Augmented Collective Digital Twins for Self-Organising Cyber-Physical Systems

Roberto Casadei[†], Andrea Placuzzi[†], Mirko Viroli[†], Danny Weyns[§]

[†] Department of Computer Science and Engineering
Alma Mater Studiorum–Università di Bologna, Cesena, Italy
{roby.casadei, andrea.placuzzi, mirko.viroli}@unibo.it

[§] Department of Computer Science
Katholieke Universiteit Leuven, Belgium, Linnaeus University, Sweden
danny.weyns@kuleuven.be

***Abstract*—*Context.* Self-organising and collective computing approaches are increasingly applied to large-scale cyber-physical systems (CPS) enabling them to adapt and cooperate in dynamic environments. Also, in CPS engineering, digital twins are often leveraged to provide synchronised logical counterparts of physical entities, whereas in sensor networks the different-but-related concept of virtual device is used to abstract groups of sensors.**

***Vision.* We envision the design concept of "augmented collective digital twin" that captures digital twins at a collective level extended with purely virtual devices. We argue that this concept can foster the engineering of self-organising CPS by providing a holistic, declarative, and integrated system view.**

***Method.* From a review and proposed taxonomy of logical devices comprehending both digital twins and virtual devices, we reinterpret a meta-model for self-organising CPSs and discuss how it can support augmented collective digital twins. We illustrate the approach in a crowd-aware navigation scenario, where virtual devices are opportunistically integrated into the system to enhance spatial coverage, improving navigation capabilities.**

***Conclusion.* By integrating physical and virtual devices, the novel notion of augmented collective digital twin paves the way to self-improving system functionality and intelligent use of resources in self-organising CPSs.**

***Index Terms*—cyber-physical systems, self-organisation, digital twins, virtual devices, collective systems, aggregate computing**

I. Introduction

Recent techno-scientific developments are promoting a vision of smart large-scale Cyber-Physical Systems (CPSs) where computational and physical processes integrate to support novel types of applications and services [1], [2]. Examples of such future-generation systems include Wireless Sensor and Actuator Networks (WSANs), swarms of robots, groups of people augmented with smart devices, and smart computing ecosystems. Often, such systems are equipped with autonomic or self-* capabilities to provide adaptive system functions and/or promote quality aspects like resilience and efficiency.

In typical CPS designs, physical devices have corresponding software counterparts, known as *digital twins* [3], to represent and exploit them in computational settings. There is a close connection between these two parts, essential for effective cyber-physical integration. In settings like WSANs and Internet-of-Things (IoT), the related but different notion of a *virtual device* (that has no twinning notion with any physical device) has often been proposed to achieve abstraction or improved Quality of Service (QoS) [4]–[7], e.g. by abstracting a set of underlying physical sensors.

This paper describes a vision and preliminary work whereby logical representations of system entities (e.g., digital twins and virtual devices) are key: (i) for separation of concerns in system design and implementation, and (ii) for improving performance and reducing costs, e.g., through smart deployment, separating the application logic from the deployment context. Our position is that programming approaches to self-organisation can especially benefit from such logical representations, e.g., in terms of declarativity and modularity.

We develop the vision through the following contributions matching the paper structure:

- A review and novel proposed taxonomy of logical devices (fitting both digital twins and virtual devices, and taking into account the "collective" dimension) based on the relationships of *identity* and *execution* with physical devices (Section II);
- A re-interpretation and conceptual extension of the meta-model of self-organising CPSs presented in [8] in light of the novel proposed concept of *augmented collective digital twin*, together with a preliminary investigation showing how virtual nodes can be opportunistically integrated [9] to improve performance of a self-organising crowd-aware navigation service (Section III).

Finally, Section IV wraps up and gives an outlook to the future.

