applied sciences

MDPI

Article

Digital Twins, Virtual Devices, and Augmentations for Self-Organising Cyber-Physical Collectives †

Roberto Casadei [1,*], Danilo Pianini [1], Mirko Viroli [1] and Danny Weyns [2,3]

[1] Department of Computer Science and Engineering, Alma Mater Studiorum-Università di Bologna, 47521 Cesena, Italy; danilo.pianini@unibo.it (D.P.); mirko.viroli@unibo.it (M.V.)
[2] Department of Computer Science, Katholieke Universiteit Leuven, 3000 Leuven, Belgium; danny.weyns@kuleuven.be
[3] Department of Computer Science, Linnaeus University, 351 95 Växjö, Sweden
* Correspondence: roby.casadei@unibo.it
† This paper is an extended version of our paper published in the SISSY 2021 workshop on Self-Improving System Integration.

Simple Summary: The engineering of self-organising cyber-physical systems can benefit from a variety of "logical devices", including digital twins, virtual devices, and (augmented) collective digital twins. In particular, collective digital twins provide for a design construct towards collective computing, which can be augmented with virtual devices to improve the performance of existing self-organising applications—as shown through swarm exploration and navigation scenarios.

Abstract: The engineering of large-scale cyber-physical systems (CPS) increasingly relies on principles from self-organisation and collective computing, enabling these systems to cooperate and adapt in dynamic environments. CPS engineering also often leverages digital twins that provide synchronised logical counterparts of physical entities. In contrast, sensor networks rely on the different but related concept of virtual device that provides an abstraction of a group of sensors. In this work, we study how such concepts can contribute to the engineering of self-organising CPSs. To that end, we analyse the concepts and devise modelling constructs, distinguishing between identity correspondence and execution relationships. Based on this analysis, we then contribute to the novel concept of "collective digital twin" (CDT) that captures the logical counterpart of a collection of physical devices. A CDT can also be "augmented" with purely virtual devices, which may be exploited to steer the self-organisation process of the CDT and its physical counterpart. We underpin the novel concept with experiments in the context of the pulverisation framework of aggregate computing, showing how augmented CDTs provide a holistic, modular, and cyber-physically integrated system view that can foster the engineering of self-organising CPSs.

Keywords: cyber-physical systems; self-organisation; digital twins; virtual devices; collective systems; aggregate computing

Citation: Casadei, R.; Pianini, D.; Viroli, M.; Weyns, D. Digital Twins, Virtual Devices, and Augmentations for Self-Organising Cyber-Physical Collectives. *Appl. Sci.* **2022**, *12*, 349. https://doi.org/10.3390/app12010349

Academic Editors: Anikó Costa and Remigiusz Wiśniewski

Received: 30 November 2021
Accepted: 24 December 2021
Published: 30 December 2021

1. Introduction

Recent techno-scientific developments are promoting a vision of smart large-scale Cyber-Physical Systems 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 Network, robotic swarms, people equipped with smart devices, and smart computing ecosystems. Often, such systems are equipped with *autonomic* or *self-** capabilities [3] to provide adaptive system functions and/or promote quality aspects like resilience and efficiency—we call these *Self-* Cyber-Physical Systems*.

In typical Cyber-Physical System (CPS) designs, physical devices have corresponding software counterparts, known as *digital twins* [4], to represent and exploit them in computational settings. There is a close connection between these two parts, essential for effective cyber-physical integration and maintained in implementations by the so-called *digital thread*

process [5]. In settings like Wireless Sensor and Actuator Network and Internet-of-Things, the related but different notion of a *virtual device* (where there is no twinning relationship but rather only hosting or execution relationships with some physical devices) has sometimes been proposed to achieve abstraction or improved Quality of Service [6–9], for example, by abstracting a whole set of underlying physical sensors.

In this paper, we focus on *self-organising Cyber-Physical Systems* [10,11], namely collectives of physical devices monitored and controlled by computer-based algorithms that allow them to autonomously organise as a whole towards global goals. In this context, we sustain a vision whereby logical representations of system entities (e.g., digital twins and virtual devices) are key for: (i) separation of concerns in system design and implementation [11,12]; and (ii) achieving improved application cost/performance profiles, for example, through smart deployment or controlled augmentation of the system to improve or facilitate self-organisation processes [13–15]. Our position is also that programming approaches to self-organisation can especially benefit from such logical representations, for example, in terms of separation of concerns, declarativity, modularity, and execution optimisability.

In particular, we address the problem of modular abstraction and the design of self-organising Cyber-Physical Systems. We aim to address it by decoupling cyber vs. physical components, as per [11], decoupling virtual devices from twinning relationships, and devising corresponding modelling concepts at a collective level. Indeed, in this work, we provide the following contributions:

- We review from literature various kinds of logical devices, and provide a conceptual characterisation including digital twins, virtual devices, and "collective" devices. We do so by distinguishing between the relationships of *identity correspondence* and *execution* existing between logical and physical devices, and by taking both an individual and collective stance (Section 2);
- We re-interpret and conceptually extend the meta-model of self-organising Cyber-Physical Systems presented in [11] in light of the novel proposed concepts, especially discussing how it can express a notion of *augmented collective digital twin* considering digital twins and virtual devices by a global perspective;
- We run an experimental investigation showing how virtual nodes can opportunistically be integrated [15] into an existing self-organising CPS (providing discovery and navigation services in dynamic environments) to improve its performance without changing its logic.

This work extends the conference paper [16] with the following:

- a revision and more formal description of the proposed taxonomy;
- a larger discussion of background, motivation, and coverage of related work (both on virtual nodes and soft control of self-organising systems);
- a whole new experimental part, with a new case study (self-organising exploration of dynamic environments), and a much larger set of simulations for an extensive coverage of parameter configurations.

The manuscript is organised as follows. In Section 2, we review the uses of the concepts of digital twins and virtual devices in the literature and propose a new taxonomy for CPS elements and digital representations. In Section 3, we provide motivation for the paper contribution and describe a reference application scenario where the discussed techniques can be applied. In Section 4, we introduce and discuss the pulverisation meta-model according to the taxonomy, and consider how system augmentations may contribute to self-organisation processes. In Section 5, we evaluate the approach by means of simulations. Finally, Section 6 wraps up and gives an outlook to the future.

2. Digital Twins and Virtual Devices for CPS Engineering: A Review and Taxonomy

In this section, we review the literature on digital twins and virtual devices—both referred to as *logical devices* in this paper, where a logical device can be defined as a software model (e.g., one or more objects, actors, or agents) of an application-level concept (e.g.,

an environment element, a robot, or a wearable-augmented person). Then, we propose an integrated taxonomy based on two concepts: *identity* (of logical and physical devices, possibly related by some notion of correspondence) and *execution* (of logical devices by physical computing devices). Moreover, we introduce the novel notion of a *collective digital twin*, namely the digital twin of an entire collective of physical devices, which we leverage in Section 4 to describe a model of self-organising CPS.

2.1. The Concepts of Digital Twin and Virtual Node in the Literature

2.1.1. Digital Twin

Rasheed et al. define a *digital twin* as a connected, virtual model of a physical entity [4]. The connection—which can be uni- or bi-directional—is established through a *digital thread* [5]. A digital thread realises the data flows that are needed to keep the physical and digital counterparts in sync. The concept of a digital twin is related to other engineering concepts such as "plain old" models, yet, there are subtle but important differences. For example, Fuller et al. make a distinction between a *digital model* (where a manual flow of data ensures the synchronisation between the physical entity and the digital entity), a *digital shadow* (where updates on a physical entity are automatically mirrored on the digital entity, however, not in the other way), and a *digital twin* (where changes are automatically reflected through a bi-directional flow of data between the physical and digital entities) [17]. To make the notion of a digital twin helpful in engineering, it is crucial to clearly demarcate it from related concepts and be precise about its design dimensions and characteristics, enabling to classify its possible realisations. Van der Valk et al. present a taxonomy of digital twins [18]. The authors classify the concept of a digital twin based on the type of data link (uni or bi-directional), the purpose of the twin (repository, transfer, processing), the physical binding (bound or unbound), its accuracy (from identical to partial), its synchronisation (present or not), and the order of creation (first the physical part, first the digital part, both parts created at the same time).

