the event-triggering mechanism as a preset threshold for maximum regret across all resources. For simplicity, let $A = 0$ and $B = I$ for the simulation study, so that the state $\xi$ in the optimization problem represents the current allocation in the network and we set $N = 1$ in (3).

### B. Results

Fig. 4 illustrates the maximum network resource allocation residual at a given simulation tick, i.e., $||r_t - a_t||_\infty$ for the four simulation scenarios. Equal allocation in ($C1$) performs the worst as it does not account for the requirements provided by the DTs of the physical systems and the online changes. The scenarios ($C2$), ($C3$), and ($C4$) have identical allocation residual at the beginning, representing the static allocation at the beginning. However, after time step 10, we see that the proposed online dynamic allocation-based controller outperforms all other methods.

Additionally, we see that the maximum resource residual is limited below 10 for the online allocation schemes ($C3$) and ($C4$), shown with the colored background in Fig. 4. Using the maximum allowable resource residual reported by the DTs, the proposed controller can achieve noticeably better performance by using the run-time data to actively adjust the network resource allocation. The online allocation ($C4$) reallocates the resources in each time step, which may be undesirable in practical scenarios due to the high frequency of computing the allocation pattern that may require considerable computation power. The event-triggered scheme in ($C3$) achieves comparable quality-of-service in the sense of keeping the maximum

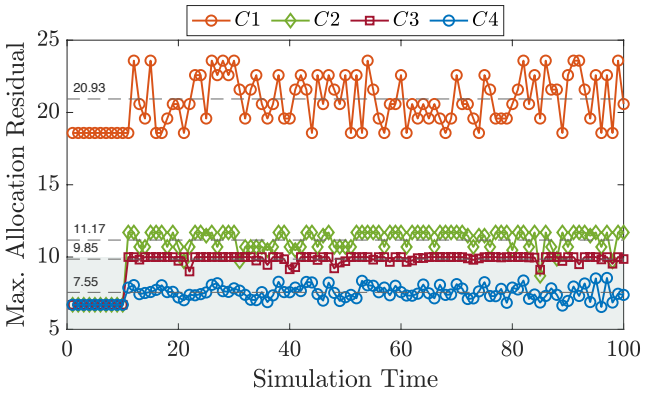


Fig. 4. Comparison of all four network resource allocation scenarios compared in the case study. Mean values after time step 10 are shown with dashed lines and values on the left. The maximum allowable allocation residual is shown with colored background.

residual below the required level by reallocating only 15 times throughout the simulation.

## V. Conclusion and Future Work

This paper proposes controller-aware network allocation schemes for efficient network resource management in CPMS settings. The proposed framework demonstrates that the flexibility and robustness of novel networking technologies can be exploited together with the application of DT to efficiently respond to critical industrial operation requirements for ICPS and CPMS. We provide practical and deployable methods to continuously reallocate resources and controller performance-based metrics to perform reallocation based on events, and provide a simulation study to illustrate the efficiency of proposed dynamic resource allocation methods. Future work will consider the effect of different triggering functions, the effect of additional network dynamics and inertia, and further analysis of the effect of control algorithms on resource allocation architectures.

## References


[1] H. Xu, W. Yu, D. Griffith, and N. Golmie, "A survey on industrial internet of things: A cyber-physical systems perspective," *IEEE Access*, vol. 6, pp. 78238–78259, 2018.

[2] F. Allgöwer, J. Borges de Sousa, J. Kapinski, P. Mosterman, J. Oehlerking, P. Panciatici, M. Prandini, A. Rajhans, P. Tabuada, and P. Wenzelburger, "Position paper on the challenges posed by modern applications to cyber-physical systems theory," *Nonlinear Analysis: Hybrid Systems*, vol. 34, pp. 147–165, 2019.

[3] S. A. Seshia, S. Hu, W. Li, and Q. Zhu, "Design automation of cyber-physical systems: Challenges, advances, and opportunities," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 36, no. 9, pp. 1421–1434, 2017.

[4] L. Monostori, B. Kádár, T. Bauernhansl, S. Kondoh, S. Kumara, G. Reinhart, O. Sauer, G. Schuh, W. Sihn, and K. Ueda, "Cyber-physical systems in manufacturing," *CIRP Annals*, vol. 65, no. 2, pp. 621–641, 2016.

[5] G. Walsh and H. Ye, "Scheduling of networked control systems," *IEEE Control Systems Magazine*, vol. 21, no. 1, pp. 57–65, 2001.

[6] J. Baillieul and P. J. Antsaklis, "Control and communication challenges in networked real-time systems," *Proceedings of the IEEE*, vol. 95, no. 1, pp. 9–28, 2007.

[7] Z. Zhou, Y. Guo, Y. He, X. Zhao, and W. M. Bazzi, "Access control and resource allocation for M2M communications in industrial automation," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 5, pp. 3093–3103, 2019.

[8] T. Taleb, I. Afolabi, and M. Bagaa, "Orchestrating 5G network slices to support industrial internet and to shape next-generation smart factories," *IEEE Network*, vol. 33, no. 4, pp. 146–154, 2019.