II. Digital Twins and Virtual Devices for CPS Engineering

In this section, we review the literature on digital twins and virtual devices (both referred to as *logical devices* in this paper) and propose an integrated taxonomy based on two concepts: *identity* (of logical and physical devices, possibly shared) and *execution* (of logical devices by physical devices). Moreover, we introduce the novel notion of a *collective digital twin*, namely the digital twin of an entire collective of physical

Ref.	Goals	Techniques	Applications	Network architecture	**Kind of virtual node** (cf. taxonomy in Table II)
[7]	Predictability	Collaborative emulation of virtual nodes	WSN applications	Ad-hoc	Virtual aggregate device
[10]	Improved QoS	TDMA prioritisation	WSN applications	Clustered	Digital copy
[5]	Abstraction and efficiency	Optimal formation and composition of resource-constrained sensors	WSN applications	Cloud-based (Hierarchical)	Virtual aggregate device
[6]	Improved QoS	QoS-aware service composition	Cloud-based IoT applications	Cloud-based (Hierarchical)	Virtual aggregate device

TABLE I: Examples of use of notions of "virtual nodes" in literature.

devices, which we leverage in Section III to describe a model of self-organising CPS.

A. Digital Twins and Virtual Nodes in Literature

A *digital twin* can be defined as a connected, virtual model of a physical entity [3]. Such a (bi-directional) connection is realised by the so-called *digital thread*, which implements the data flows needed to keep the digital and physical counterparts synchronised. The digital twin concept is related but different from other associated engineering concepts; for instance, in [11], the authors distinguish between a digital model (where a physical object and a digital object are synchronised via a manual data flow), a digital shadow (where a change in the physical object is automatically reflected to the digital object but not vice versa), and a digital twin (with automatic, bi-directional data flow). In [12], a taxonomy of digital twins is given, classifying the concept by, e.g., data link (one-directional or bi-directional), purpose (processing, transfer, repository), physical binding (unbound, bound), accuracy (partial, identical), synchronisation (with or without), and creation time (physical part first, digital part first, simultaneous creation).

The notion of a digital twin implies the presence of a corresponding physical twin. We define a *virtual device* as a logical device that does not correspond to any physical device. This notion leverages on various works as summarised in Table I. In [4], Bose et al. define a virtual sensor as an *"entity consisting of a group of sensors along with associated knowledge which enables it to provide services which are beyond the capabilities of any of its individual member sensors."* They also classify virtual sensors into three classes: *singleton* virtual sensors, which represent a corresponding physical sensor; *basic* virtual sensors, which consist of a homogeneous group of singleton sensors; and *derived* virtual sensors, which consist of a heterogeneous group of virtual sensors. In this view, a virtual node abstracts over a set of underlying physical nodes. This introduces the problem of finding efficient strategies for creating such abstractions. For instance, in [5], new algorithms are introduced for efficient virtualisation of *groups* of constrained physical sensors as virtual sensors in the context of sensor-cloud infrastructures. Virtual nodes as group abstractions may serve specific purposes. For instance, in [6], a notion of virtual sensor is used to achieve load-balancing and data-sharing and promote QoS-aware service composition in cloud-based IoT applications. Conversely, in [7], a virtual node layer and programming abstraction is proposed to provide *reliability* in low-cost ad-hoc networks. In this approach, the programmer deals with "predictable" *virtual nodes* and "unpredictable" *client nodes*, namely, the underlying physical devices that emulate the former. The emulation architecture uses a regional management pattern [13] where the network is divided into leader-regulated regions; the local node support deals with leader management, regional membership, and consistency (e.g., through consensus and synchronisation). In other works, virtual nodes have uses different from group abstraction. In [10], a concept of virtual node is used to improve the QoS in clustered WSNs operating through a Time-Division Multiple Access (TDMA) protocol by giving higher priority to "critical nodes".

B. Logical Devices: A Taxonomy

We consider a *cyber-physical system (CPS)*. In a CPS, three main aspects are addressed: sensing, computation, and actuation (communication is considered as part of sensing and actuation). Of these three aspects, two of them – sensing and actuation – are *situated*; their *location* (i.e., *where* they are carried out) in the physical world is important for the application. This property is achieved by placing physical sensors and actuators in the environment, either in a standalone fashion or as a part of a larger, possibly mobile device (e.g., a robot or a smartphone) from which they inherit a situation. On the other hand, computation is often *disembodied*, i.e., it may in principle happen *anywhere*. However, there are typically constraints (cf. latency, bandwidth, cost, energy) affecting where computation should reside, since inputs and outputs must be communicated among physical nodes of the network. Modulo these constraints, this disembodied nature of many computations enables trade-offs and flexibility w.r.t. deployment choices and application execution strategies.