2.1.2. Virtual Node

For a digital twin to exist, there must be an associated physical twin. In contrast, we define a *virtual device* as a logical device that *has no corresponding physical device*, that is, the virtual device does not provide a model for any physical device. In other words, a virtual device models an application-level concept that has no corresponding physical counterpart.

The concept of virtual device has been adopted in multiple works, as discussed in the following and reported in Table 1. Bose et al. refer to 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"* [6]. These authors group virtual sensors in three distinct categories: *singleton* is a virtual sensor that represents an associated physical sensor; *basic* is a virtual sensor that comprises a homogeneous collection of sensors of the singleton class; and *derived* is a virtual sensor that comprises a heterogeneous collection of sensors of the singleton class. Following this view, a virtual device provides an abstraction for a set of underlying physical devices. This leads to the problem of how to find efficient ways for the creation of the aforementioned abstractions, for example, in the context of sensor-cloud infrastructures, Chatterjee and Misr introduce new efficient algorithms for the virtualisation of *groups* of constrained physical sensors in the form of virtual sensors [7]. The concept of virtual node used as an abstraction for groups may serve specific purposes. For instance, Khansari et al. use the concept of a virtual sensor for load-balancing and data-sharing, promoting service compositions that are Quality of Service-aware in cloud-based Internet-of-Things applications [8]. On the other hand, Brown et al. propose a layer of virtual nodes and offer programming abstraction to obtain *reliable* low-cost ad-hoc networks [9]. In this proposal, the programmer has "predictable" *virtual nodes* at his or her disposal to deal with "unpredictable" *client nodes*, that is, the underlying collection of physical devices that are emulated by the virtual nodes. The emulation architecture relies on the "regional coordination pattern" [19] that divides the network into regions regulated by a leader, where the individual nodes coordinate to

support leader election, joining of regions, and consistent data processing. Other works do not associate the concept of a virtual with a group-like abstraction. For instance, Almobaideen et al. use the concept of virtual node to improve the Quality of Service in clustered Wireless Sensor Networks (WSNs) that work using a Time-Division Multiple Access (TDMA) protocol that gives precedence to "critical nodes" [20]. The latter work is especially interesting, as it shows an example of how purely virtual devices may be used to induce application-level effects without direct actions on the physical system.

Table 1. Occurrences of "virtual nodes" in the literature.

Work	Objectives	Techniques	Scenarios	Topology	Type of Virtual Device (See Table 2)
[9]	Predictability	Collaborative emulation of virtual nodes	WSN applications	Ad-hoc	Virtual aggregate
[20]	Improved QoS	TDMA prioritisation	WSN applications	Clustered	Virtual copy
[7]	Abstraction and efficiency	Optimal composition of resource-constrained sensors	WSN applications	Cloud-based (Hierarchical)	Virtual aggregate
[8]	Improved Quality of Service	Quality of Service-aware composition of services	Cloud-based IoT applications	Cloud-based (Hierarchical)	Virtual aggregate
[14]	Swarm behaviour control	Potential field	Mixed swarm-WSN systems	Layered	Virtual device

Table 2. Different identity relations between cyber and physical nodes help us derive different concepts (and corresponding terms). This paper has an explicit focus on the notions in the highlighted cells.

Correspondence of Logical Identities with Physical Identities	Logical Device (Through Their Software Components)	Physical Device	Description
one-to-zero	Virtual device	–	A logical device identifies with no physical device.
zero-to-one	–	Infrastructural device	A physical device identifies with no logical device (e.g., it only provides execution support).
one-to-one	Digital twin	Physical twin	A logical device identifies with exactly one physical device.
one-to-N, $N > 1$	Virtual aggregate	Physical view/copy/member	A logical device identifies with a group of physical devices.
N-to-one, $N > 1$	Digital view/copy/member	Physical aggregate	A physical device identifies with a group of multiple distinct logical devices.

2.2. Logical and Physical Devices: A Taxonomy

2.2.1. Background: Cyber-Physical Systems (CPS)

A *cyber-physical system (CPS)* is a system where software and physical components are connected in a feedback loop and carry out sensing, actuation, and computation (communication can be regarded as a form of sensing and actuation). Of such aspects, sensing and actuation typically need to be *situated*, that is, *localised* in the physical context to accomplish their application-specific function of perceiving and affecting the environment. The situatedness is obtained by deploying concrete sensors and actuators in the physical world, for example, by embedding them in a larger device (such as a smartphone or a drone). Conversely, computations may be often run (almost) *anywhere* and hence be *offloaded* from the concrete devices. However, since data must be communicated to/from computational processes, there might exist constraints on where computations can actually be executed while preserving functionality. Still, the possibility of detaching computation from for example, the device requesting a service or the device to be controlled, provides engineers with larger design spaces where trade-offs (in terms of flexibility or performance metrics) can be made.

In general, it is possible to distinguish physical devices based on their role in the design of the system. For instance, it is significant to distinguish between *application-level* physical nodes (APN), which are the target of monitoring and control in the application; and *infrastructure-level* physical nodes (IPN), which are those providing support, for example, in terms of networking, storage, and execution offloading. For example, consider a cyber-physical swarm: the APNs are the robots, while the IPNs may consist of edge servers upon which certain computations can be offloaded (e.g., allowing robots to save energy). Essentially, a typical APN may be a sensor, an actuator, or a larger device (such as a smartphone or a drone) packaging multiple sensors, actuators, and computing units together.

Another sensible demarcation in the CPS engineering approach, based on the nature of the modelled entities, is between *logical* devices (which exist as abstractions or components in models), *software* entities (providing a representation of logical devices in terms of computational and possibly deployable components), and *physical* devices (namely, actual entities existing in the real world—such as smartphones and drones). Notice that the logical and software devices are related in a way similar to digital twins with their digital threads.

2.2.2. Taxonomy

As per the above discussion, it makes sense to distinguish between *identity correspondence* and *execution relationship* to characterise the relation (mediated by software components) between logical devices and physical devices. The result of such a characterisation is summarised in Tables 2 and 3. In the following, we focus on the former, whereas the latter is discussed in Section 4.

Let $\mathcal{L}$ be a set of unique logical identities (which allow us to distinguish between different software components—for example, one Java object from another) and $\mathcal{P}$ be a set of unique physical identities (which allow us to distinguish between different physical components—for example, one robot from another).

Moreover, let $\mathcal{N}$ be a label used to distinguish different associations. Then, we can define digital twin, virtual device, and other notions based on a labelled binary symmetric relation $\mathcal{I} \subseteq \mathcal{L} \times \mathcal{N} \times \mathcal{P}$ that exists between logical and physical identities (and corresponding roles) and that we call *identity correspondence*. Accordingly, we call a *virtual node (VN)* an entity whose logical identity $l \in \mathcal{L}$ has no corresponding physical identity, that is, $(l, \rightleftharpoons, p) \notin \mathcal{I}, \forall p \in \mathcal{P}$ for a given label $\rightleftharpoons \in \mathcal{N}$. With abuse of language, we denote entities by their identities and vice versa, and so we also say that the virtual device has no corresponding physical device. Similarly, we call an *infrastructural physical node (IPN)* a physical device $p \in \mathcal{P}$ which has no corresponding logical device, that is, $(l, \rightleftharpoons, p) \notin \mathcal{I}, \forall l \in \mathcal{L}$; we can think of such a device as providing "infrastructural services" in terms of execution support or environmental functionality. Instead, a *digital twin (DT)* is a logical device $l \in \mathcal{L}$ which has *one* corresponding physical device $p \in \mathcal{P}$, known as its *physical twin (PT)*, that is,

$(l, \rightleftharpoons, p) \equiv (p, \rightleftharpoons, l) \in \mathcal{I}$. Considering multiplicities, further notions could be derived as shown in Table 2.

Table 3. Different execution relationships between cyber and physical nodes help us derive different concepts (and corresponding terms). This paper has an explicit focus on the notions in the highlighted cells.

Execution Relationship of Logical Devices with Physical Devices	Logical Device (e.g., via Software Components)	Physical Device	Description
one-to-zero	Undeployed/idle device	–	The logical device is not executed by any physical device.
zero-to-one	–	Free/thin host	The physical device does not (or cannot) host any logical device.
one-to-one	Embedded logical device	Host	A logical device is executed by exactly one physical device, namely its host.
one-to-N, $N > 1$	Distributed/Pulverised device	Pulverisation infrastructure element	A logical device is executed in a distributed fashion by a group of physical devices.
N-to-one, $N > 1$	Tenant	Server	A collection of logical devices is executed by an individual physical device.