[9] C. Papagianni, J. Mangues-Bafalluy, P. Bermudez, S. Barmpounakis, D. De Vleeschauwer, J. Brenes, E. Zeydan, C. Casetti, C. Guimarães, P. Murillo, A. Garcia-Saavedra, D. Corujo, and T. Pepe, "5growth: AI-driven 5G for automation in vertical industries," in *European Conference on Networks and Communications*, pp. 17–22, 2020.

[10] J. Ordonez-Lucena, J. F. Chavarria, L. M. Contreras, and A. Pastor, "The use of 5G non-public networks to support Industry 4.0 scenarios," in *2019 IEEE Conference on Standards for Communications and Networking (CSCN)*, pp. 1–7, 2019.

[11] J. García-Morales, M. C. Lucas-Estañ, and J. Gozalvez, "Latency-sensitive 5G RAN slicing for Industry 4.0," *IEEE Access*, vol. 7, pp. 143139–143159, 2019.

[12] L. Bonati, M. Polese, S. D'Oro, S. Basagni, and T. Melodia, "Open, programmable, and virtualized 5G networks: State-of-the-art and the road ahead," *Computer Networks*, vol. 182, 2020.

[13] L. Yin, J. Luo, and H. Luo, "Tasks scheduling and resource allocation in fog computing based on containers for smart manufacturing," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 10, pp. 4712–4721, 2018.

[14] I. Kovalenko, J. Moyne, M. Bi, E. C. Balta, W. Ma, Y. Qamsane, X. Zhu, Z. M. Mao, D. M. Tilbury, and K. Barton, "Towards an automated learning control architecture for cyber-physical manufacturing systems," *IEEE Access*, 2022.

[15] F. Lopez, Y. Shao, Z. M. Mao, J. Moyne, K. Barton, and D. Tilbury, "A software-defined framework for the integrated management of smart manufacturing systems," *Manufacturing Letters*, vol. 15, pp. 18–21, 2018.

[16] X. Li, A. Garcia-Saavedra, X. Costa-Perez, C. J. Bernardos, C. Guimarães, K. Antevski, J. Mangues-Bafalluy, J. Baranda, E. Zeydan, D. Corujo, P. Iovanna, G. Landi, J. Alonso, P. Paixão, H. Martins, M. Lorenzo, J. Ordonez-Lucena, and D. R. López, "5growth: An end-to-end service platform for automated deployment and management of vertical services over 5G networks," *IEEE Communications Magazine*, vol. 59, no. 3, pp. 84–90, 2021.

[17] M. C. Lucas-Estañ, T. P. Raptis, M. Sepulcre, A. Passarella, C. Regueiro, and O. Lazaro, "A software defined hierarchical communication and data management architecture for industry 4.0," in *2018 14th Annual Conference on Wireless On-demand Network Systems and Services (WONS)*, pp. 37–44, 2018.

[18] Y. Qamsane, C.-Y. Chen, E. C. Balta, B.-C. Kao, S. Mohan, J. Moyne, D. Tilbury, and K. Barton, "A unified digital twin framework for real-time monitoring and evaluation of smart manufacturing systems," in *IEEE 15th international conference on automation science and engineering (CASE)*, pp. 1394–1401, IEEE, 2019.

[19] J. Moyne, Y. Qamsane, E. C. Balta, I. Kovalenko, J. Faris, K. Barton, and D. M. Tilbury, "A requirements driven digital twin framework: Specification and opportunities," *IEEE Access*, vol. 8, pp. 107781–107801, 2020.

[20] J. Vachálek, L. Bartalský, O. Rovný, D. Šišmišová, M. Morháč, and M. Lokšík, "The digital twin of an industrial production line within the Industry 4.0 concept," in *2017 21st International Conference on Process Control (PC)*, pp. 258–262, 2017.

[21] X. Wang and M. D. Lemmon, "Event-triggering in distributed networked control systems," *IEEE Transactions on Automatic Control*, vol. 56, no. 3, pp. 586–601, 2011.

[22] M. H. Mamduhi, A. Molin, and S. Hirche, "Event-based scheduling of multi-loop stochastic systems over shared communication channels," in *21st International Symposium on Mathematical Theory of Networks and Systems (MTNS)*, pp. 266 – 273, 2014.

[23] M. H. Mamduhi, D. Maity, J. S. Baras, and K. H. Johansson, "A cross-layer optimal co-design of control and networking in time-sensitive cyber-physical systems," *IEEE Control Systems Letters*, vol. 5, no. 3, pp. 917–922, 2021.

[24] W. Langson, I. Chryssochoos, S. Raković, and D. Q. Mayne, "Robust model predictive control using tubes," *Automatica*, vol. 40, no. 1, pp. 125–133, 2004.

[25] A. Mesbah, "Stochastic model predictive control: An overview and perspectives for future research," *IEEE Control Systems Magazine*, vol. 36, no. 6, pp. 30–44, 2016.

[26] A. Chambolle and T. Pock, "An introduction to continuous optimization for imaging," *Acta Numerica*, vol. 25, pp. 161–319, 2016.