Additionally, physical nodes may be distinguished by their role in system design, in:

- *Application-level* physical nodes (APN): those monitored and/or controlled by the application;
- *Infrastructure-level* physical nodes (IPN): those supporting application execution and connectivity.

For instance, in a multi-robot CPS, the robots are the APNs, and other computers providing connectivity or support for computation offloading are the IPNs. In other words, APNs include physical sensors, physical actuators, or larger physical devices aggregating several physical sensors and actuators (e.g., robots or smartphones).

In an engineering approach to CPSs, we can distinguish between *logical* devices and components (existing in mod-

Cyber/physical identity correspondence	**Logical device**	**Physical device**	**Description**
1-to-0	Virtual device	–	A logical device corresponds to no physical device.
1-to-1	Digital twin	Physical twin	A logical device corresponds to exactly one physical device.
1-to-N, $N > 1$	Virtual aggregate device	Physical component	A logical device is a virtual abstraction for a group of physical devices.
0-to-1	–	Infrastructural device	A physical device has no corresponding virtual device (i.e., it only provides execution support).
N-to-1, $N > 1$	Digital view (heterogeneous), digital copy (homogeneous)	Physical host	A physical device has multiple corresponding virtual devices (with different identities).
N-to-M, $N > 1, M > 1$ (mapping unspecified)	Collective digital twin	Collective physical twin	There is a (unknown) mapping between a group of digital identities and a group of physical devices.

TABLE II: Different correspondences between cyber nodes and physical nodes lead to different notions (denoted by specific terms). This manuscript explicitly focusses on the highlighted notions.

Cyber/physical execution relationship	**Logical device** (through their **software components**)	**Physical device**	**Description**
1-to-1	Controller/Virtualised node	Controlled node/Virtualiser	A logical device is executed by exactly one physical device.
1-to-N, $N > 1$	Logical cluster	Physical component	A group of physical devices execute a single logical device.
N-to-1, $N > 1$	Offloaded logical component	Server / Surrogate	A group of logical devices is executed by a single physical device.
N-to-M, $N > 1, M > 1$ (mapping unspecified)	Logical system	Infrastructure	A group of logical devices runs (in an unspecified way) on a group of physical devices.

TABLE III: Different execution relationships between cyber nodes and physical nodes lead to different notions (denoted by specific terms).

els), *software* components (representing logical components as computational and deployable units), and *physical* devices (physical entities in the world—e.g., computers, robots, smartphones). Note that the relationship between logical and software components resembles the relationship between a digital twin and its digital thread. In [8], a logical device is (assumed to be) associated to *one* physical device (APN), but its execution may be supported by multiple physical devices—one or more IPNs *together with or without* the APN. In this work, we extend such a partitioning schema considering that a logical device may be associated to zero, one, or more physical devices. Refer to Table II for possible ways to logically associate logical with physical devices. Note that such an association relationship (cf. Table II) is different from a deployment/execution relationship (cf. Table III): the associated logical and physical devices conceptually share an *identity*. Indeed, we propose (as shown in Tables II and III) to use the notions of *identity correspondence* and *execution relationship* to characterise the relationship (mediated by software components) between logical devices and physical devices.

III. Digital Twins and Virtual Nodes in Self-Organising Cyber-Physical Systems

After presenting a motivating example (Section III-A), in this section we reinterpret the meta-model of self-organising CPSs introduced in [8] according to the conceptual framework of the previous section (Section III-B), then show how it can be used to support purely virtual devices (Section III-C3).