Then, we define a *collective digital twin (CDT)* as a (possibly dynamic) collection of DTs. Similarly, we define a *collective physical twin (CPT)* as a (possibly dynamic) collection of PTs. The relevance of such conceptual constructs is fostered by emerging paradigms like *aggregate computing* [21,22] that address collectives as whole computational machines, namely by a global perspective. Indeed, a CDT could also be reified as a single, logical device corresponding to a group of physical devices—what has been called a *virtual aggregate* in Table 2. Finally, we consider *augmentations* as systems involving also entities not appearing in $\mathcal{I}$. So, an *augmented CDT* is a CDT including virtual devices, whereas an *augmented CPT* is a CPT including infrastructural devices. Notice that logical augmentations indirectly affect physical devices through their DTs and, vice versa, physical augmentations indirectly affect logical devices through their PTs. So, we can also say that a CPT (resp. CDT) is augmented if so is the corresponding CDT (resp. CPT). Refer to Figure 1 for a pictorial representation of this idea.

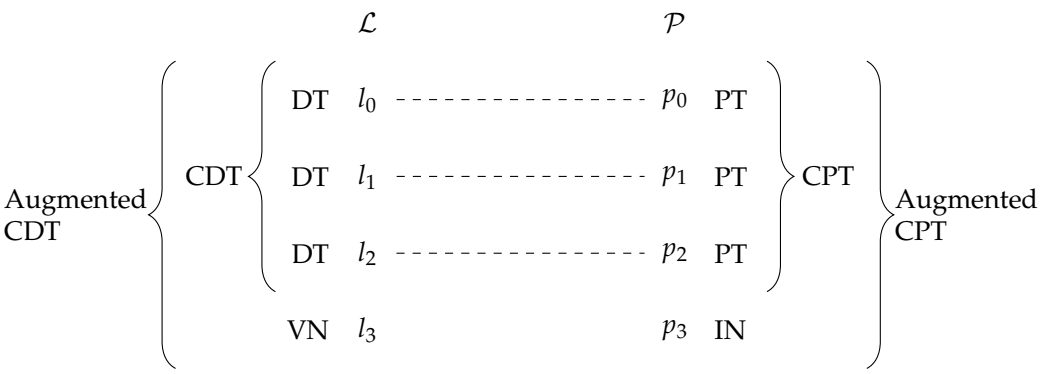

Figure 1. Augmented collectives of digital and physical twins.

3. Motivation and Related Work

Our target type of system is a *cyber-physical collective*, namely a collection of interacting cyber-physical components situated in some cyber-physical environment (e.g., an actual city plus ICT infrastructure including e.g., cloud data centres). Examples include robot swarms, crowds equipped with smart wearables, smart buildings, or smart cities. Our goal is to design the collective, self-organising behaviour of the entire system of devices, which involves coordinating and collectively computing local data and actions for each individual device such that the desired global goal is achieved (or pursued, in case of never-ending operational goals—cf. so-called *eternal systems* [23]).

To this end, one approach is to define a DT for each physical device of the collective, and then implementing communications between the DTs and computations to determine the actions they would need to perform, which may ultimately drive the actuators of the corresponding physical devices. This approach is followed by the aggregate computing paradigm, which enables the definition of a global program, to be repeatedly executed by all the DTs, which is meant to specify the collective behaviour of the whole. Using the terminology introduced in this paper, the aggregate computing approach considers the CDT of the physical collective as the "target computational machine".

The implementation of the digital thread may have a non-functional (and sometimes even functional) impact on the application. In this regard, it is important to consider two main elements: (i) the partitioning of the digital thread logic into deployable software components; and (ii) the deployment strategy, namely the kind of mapping of software components to the available physical devices. The most straightforward deployment is where each DT is hosted by the corresponding PT, but more complex systems can be designed. The *pulverisation meta-model* [11] provides exactly that: a partitioning schema and formal model for describing deployments of self-organising Cyber-Physical Systems like aggregate computing systems. However, previous research on aggregate computing and pulverisation model did not explicitly consider the notions like digital twin, virtual device, and the other concepts introduced in Section 2.2. In particular, we would like to show that the notion of virtual devices can be especially useful for improving or driving self-organisation processes.

3.1. Reference Example: Self-Organising Navigation and Exploration in Dynamic Environments

Virtual nodes can support situated computations in dynamic environments, for example, in applications where the underlying structure of the environment needs to be mapped (i.e., represented) on a distributed system and exploited computationally. The challenge is to adapt distributed state and behaviour to continuous changes—including those induced by the computation itself (e.g., through hard or soft control on mobility of devices).

For instance, consider a post-disaster scenario, in which an urban area has been hit by a natural disaster (e.g., an earthquake or a hurricane) which damaged the local infrastructure, occluding roads and making the pre-existing information about road connections obsolete. A similar scenario occurred in central Italy between 2016 and 2017, when a tragic seismic sequence caused multiple building collapses that in turn occluded roads [24,25]; the geophysics changes induced by the earthquakes [26] triggered landslides [27] and avalanches [28] few weeks later, continuously disrupting the actual shape of the road infrastructure. Consider for this scenario a team of explorers who can sense their own position. The explorers could be human beings equipped with a portable device or unmanned ground vehicles (UGV). We assume that devices are able to communicate to every other device, either directly (e.g., via long-range wireless links such as LoRaWAN), or through the support of a local network node, (e.g., leveraging a temporary emergency network access using airborne base stations [29]), or through to the cloud (for instance, in the case the event did not disrupt the cellular network infrastructure, or in the case it had been recovered quickly enough). In other words, we assume that the network is not segmented, or that at least that segmentations are momentary (e.g., cellular network signal losses, or temporary occlusion of the line-of-sight connection with the airborne base station). The goal is to

explore the area and build a reachability map. To do so, the system runs an aggregate program that collectively computes paths among devices in real time, considering their current position. The aggregate system considers *logically* connected devices within some range, and spawns a new virtual device situated at its current position whenever no other device is logically reachable (isolation), or all the reachable devices are considered "too far away". The idea is to exploit logical connectivity to create a live physical reachability map quickly, at the expense of some virtualisation overhead, as it is likely that devices separated by a short space are directly reachable in physical space.

A similar scenario is crowd-aware navigation. The system we consider counts hundreds of nodes deployed in an urban environment: these may include infrastructural elements (e.g., fog nodes), smart city devices, smartphones, and wearables held by people. The system is programmed via aggregate computing [22], and collectively computes an estimate of the local crowding in order to handle "navigation requests" in a crowd-sensitive fashion—extending over the crowd tracking example seen in [21]. The system's goal is to react to navigation requests to reach a Point of Interest (PoI), providing as a response a route that does not get across overly crowded areas. Of course, since the phenomenon is dynamic, such a route may need to be readjusted at runtime. One possible collective solution is to leverage multiple *gradients* [30] to build a spatial data structure carrying information about position and direction of neighbouring nodes, distorting the information about distance considering the crowding level (e.g., as perceived with Bluetooth proximities), and thus making overcrowded areas appear naturally farther away than they actually are to model the inconvenience and risk of passing through these locations. Further, these gradients can be leveraged to sustain a collective infrastructure known as the self-healing "channel". Devices that are located inside the channel represent safe locations that can be followed as waypoints to move from the current location to the PoI.

Usually, the approach is implemented through repeated computation of the self-organising logic in asynchronous rounds (although some models promote reactivity [31]), interwoven by exchanges of the information processed locally with devices considered to be logical neighbours. Devices can be neighbours because of the physical structure of the network (e.g., because they are communicating through a direct channel, such as a managed Wi-Fi network), or, more commonly, because of a logical network layer. In principle, the programming model takes no stance on the nature of a device: it could be a piece of hardware as well as a digital twin corresponding to some situated physical device. The consequence of this flexibility has been partly explored in [11], where different architectural deployments, with very diverse performance and cost profiles, are shown to be viable when implementing the "digital thread" (synchronising sensor readings from physical sensors to digital twins, forwarding actuation commands to physical twins, and dealing with the network required to enable logical interaction).