A. Motivating Example: Crowd-Aware Navigation

We consider a crowd-aware navigation application as a motivating example. The supposed system consists of hundreds of nodes deployed in a city: these may include smart city devices, infrastructural elements (e.g., fog nodes), smartphones or wearable devices hold by citizens. The system runs an "aggregate program" [14] that collectively computes an approximated estimate of the level of crowding in granular areas, and handles "navigation requests"—extending over crowd tracking and dispersal as in [15]. Indeed, during the system operation, people may want to reach Points of Interest (PoIs), using the system to calculate the path. For safety or traffic management reasons, the system may not suggest paths passing through overcrowded areas. The basic idea of the self-organising logic is to use *gradients* [16] and neighbour-based gossip to create self-healing "channels", where the dynamic set of devices belonging to the channel provide the spatial locations to be followed to move from the source area to the destination area. Such an approach assumes that stateful devices repeatedly compute the self-organising logic and interact in asynchronous rounds with neighbours in a reasonable range (e.g., a 100-metre WiFi range): by repeated computation and communication, locally processed information dependent on neighbours affects neighbours in turn, and larger portions of the system consequently. Note that this programming model can be applied to the logical system made up of the digital twins corresponding to the situated physical devices. In [8], it is shown that different architectural deployments are possible when realising the "digital thread" (synchronising digital twins with data from physical sensors, redirecting actuations to physical twins, and handling physical communication for logical interaction), for different performance and cost profiles.

Such an approach to crowd-aware navigation is scalable due to logical decentralisation. Additionally, it can be implemented in mobile ad-hoc networks, where cloud-based navigation services like Google Maps are not (temporarily or permanently) available. The only requirement is the ability to estimate the

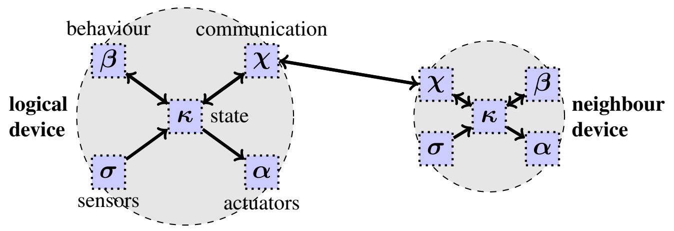

Fig. 1: A logical device, pulverised into logical components, connected to another neighbour logical device. Arrows denote directions of data flow.

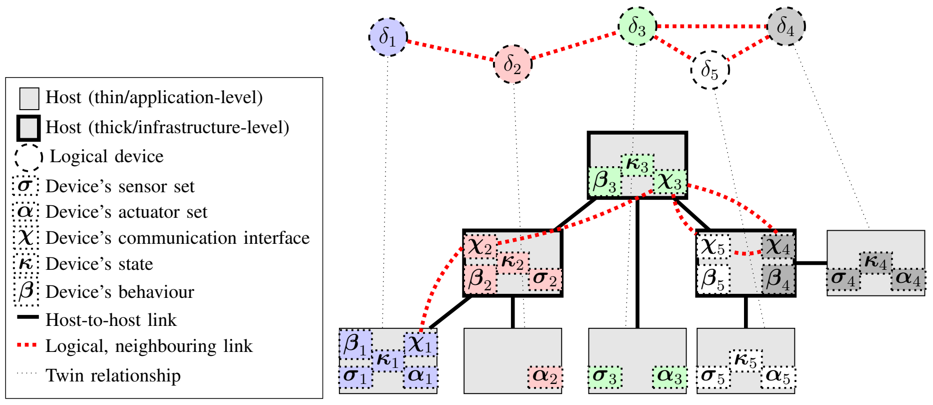
Host (thin/application-level)
Host (thick/infrastructure-level)
Logical device
Device's sensor set
Device's actuator set
Device's communication interface
Device's state
Device's behaviour
Host-to-host link
Logical, neighbouring link
Twin relationship
δ_1
δ_2
δ_3
δ_4
δ_5
β_1 χ_1 κ_1 σ_1 α_1
χ_2 κ_2 β_2 σ_2 α_2
κ_3 β_3 χ_3 σ_3 α_3
χ_4 β_4 κ_4 σ_4 α_4
χ_5 β_5 κ_5 σ_5 α_5

Fig. 2: Example instantiation of the CPS meta-model. The digital collective twin (top) is mapped to its physical collective twin (bottom) augmented with infrastructure-level nodes. Notation: dashed circles are logical devices; different colours or subscripts denote different identities; red dotted links are logical connections; dotted squares are logical deployable components; solid-line elements are platform or physical components/connections. Intra-device connections (cf. Figure 1) are not shown but they can cross host boundaries by following direct host-level links.

distance to neighbours, and to describe where the PoI is located. However, it has limitations: on one hand, considering overcrowded areas as inaccessible areas may arise reachability issues (the channel could not be formed); on the other hand, spatial areas that are not covered by any device could not equally be exploited for path computation. To solve the latter issue, we would like to spawn virtual nodes to provide a basic coverage of space for the sake of application functionality and path efficiency, at the expense of some virtualisation overhead. See Figure 3 for a graphical example of the idea.

B. A Pulverisation Meta-Model of Self-Organising CPSs with Collective Digital Twins

In [8], a pulverisation meta-model of self-organising CPSs is proposed, consisting in:

1) *Cyber* subsystem: a logical network consisting of *logical devices* connected as per a *neighbouring relationship*; such logical devices are split (i.e., "pulverised") into a number of *logical components* (sensors σ, actuators α, computation β, communication χ, and state components κ) that can be independently deployed (see Figure 1).
2) *Platform* subsystem: an abstraction over the physical system consisting of an actual network of *hosts* (physical or virtualised), each with a *network identity* (e.g., an IP address or URI) and *network links* for communicating with other hosts.
3) *Deployment*: a cyber-physical mapping whereby and logical device components are *deployed* to specific hosts (cf. execution relationship). See Figure 2.

The underlying assumption is that the cyber subsystem represents the *collective digital twin* of the *collective physical twin* given by the subset of the application-level hosts of the platform subsystem. The idea is that the programmer focusses on the cyber subsystem and the integrator and middleware provider focus on the platform subsystem and the mapping.

The meta-model enforces an architectural organisation principle that *fosters* deployment-independent self-organising logic, but self-organisation is not implied by the model: it must be injected through proper behaviour and interaction logic for the logical components. In [8], a behavioural meta-model is provided in terms of small-step operational semantics. Hereby, the cyber subsystem evolves through pre-defined, "continuous" computations and interactions of the logical components. Such interactions are congruently carried out by actual interactions within one host or between two hosts using network communication. More specifically, the logical model promotes an organisational principle for self-organising behaviour as follows:

- Sensors σ integrate their sensor readings to the device state κ, making them available for decision-making and coordination;
- Actuators α retrieve state information to control their actuation to the device itself or the environment;
- The computation behaviour element β provides a "reasoning step" based on the current context stored in state κ, and accordingly updates the latter in order to coordinate with other devices or act on context;
- The communication element χ integrates information from neighbours to the device state κ, and retrieves local information from the latter to provide neighbours with local data for coordination purposes.

The model abstracts from the concrete shape of the logical components. In [8], the model has been instantiated into the aggregate computing framework [14], [15], whereby the behaviour β is the same and fixed for every device, the state component κ keeps a tree-like structure updated in the computation steps, and communication components χ are unspecified—they must just reify some domain-specific logical neighbouring relationship. The model also abstracts from the actual *scheduling* of the described logical (and corresponding physical) interactions. This leaves execution details to implementations, enabling them to regulate reactivity to inputs, actuation rate, and self-organising process pace according to various (possibly dynamic) cost and performance goals.

Finally, as mentioned previously, the other element affecting cost and performance of a system is the actual deployment of the software components, corresponding to the logical entities, over the available edge-fog-cloud infrastructure. In [8], it is shown that different deployment configurations result in different cost-performance profiles, and simulations can be carried out to make informed decisions before actual deployment. Next, we propose to extend over [8] by considering the integration of purely virtual devices in self-organising CPSs.