Logical decentralisation makes the proposed approach to crowd-aware navigation scalable. Additionally, it is agnostic to the specific network infrastructure: it can work with mobile, opportunistic ad-hoc networks, and thus be available even when cloud-based navigation services such as Google Maps are not available. The sole requirements are: the ability to communicate with other devices, the capacity to provide an estimation of the distance to neighbours, and the ability to locate the PoI when in-range. Of course, the approach also has limitations: on the one hand, reachability issues may arise if crowded areas are considered entirely inaccessible (the channel may not be formed); on the other hand, diverse device density across the area may result in spotty service quality, with areas less covered by devices getting underexploited during path computation. To mitigate the latter issue, we consider spawning virtual devices to support coverage for areas with a low density of physical nodes, improving the application performance and availability, potentially finding shorter paths, at the cost of some additional virtualisation overhead. See Figure 2 for a graphical example of the idea.

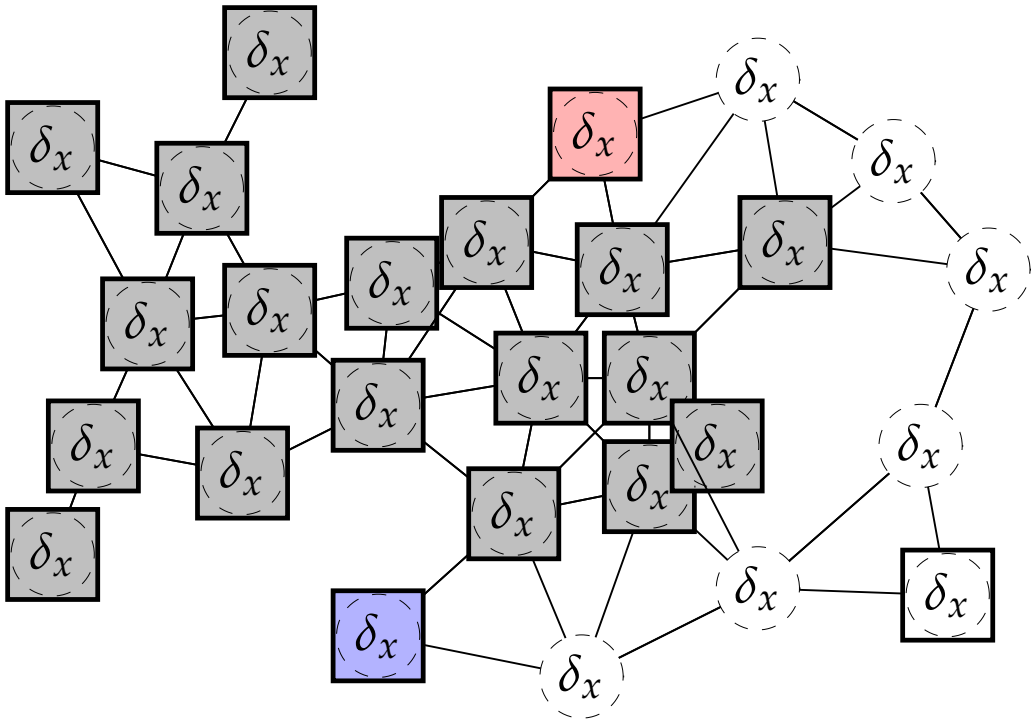

Figure 2. Pictorial representation of the crowd-aware navigation scenario. Physical devices are depicted with a solid square, logical devices are depicted with a dashed circle; an overlap of the two indicates that a logical device is associated to a physical device, a circle outside any box denotes a purely virtual device. Nodes sensing overcrowding are depicted in grey, the requester is coloured in blue, and the PoI is filled in red. The requester desires to reach the red device, walking along a path that can avoid overcrowded areas. Considering solely the physical devices as part of the system, no available route can be computed in a self-organising way; however, by populating the system with virtual nodes, a route circumventing the problematic area can still be found. The system could also (in principle) continuously look at strategies to self-improve such an integration for non-functional benefits.

3.2. Summary and Related Work

In a nutshell, the motivation of the proposed *augmented CDT* notion has two parts:

1. *Motivation for the CDT notion.* Taking a collective viewpoint leverages abstracting groups of devices (cf. virtual aggregates in Table 1) and "macro-programming" groups of devices, as discussed previously in this section. This aligns with approaches like aggregate computing that are explicitly designed to address the global behaviour of collective adaptive and self-organising systems.
2. *Motivation for the "augmentation" of CDTs.* The part of augmentation leverages on concepts proposed in the literature, such as virtual agents, that are exploited for soft control of swarms [13,14].

Therefore, the novel concept of augmented CDT integrates the collective viewpoint with augmentation, making it—we believe—especially interesting and a useful contribution towards tackling some challenges of self-organising CPS engineering.

Our proposal differs from related work for it provides an integrated approach that considers: (i) a meta-model where logical entities are decoupled from physical entities (see Section 4 for details)—like in [32,33]; (ii) a distinction between identity and execution relationships, allowing to distinguish between digital twins and virtual devices; and (iii) a collective computing approach whereby the focus is on collective-level behaviour, rather than individual behaviour (so, e.g., we do not assume behavioural differences between digital twins and virtual devices).

- **Models for self-organising Cyber-Physical Systems.**

The approach aims to provide a structure for self-organising collective systems. This is in contrast with approaches that distinguish between *managed system* and *managing system*—the latter being typically organised around the MAPE-K pattern [3]. In particular, there are similarities with the idea of *self-organising software models* [32], where the goal is to reduce the need for manual composition. In our case, we assume a composition of collective behaviours is specified by a developer (e.g., using aggregate programming), and let the platform deal automatically with deployment and possibly with the de/augmentation (management of virtual devices) for example, given high-level goal specifications. A similar approach for separating cyber entities from physical entities as well as other responsibilities when modelling self-organising Cyber-Physical Systems is provided in [33], with a focus on production systems; however, it somewhat neglects the collective viewpoint, and does not clarify a difference between digital twins and virtual devices. Other approaches exist for general specification and analysis of dynamic architectures, including *DR-BIP (Dynamically Reconfigurable - Behaviour Interaction Priority model)* [34] and *DReAM (Dynamically Reconfigurable Architectural Modelling)* [35], using meta-entities like *components* (for modelling behaviour), *connectors* (for modelling interaction), *maps* (for modelling logical connections), and *deployments* (for mapping components to deployment loci), and *motifs* (modelling architectural configurations that may change at runtime); however, they are not tailored to self-organising systems.

- **Self-organisation algorithms and their control.**

Our reference example leverages self-organising algorithms such as gradients [30] and *information flows* which are typically used in multi-scale and situated systems [36,37]. Our emphasis is not on the algorithms themselves; rather, we show how self-organisation, for example, based on information flows, can be improved through virtual devices. The general idea of using a kind of internal or external intervention for driving the emergent behaviour of a "given" system (for which e.g., the behavioural and interactional rules of the member agents are not considered to or cannot be re-designed) is not new, and has been referred to with different terms like *soft control* [13,38], *control kernel* [39], and *pseudo-emergence* [40]. For instance, in [13], a "shill agent" is used for soft control of a flock of birds—idea leveraged in subsequent works proposing multiple "influencing agents" [41]. In [42], the authors introduce a notion of "virtual leader" to control groups of vehicles; however, the virtual leader is not really an entity but rather a construct used to reify additional artificial potentials. Similarly, in [43], decoy objects are introduced to improve performance in swarm searching problems. Finally, in [14], notions of virtual individuals, virtual targets, and virtual pheromones, are proposed as a way to combine WSNs and robot swarms to steer collective behaviours of the latter. In this paper, we bring and extend these ideas to cyber-physical aggregate computing settings, where the influencing agents are virtual and the control target is the collective physical twin of a virtually-augmented collective digital twin. Moreover, differently from the aforementioned related work, the introduced virtual entities will play *the same control behaviour* of the other, regular entities—hence showing an interesting, original "structural" interplay with self-organisation algorithms. Indeed, we also suggest further uses for virtual entities (see Section 3.1) that, to the best of our knowledge, are not present in literature. Additionally, we suggest that dynamic addition and removal of virtual devices can be key from a self-improving system integration perspective [15].

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

In this section, we reinterpret and extend the meta-model of self-organising Cyber-Physical Systems introduced in [11] (Section 4.1) according to the conceptual framework of the previous section, then show how it can be used to support purely virtual devices (Section 4.2).