C. Augmented Collective Digital Twin

1) Digital Twins and Virtual Devices: The discussed meta-model, as covered in [8], only captures the execution relationship but not the twinning relationship. Such a twinning relationship is generally enforced through deployment by mapping sensors and actuators of logical devices (digital twins) to those of a set of designated application-level hosts (physical twins). The synchronisation of state works then through the aforementioned operational semantics. Therefore, in order to synthesise virtual devices, their sensors and actuators must be virtualised as there is no direct correspondence with physical sensors and actuators. Typically, virtual devices are positioned at arbitrary positions in space, and their position and distance-to-neighbour sensors are accordingly populated. Moreover, the middleware must synchronise, collect, reason about, and propagate information about the virtual device—such that its neighbours can actually perceive and interact with it. This issue is relatively easy in centralised deployments, but becomes more challenging in decentralised settings.

2) Augmentation: The platform subsystem is generally "augmented" with an additional (not necessarily disjunct) set of infrastructure-level nodes to support execution aspects like connectivity and computation offloading. On the other hand, the cyber subsystem can also be "augmented" through purely virtual devices which are not associated to any host—what we call an *augmented collective digital twin*. Such virtual devices correspond to an *augmented reality* for the physical twins.

3) Self-adaptive virtual node management: Smart use of virtual devices can help improve the application performance and enable new functionality *without changing the self-organising logic of the system*. For instance, in the crowd-aware navigation example, virtual devices may be added to cover additional areas of the system and hence enable the computation of additional paths. In the same application, the creation of several virtual devices may be used as a way to *prevent* navigation paths to pass through in a limited spatial region—e.g., as a way to substantiate a prediction of a future crowding due to planned events. Another use case could be the injection of virtual devices to support in-field testing (cf. chaos engineering), mixed-reality simulations, or learning.

However, execution of virtual devices may introduce a relevant overhead, especially in decentralised implementations. So, virtual devices might be dynamically created or removed depending on *opportunistic computing* considerations. This makes the case for a *self-adaptive managing system* [17] for virtual devices. Interestingly, such a managing system may be implemented through a lower application layer programmed using the same discussed approach, where the collective system provides a distributed virtualisation platform.

D. Preliminary Evaluation of the Motivating Example

We have run simulations to validate the idea through the motivating example discussed in Section III-A. We leverage the Alchemist simulator [18] and the ScaFi aggregate programming language [19]. The experiments are open-sourced and available at a public repository[1], which also contains further details omitted here for space reasons. The results in Figure 4 show that the system, augmented with virtual devices and implementing the idea represented in Figure 3, can (i) provide a shorter path and (ii) solve reachability issues.

IV. Conclusion and Outlook

In this paper, we proposed a vision that integrates different notions of logical components (cf. digital twins and virtual devices) within a framework of self-organising CPS engineering. Also, we introduced a notion of *collective digital twin* as the digital twin of a collective of physical devices. Such a notion is then evolved into a *logical collective* by augmenting (cf. augmented reality) the collective physical twin with purely virtual devices (possibly by a self-improving integration process [9]). Moreover, we showed that such notions are not merely an exercise of abstraction, but can provide actual benefits. In previous work, we showed that it is possible to come up with multiple deployment architectures of a CPS without affecting its self-organising logic [8]. In this paper, we discussed preliminary results showing that integration of virtual devices can improve the application performance. However, we think that the discussed concepts can promote discussion and research along various perspectives:

- *Digital twins for collective and self-organising CPSs.* Previous work in [8] shows that defining an identity correspondence between logical and physical entities, both at the individual (a logical device and a physical device—twins) and collective level (a logical collective and a physical collective—collective twins), is instrumental for separating three concerns: (i) self-organisation programming; (ii) a partitionable architecture for the "digital thread" (for synchronising the digital and physical twins); and (iii) the deployment of architectural components to physical devices and supporting infrastructure.
- *Adaptive deployment of self-organising CPSs.* The latter result paves the path to a self-adaptive deployment

[1]https://github.com/aPlacuzzi/Experiment-acsos-2021

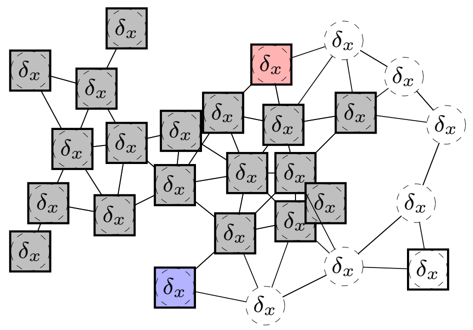

Fig. 3: Virtual nodes and full nodes in a crowd-aware navigation scenario. Nodes sensing an overcrowded situation are in grey. The blue device aims to reach the red device through a path computed in a self-organising way by the system. Unfortunately, the central part of the space is too crowded, and should be avoided for safety reasons. By integrating virtual devices into the system, however, it is possible to provide a minimal coverage of the space, allowing for more paths to be computed. The system could also (in principle) continuously look at strategies to self-improve such an integration for non-functional benefits. (Notation: square boxes denote physical devices; circles denote logical devices; circles outside any box denote purely virtual devices. The two-dimensional space where the nodes are located provides an approximated model of physical space.)

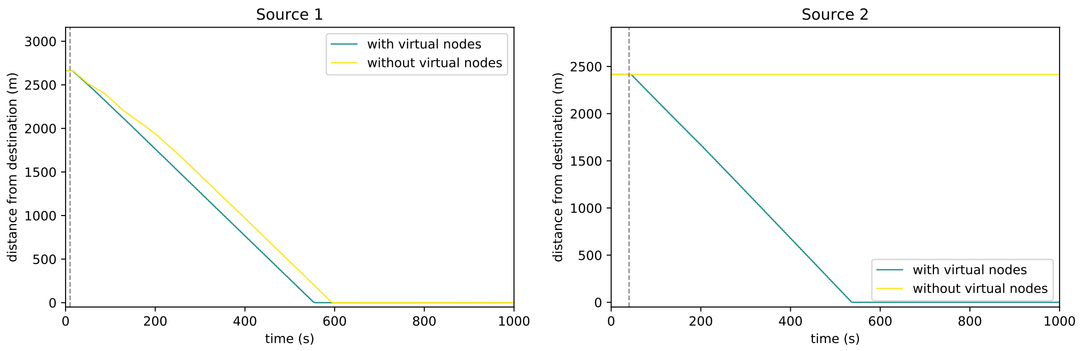

Fig. 4: Each plot represents the processing of a separate navigation request, showing the distance from the source to the destination over time, for two configurations: (i) with and (ii) without augmentation of the physical collective twin. Results are averaged over multiple runs, differing by the scheduling of rounds.

system whereby the software components realising the digital thread are strategically and tactically (opportunistically) re-located to enable efficient execution of a self-organising CPS. It is interesting to explore how different application partitioning schemas, constraints, and policies, may affect the design of the self-managing system.

- *Strategies for self-improving integration of physical and virtual devices.* The insight of the paper is that self-organising systems may be affected by *local* augmentations or diminutions to virtuality and reality. So, it will be interesting to explore general strategies and specific algorithms enabling virtual devices to be integrated within the system in a self-reconfiguring or self-improving manner.

References

[1] T. Bures *et al.*, "Software engineering for smart cyber-physical systems: Challenges and promising solutions," *SIGSOFT Softw. Eng. Notes*, vol. 42, no. 2, p. 19–24, 2017.

[2] D. Weyns *et al.*, "A research agenda for smarter cyber physical system," *Journal of Integrated Design & Process Science*, 2021. [Online]. Available: https://people.cs.kuleuven.be/danny.weyns/papers/2021JIDPS.pdf

[3] A. Rasheed, O. San, and T. Kvamsdal, "Digital twin: Values, challenges and enablers from a modeling perspective," *IEEE Access*, vol. 8, pp. 21 980–22 012, 2020. [Online]. Available: https://doi.org/10.1109/ACCESS.2020.2970143

[4] R. Bose, A. Helal, V. Sivakumar, and S. Lim, "Virtual sensors for service oriented intelligent environments," in *Proceedings of the 3rd IASTED International Conference: Advances in Computer Science and Technology*, ser. ACST'07. USA: ACTA Press, 2007, p. 165–170.