4.1. The Pulverisation Meta-Model of Self-Organising Cyber-Physical Systems

The *pulverisation meta-model* of self-organising Cyber-Physical Systems [11] can be summarised with the following elements:

1. *Platform* subsystem: an abstraction over the physical networked system consisting of *hosts* possessing a *network identity*; these can be either physical (bare metal), virtualised, or containerised systems, as far as they can be identified as network devices (e.g., through an IP address or a URI), and can communicate with other hosts.
2. *Cyber* subsystem: a network of *logical devices* connected as per a logical *neighbouring relationship*; logical devices are segmented (i.e., "pulverised") into five *logical components* (computation β, sensors σ, actuators α, state κ, and communication χ); these can be deployed independently (see Figure 3), as far as the physical devices where they are located can communicate with each other.
3. *Deployment*: a cyber-physical mapping between logical device components and hosts, defining the way the former are *deployed* on the latter (the so-called execution relationship). See Figure 4.

In such a system, the software architect focusses on the cyber subsystem, while the system integrator and the middleware provider focus, respectively, on the platform and the deployment.

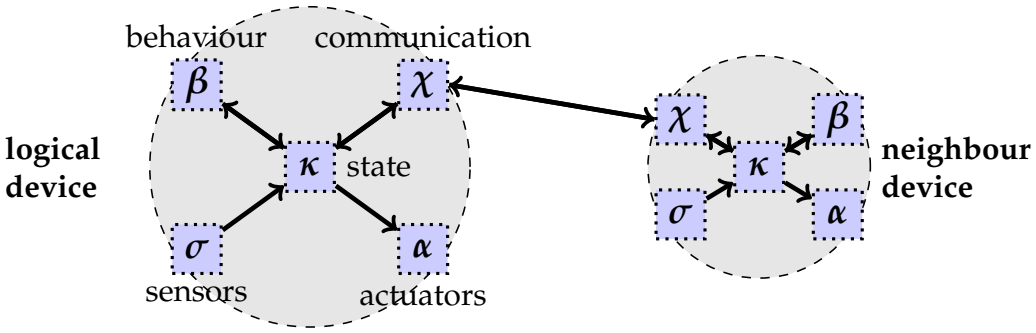

Figure 3. Two neighbouring logical devices, depicted as pulverised into their logical components. Arrows mark directional data flows among pulverised components. The neighbour relationship is reified by the existing connection between the two communication components of the devices.

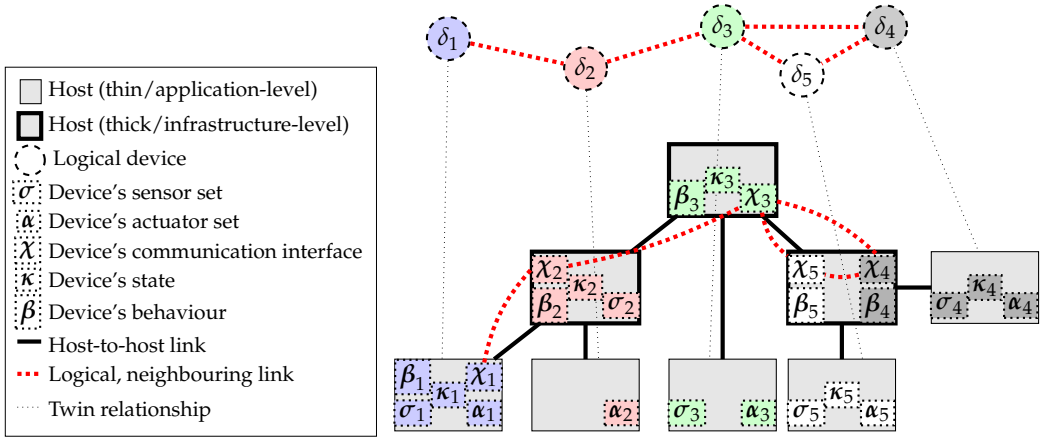

Figure 4. A view of an instance of the extended pulverisation architectural meta-model. The image shows the correspondence between the logical system (**top**) and its actual physical deployment, over an infrastructure enriched with support nodes (e.g., fog devices, edge servers, or cloud service providers). Different colours are associated with different logical devices: in some cases, components (dotted boxes) of a single logical device (dotted circle) are spread across multiple physical nodes (solid boxes), as intra-logical-device connections (not shown here, refer to Figure 3) are allowed to cross the host boundaries. Logical inter-device connections are represented with thick dotted red lines.

The meta-model enforces an architectural organisation principle that *fosters* deployment-independent logic, thus fostering self-organisation by allowing the software designer to reason without any constraint in terms of the underlying platform. Although providing an ideal substrate, the architectural model by itself does not imply self-organisation: behavioural and interaction logic are still to be designed and injected into the logical components. More specifically, self-organising behaviour can be achieved by organising the roles of the pulverised components as follows:

- Sensors σ perceive the context if the logical device, integrating their readings into the device state κ;
- based on the current state, the behaviour component β performs a "reasoning step", and accordingly updates the latter;
- the communication component χ integrates information from neighbours into the device state κ, and shares the results of the reasoning step with the χ components of neighbouring virtual devices to enable coordination;
- finally, Actuators α retrieve state information to control the device itself.

The model abstracts away the actual implementation of the logical components and does not mandate any specific platform. In [11], the model has been instantiated into the aggregate computing framework [21,22]. In this incarnation, all devices share the same behaviour component β; namely, they run the same program, represented by a pure function operating over the previous state κ and producing a new state that includes messages to be forwarded through χ and actuations to be performed by α. In this setup, the cost and performance of an actual system vary importantly with different deployments of the logical entities over the available edge-fog-cloud infrastructure, as shown by the cost-performance profiles presented in [11]. Importantly, the model is simulation-friendly, and allows informed decisions on the cyber-physical mapping to be performed before the actual deployment of the system. In this work, we extend [11] to its extreme consequence, by considering the integration of purely virtual devices into self-organising CPSs.

Pulverised Aggregate Model of the Reference Example

Here, we briefly explain how the reference example presented in Section 3.1 can be modelled using the aggregate instantiation of the pulverisation framework—whose details can be found in [11].

- Each explorer logically maps to a logical device δ, which is partitioned into sub-components as per Figure 3;
- The behaviour β consists of a round-based execution of an aggregate program expressing the logic of coordination and exploration (the technical details are not important here but the program is publicly available at [44]);
- The sensors σ include those logical sensors needed by program β; in this application, we require a identity sensor (providing the UID of the device), a position sensor (e.g., providing the GPS coordinate of the device), and a sensor for estimating the distance from neighbours;
- The actuators α might include a clue for the direction of movement (e.g., for a human with a smartphone), or a high-level control for moving in that direction (e.g., for an autonomous robot);
- The communication component χ handles round-based interaction with neighbours—identified according to a logical neighbouring relationship based on the estimated spatial distance—cf. Figures 2 and 4. The specific content of the message broadcast to neighbours depends on the executed aggregate program;
- A deployment delivers such components, properly integrated into a working system/middleware, onto physical devices for execution—cf. Figure 4. For instance, a straightforward deployment would deploy logical devices to the computer of each corresponding explorer; however, more complex deployments are possible in general [11] (e.g., only σs and αs on the explorers' physical devices and the other sub-components on the cloud).

Additional details about this configuration are provided when discussing the simulation environment in Section 5.

4.2. Augmented Collective Digital Twins in the Pulverised Aggregate Computing Meta-Model

4.2.1. Digital Twins and Collective Digital Twins

The meta-model discussed in detail in [11], captures the execution relationship through a deployment relation specifying the cyber-physical mapping, but does not consider the twinning relationship, which is enforced implicitly depending on the deployment of the sensors σ and actuators α components of each logical device (digital twin) to the designated host (physical twins). Indeed, whereas computation, communication, and state can be offloaded and disembodied, sensors and actuators should typically be situated together with their physical twins in order to properly perceive and act, in real time, on their very context. Instead, we propose to make the twinning relationship explicit—see Figure 4. Making the connection explicit, in terms of an identity correspondence between logical devices and physical hosts, has both a descriptive role and an operational role. For instance, it may be used to define custom deployment constraints or digital thread mechanisms.

Moreover, the cyber subsystem is ultimately a network of logical devices, and as such it can be seen as the *collective digital twin* of the *collective physical twin* composed of the hosts of the platform subsystem. In aggregate computing, this is even more evident since the aggregate program defining the behaviours β is conceptually associated with the system as a whole. The benefit of considering the aggregate abstraction of a CDT lies in the possibility of getting and reasoning about collective descriptors of the system, expressed, for example, in terms of *computational fields* [22] mapping each device to a corresponding computational value (such that, e.g., a temperature field maps each device to the temperature it senses, a Boolean field represents a domain of devices, for instance denoting those available to provide a service, or those belonging to a team, and a field of percentages may denote the level of charge of the devices in the system).

4.2.2. Augmentation with Virtual Devices

The platform subsystem is "augmented" with an additional (not necessarily disjunct) set of infrastructural nodes supporting improved connectivity, computation offloading, and other non-functional requirements. Similarly, the cyber subsystem can be "augmented" through the addition of purely virtual devices, not mapped (in terms of identity correspondence) to any existing host, giving birth to what we call an *augmented collective digital twin*: these virtual devices constitute an *augmented reality* for the physical twins.

Since they have no direct correspondence with any application-level host, sensors and actuators of virtual devices need to be virtualised. Obviously, there must be at least one physical device providing the mere *execution* support (i.e., in terms of virtualisation) for any given virtual device. A virtual device exists for another device once the latter can perceive the former: the details of how this existence is reified essentially depend on the interface component χ (which is the mediator for the interactions between any couple of devices) and how the state κ of the virtual device is populated. One approach is to model the virtual device exactly as a normal device, that is, by letting the state be determined by proper sensors σ, messages χ, and behaviour β. In particular, virtual devices—being decoupled from reality (besides mere execution needs)—can be positioned at arbitrarily in space, and their position and distance sensors can be populated based on the position information of devices bound to physical hosts. In other words, as far as it is possible to generate reasonable data for synthesised sensors and actuators and the network performance of the specific deployment (in particular, information propagation speed) remains adequate for the application at hand, virtual devices and their physical hosts are *spatially decoupled*. For the virtual devices to be part of the logical network, the middleware treat them as it does with physical devices: it must synchronise, collect, reason about, and propagate information as it would for any ordinary device, such that any of its neighbours perceive and interact with it exactly as if it were real. Achieving a completely transparent emulation is relatively easy in centralised deployments (e.g., cloud-based or hierarchical), but it grows

increasingly challenging with the degree of decentralisation of the physical network. In particular, in decentralised settings there is the problem of network partitions which may lead to inconsistencies or availability issues (cf. CAP theorem [45]). It seems opportune to assume a single reachable and available "virtualiser node" owning a given virtual device, otherwise the determination of the state of a virtual device, and interaction with it, becomes a problem of distributed consensus. Another typical requirement for spatial situation of virtual devices is the availability of a global coordinate system and accessibility of services like the GPS. Finally, whereas the set of virtual nodes (and the corresponding situation) might be determined statically, an interesting opportunity arises when a more dynamic management is considered.

In summary, augmenting a system with virtual devices requires solving two key problems:

1. *virtual device synthesis*—which involves determining what must be virtualised and how (namely, filling it the details);
2. *virtual device integration*—which involves determining how to make the actual system perceive the virtual device as a regular device.

Such problems can, and often need to be addressed at runtime.

4.2.3. Self-Adaptive Management of Virtual Nodes

The main insight of the paper is that virtual devices can be leveraged to enable new functionality and improve the application performance *reusing the pre-existing self-organising logic*. As an example, consider the swarm exploration scenario of Section 5: a system enriched with virtual devices better covers or approximates physical environments, hence enabling the computation of additional, possibly shorter, and more precise reachability maps. In other scenarios like crowd-aware mobility systems, the creation of virtual devices can be leveraged to *prevent* navigation paths from using some slices of space and time—for example, as a way to reify a prediction of a future crowding due to planned events. Other use cases include mixed-reality simulations, in-field testing (cf. chaos engineering), or learning.

Depending on the actual deployment conditions (and especially in decentralised implementations) the execution of virtual devices may introduce a significant overhead. Consequently, the existence of virtual devices might be ephemeral, they could be spawned or torn down depending on *opportunistically* [46], making the case for a *self-adaptive managing system* [47]. Interestingly, the approach could be used at the lower level to implement the management platform itself, resulting in a lower application layer programmed with the same principles, where the collective system reifies a distributed virtualisation platform that is in turn exploited at the application level. The exploration of such a strategy is left as a potential future direction of research.

5. Evaluation

Our main evaluation goals are the following:

- show how purely virtual nodes, that do not exist physically but are synthesised virtually at static time or runtime, can improve the performance of a self-organising cyber-physical system;
- understand under which conditions the proposed techniques are effective;
- roughly understand (via proxy metrics) how the cost of keeping the system in operation varies with changes in the setup.

5.1. Case Study: Post-Disaster Area Exploration

We consider a post-disaster area exploration application as a case study, inspired by the Italian seismic sequence events mentioned in Section 3.1. The system consists of a team of N_r explorers that can sense their own position, the aggregate system considers *logically* connected devices within a range R, and spawns a new virtual device situated at its current position whenever one of the following conditions hold:

- no other device is logically reachable (isolation); or

- all the reachable devices are at a distance higher than a parameter R_v.

The idea is to exploit logical connectivity to create a live physical reachability map quickly, at the expense of some virtualisation overhead: low values of R and R_v are expected to raise the system precision, as it is increasingly likely that devices separated by a short space are directly reachable in physical space.

5.2. Experiment Setup

In order to exercise the idea on a challenging environment, we setup our experiments to run in the city map of Urbino in central Italy. We picked the city as it features a medieval, irregular road structure with sharp turns, U-shaped roads, and considerable altimetric changes; making exploration harder and naturally producing the potential problems that the system would encounter in a real-world situation.

N_r explorers, equipped with a physical device, are deployed at random location in the urban area. These are instructed to explore the area by generating random destinations and renewing them every time they are reached (or, if they are unreachable, the closest point on a reachable road has been reached).

Explorers move at a mean speed of 1.4 ms, which is a reasonable approximation of the average walking speed of a pedestrian [48] (although people in charge of exploring would probably walk faster than the average person, the terrain in a post-disaster situation is often rough and attention is advised, we thus think that the mean speed should not differ too much). Devices, both physical and virtual, run rounds asynchronously with an average frequency of 0.5 Hz. We assume message delivery to be instantaneous, namely, we do not simulate network delays. This choice was made, as reasonable network delay modelling would require assumptions on the specific deployment (including networking technologies and protocols and actual execution location of pulverised logical components) that we do not want to investigate in the current work, whose focus is on evaluating the approach regardless of the actual deployment details. We simulate one hour of area exploration for every run (3600 s). Figure 5 depicts subsequent snapshots of a simulation run, providing an idea of the correspondence between deployed virtual devices and road shapes, as well as showing the reachability map formation with time.

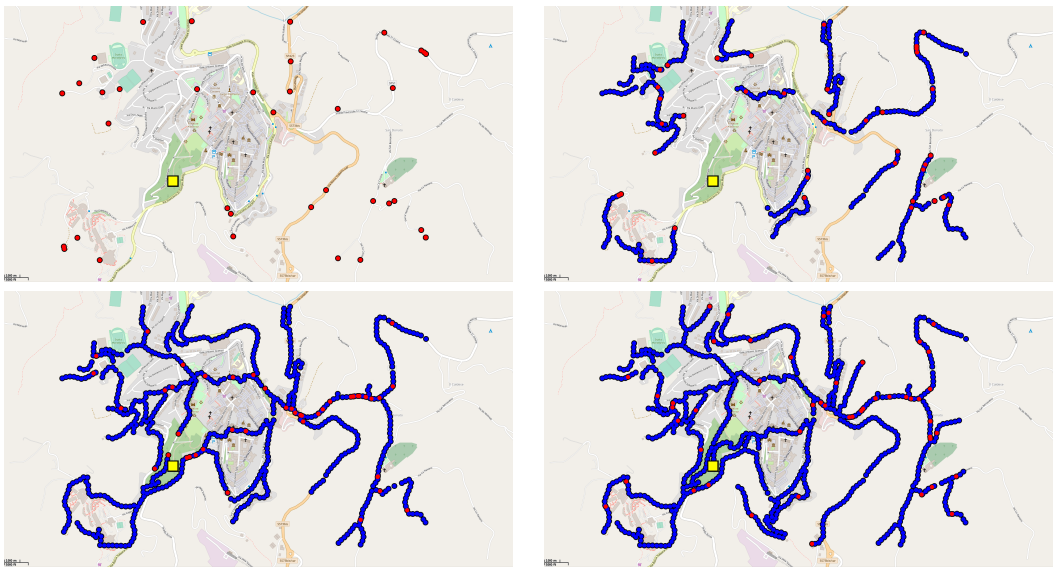

Figure 5. Snapshot of the simulation. Initially, (**top left**) explorers (red dots) are deployed in random positions in the simulation arena. A PoI (yellow square) is generated randomly as a reference point for route computation. When the simulation begins (**top right**), explorers begin wandering the map along the available routes, and when conditions are appropriate, they spawn virtual devices (blue dots). With time, explorers indirectly increase the knowledge of the actual road structure (**bottom**).