[5] S. Chatterjee and S. Misra, "Optimal composition of a virtual sensor for efficient virtualization within sensor-cloud," in *International Conference on Communications*. IEEE, 2015, pp. 448–453. [Online]. Available: https://doi.org/10.1109/ICC.2015.7248362

[6] M. E. Khansari, S. Sharifian, and S. A. Motamedi, "Virtual sensor as a service: a new multicriteria qos-aware cloud service composition for iot applications," *J. Supercomput.*, vol. 74, no. 10, pp. 5485–5512, 2018. [Online]. Available: https://doi.org/10.1007/s11227-018-2454-y

[7] M. Brown, S. Gilbert, N. A. Lynch, C. C. Newport, T. Nolte, and M. Spindel, "The virtual node layer: a programming abstraction for wireless sensor networks," *SIGBED Rev.*, vol. 4, no. 3, pp. 7–12, 2007. [Online]. Available: https://doi.org/10.1145/1317103.1317105

[8] R. Casadei, D. Pianini, A. Placuzzi, M. Viroli, and D. Weyns, "Pulverization in cyber-physical systems: Engineering the self-organizing logic separated from deployment," *Future Internet*, vol. 12, no. 11, 2020. [Online]. Available: https://doi.org/10.3390/fi12110203

[9] K. L. Bellman *et al.*, "Self-improving system integration: Mastering continuous change," *Future Gener. Comput. Syst.*, vol. 117, pp. 29–46, 2021. [Online]. Available: https://doi.org/10.1016/j.future.2020.11.019

[10] W. Almobaideen, M. Qatawneh, and O. AbuAlghanam, "Virtual node schedule for supporting qos in wireless sensor network," in *2019 IEEE Jordan International Joint Conference on Electrical Engineering and Information Technology (JEEIT)*. IEEE, 2019, pp. 281–285.

[11] A. Fuller, Z. Fan, C. Day, and C. Barlow, "Digital twin: Enabling technologies, challenges and open research," *IEEE Access*, vol. 8, pp. 108 952–108 971, 2020. [Online]. Available: https://doi.org/10.1109/ACCESS.2020.2998358

[12] H. van der Valk, H. Haße, F. Möller, M. Arbter, J. Henning, and B. Otto, "A taxonomy of digital twins," in *26th Americas Conference on Information Systems, AMCIS 2020*. AIS, 2020.

[13] R. Casadei, D. Pianini, M. Viroli, and A. Natali, "Self-organising coordination regions: A pattern for edge computing," in *LNCS*. Springer International Publishing, 2019, pp. 182–199.

[14] M. Viroli, J. Beal, F. Damiani, G. Audrito, R. Casadei, and D. Pianini, "From distributed coordination to field calculus and aggregate computing," *J. Log. Algebraic Methods Program.*, vol. 109, p. 100486, 2019.

[15] J. Beal, D. Pianini, and M. Viroli, "Aggregate programming for the internet of things," *IEEE Computer*, vol. 48, no. 9, pp. 22–30, 2015.

[16] G. Audrito, R. Casadei, F. Damiani, and M. Viroli, "Compositional blocks for optimal self-healing gradients," in *IEEE SASO*, 2017.

[17] D. Weyns, *Introduction to Self-Adaptive Systems: A Contemporary Software Engineering Perspective*. Wiley, IEEE Press, 2020.

[18] D. Pianini, S. Montagna, and M. Viroli, "Chemical-oriented simulation of computational systems with ALCHEMIST," *J. Simulation*, vol. 7, no. 3, pp. 202–215, 2013.

[19] R. Casadei, M. Viroli, G. Audrito, D. Pianini, and F. Damiani, "Engineering collective intelligence at the edge with aggregate processes," *Engineering Applications of Artificial Intelligence*, 2020.