5.2.1. Metrics

Given the set of unique device identifier associated to physical devices $H \subset \mathbb{N}$, we evaluate the system performance by measuring the following metrics, summarised in Table 4:

- *distance error* δ_j, defined, for each *physical* device with id $j \in H$, as the difference in length between the route towards a PoI computed by the collective system and the actual shortest length as provided by an oracle, provide a measure of precision in route estimates;
- *isolated device count* I, defined as the number of *physical* devices located on a *logical* network segment that does not allow to find any valid route to the PoI, provides a measure of how densely the area is being covered;
- *virtual device count* N_v, defined as the number of *virtual* devices currently operating, is a proxy metric for the cost in computational resources;
- *communication cost* M_j, defined, for each *physical* device with id $j \in H$, as the occupied dimension of the buffer for received messages normalized by the size of the last message sent—in other words, it measures how many messages, produced either by real or virtual devices, have been received by the real node with id j. Higher values indicate more stress on the network capacity. Since messages require processing, if the behaviour component β of pulverised devices is executed on physical devices (as it is likely in the case of "thick" end devices, such as modern smartphones), then this metric also correlates with higher battery energy consumption.

Table 4. Summary of all symbols used in the evaluation.

Symbol	Meaning	Unit	Values
δ_j	distance error for device j	m	n.a. (metric)
$\overline{\delta}$	mean distance error: $\sum_{j \in H} \delta_j N_r$	m	n.a. (metric)
I	isolated devices	devices	n.a. (metric)
M_j	comm. cost for device j	$\frac{messages}{round}$	n.a. (metric)
H	set of physical devices UIDs	adim.	n.a. ($H \subset \mathbb{N}$)
$\sum_{j \in H} M_j$	total communication cost	$\frac{messages}{round}$	n.a. (metric)
$\overline{M}$	mean comm. cost: $\sum_{j \in H} M_j N_r$	$\frac{messages}{round}$	n.a. (metric)
N_v	virtual device count	devices	n.a. (metric)
N_r	physical devices (explorers)	devices	$5, 15, 50, 158, 500$ ($N_r = \lvert H \rvert$)
R	logical communication range	m	$10 \cdot (1, 2, 5, 10, 20, 50, 100, 200)$
$R_v R$	relative spawn range	adim.	$0.25, 0.5, 0.75, 1, 1.25, 1.5, \infty$
R_v	virtual device spawn range	m	n.a. (derived as $R \cdot R_v R$)

5.2.2. Parameters

We let the system run varying the following parameters, whose meaning and set of tested values are also summarised in Table 4. We run a simulation for each combination of the variable values (we test the Cartesian product of the parameters values).

- *Logical communication range* R: the maximum distance as the crow flies at which communication between devices is allowed. We explore the system behaviour with a large range of options, exploring the possible space approximately geometrically, ranging from 10 m to 2000 m. The former option builds a logical network that behaves similarly to an opportunistic network with short range radio systems (e.g., Bluetooth 2.0); the latter option, considering the map size of the town centre of Urbino, implies a fully connected map. We experiment by (approximately) doubling R in

each simulation set. Since the performance of the system is largely unknown, we preferred to investigate using an approximately geometric progression in place of a linear progression.

- *Virtual device spawn range* R_v: the maximum distance from the closest virtual device than a physical device tolerates before spawning a new virtual device in its current position, when set to ∞ disables spawning new devices entirely. We decided to control this parameter indirectly by tuning the relative span range $R_v R$. The decision was taken after an initial exploratory testing which showed that varying R and R_v independently produced results of little significance and/or extremely expensive simulations; for instance, setting $R = 2000$ m and $R_v = 10$ m quickly produced a fully connected network of tens of thousands virtual devices, which had extremely high distance error δ and cost (both in terms of virtual device count N_v and communication cost M_j), and also resulting in a simulation that required tens of gigabytes in memory and weeks of computation to terminate. The opposite extreme situation, with $R = 10$ m and $R_v = 2000$ m (but we obtained similar results values of R_v larger than few times R) resulted in the same results fo setting $R_v = \infty m$, de-facto disabling the virtual device generation. We concluded that what really matters for the system performance and cost is the ratio $R_v R$ and one between R_v and R (of course, the interested reader may experiment with the provided kit for reproducing the experiments, and explore the system behaviour with these or even more extreme settings). Thus, we decided to control independently R (spanning over a geometrically growing range) and $R_v R$ (spanning over a linear range, as discussed below).
- *Physical device count* N_r: the number of explorers in the scenario, equal to the number of physical devices in the scenario, hence $N_r = |H|$. The values for this parameter are selected by taking five logarithmic samples between two extreme (but still reasonable) values: five people (a severely undermanned team, caused for example, by a limited number of working devices), and 500 people (a rather impressive exploration force, considering a city of approximately 15,000 residents, that could be achieved if the application can run on smartphones and citizens are allowed to help in the mapping operations). To take $S \in \mathbb{N} : S > 1$ samples geometrically distributed between two values $v, V \in \mathbb{R}^+ : V > v$, we use the following function $f : \{0, 1, \ldots, S-1\} \to \mathbb{Z}$ given by $f(i) = \lfloor m \cdot (Mm)^{\frac{i}{S-1}} \rfloor$ ($N_r = 158$, the only tested value that is not a multiple of five results from the formula when $i = 3$).
- *Relative spawn range* $R_v R$: the ratio between R_v and R. As mentioned when discussing the set of values selected for R_v, the ratio $R_v R$ provides more insightful information than modifying R and R_v independently. Since initial testing showed that both values of $R_v R$ close to 0 and values much higher than 1 produced respective situations with a gigantic cost and poor performance and results akin to those of $R_v R = \infty$, we decided to use a linear range of six samples between 0.25 and 1.5, plus the special value ∞ that entirely disables the generation of virtual devices.

5.2.3. Toolchain and Reproducibility

We implemented the route-tracing program in the Protelis aggregate programming language [49], and we leveraged the Alchemist simulator [50] to model the environment and the devices' movements. Data for the road infrastructure have been extracted from OpenStreetMap (OSM) [51], as Alchemist provides native support for route computation and navigation within these maps.

All experiments are entirely reproducible, all data have been archived and made publicly available [44]. The code and the software infrastructure have been open-sourced, and they are freely available at a public repository (https://github.com/DanySK/experiment-2021-virtual-devices, accessed on 30 November 2021), which also contains further details and charts, omitted here for space reasons.

5.2.4. Repetitions

For each combination of the values of the free values listed in Table 4, we run 50 repetitions changing the random seed. Random seed changes influence the initial position of the real devices, the generated destinations to explore, the position of the PoI, and the scheduling order of the computation rounds. The results presented in this paper are averaged across these 50 runs.

5.3. Experiment Results and Discussion

The results in Figure 6 (mean distance error $\overline{\delta}$) and Figure 7 (isolated devices I) show that the system, augmented with virtual devices, can (i) provide more precise route estimates than the original version (identified with $R_v R = \infty$) and (ii) improve reachability. In more detail, the system achieves good performance across the board when $R_v R \leq 0.5$, indicating that the generation of virtual devices can indeed provide an increased system performance.

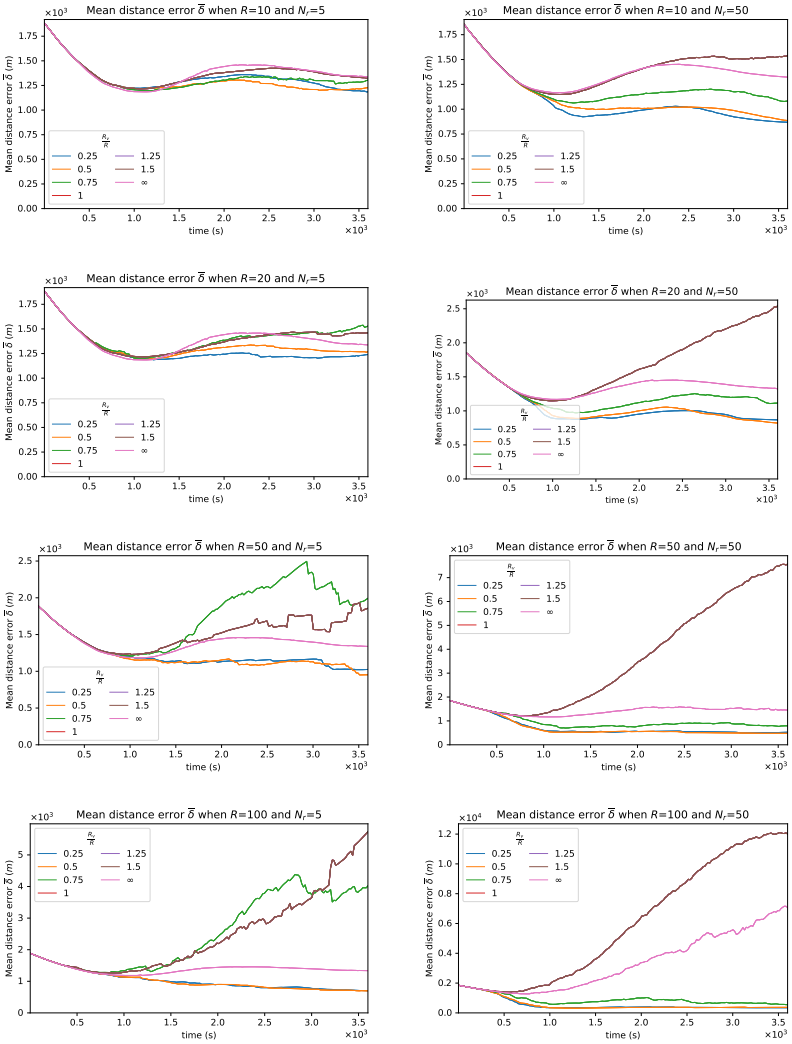

Figure 6. System performance measured as distance error for increasing communication range R (**top** to **bottom**) and different count of explorers N_r (5 in the **left column**, 50 in the **right column**). Implementations that generate virtual devices outperform the base implementation consistently, as far as virtual devices are generated densely enough. In most scenarios, setting $R_v = R2$ achieves a consistently good result. When R_v is larger than R, the system performance degrades very quickly, mostly due to insufficient coverage, which, with few explorers, also causes erratic behaviour.

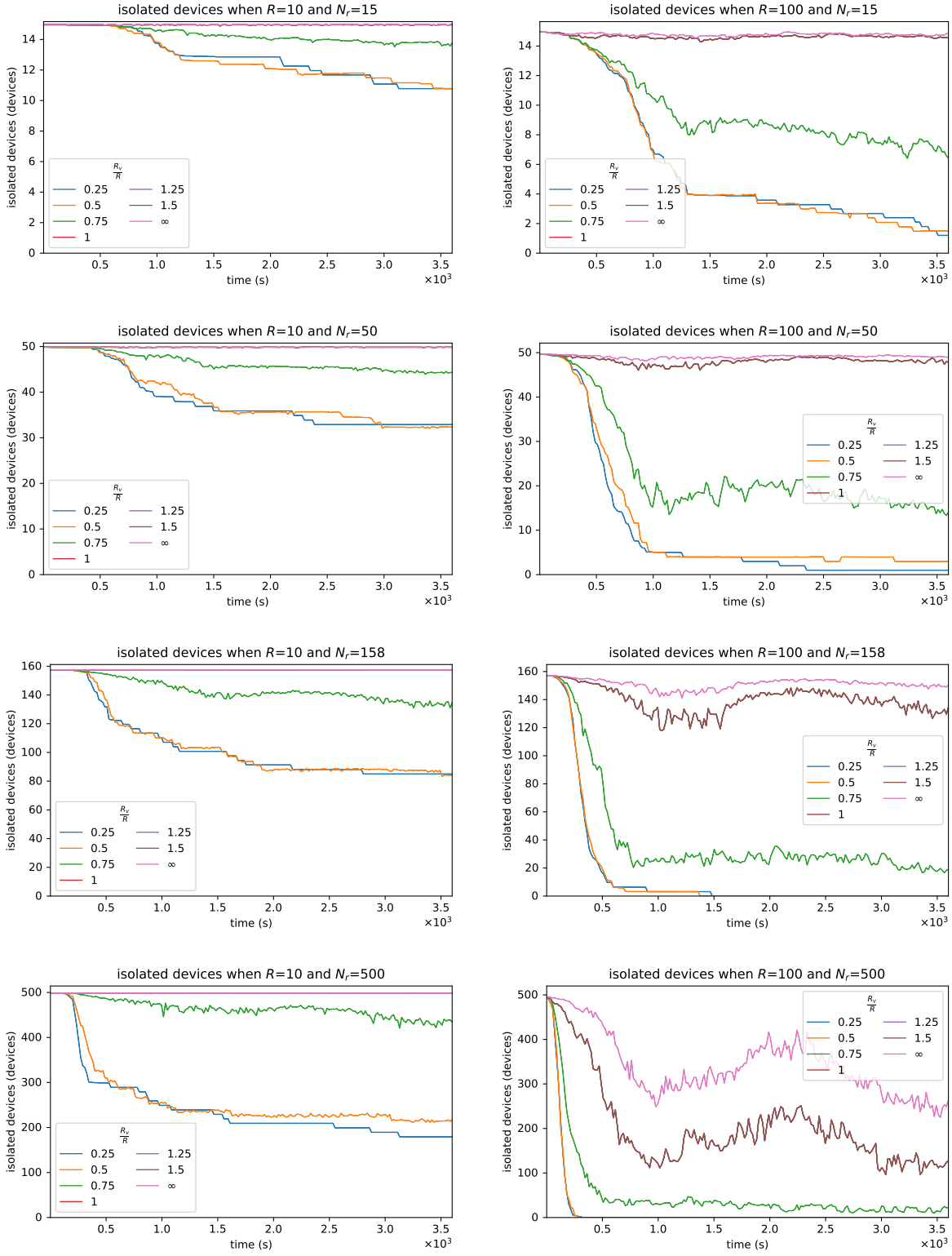

Figure 7. System performance measured as count of isolated devices I for communication range $R = 10$ m (**left column**) and communication range $R = 100$ m (**right column**), and increasing count of explorers N_r (**top** to **bottom**). As expected, systems that generate virtual devices more densely achieve better coverage. With many explorers, the benefits of a denser generation lower. Even with few explorers, there is little difference between $R_vR = 0.25$ and $R_vR = 0.5$, confirming the results seen in Figure 6.

From Figure 6, we see that the system has an initial stabilisation phase: initially, explorers can hardly communicate with each other, and with time, through the generation of virtual devices, the virtual communication network gets closer and closer to the shape of the actual communication routes. We can see that the larger is the count of explorers, the shorter is the time required for the system to stabilise (as more roads get explored more quickly). We note however that in some cases the system error, after a period of reduction, grows out of control. This phenomenon occurs in particular when $R_v R \geq 1$, and when $R_v R = 0.75$ and $R \geq 100$ m, and it is typically coupled with an erratic behaviour of the error. We believe that this effect is due to the logical network no longer mapping the underlying physical infrastructure correctly: virtual devices become too sparse and their connection range too large, the virtual network infrastructure that they build grows further and further different from the curvy structure of the Medieval roads that make the city centre. In some cases, a device placed in an unexplored area may lead to an improvement in the distance error, yet normally there is a high chance that its reification may cause the creation of a non-existing passage between two locations.

By observing Figure 7, we can further appreciate how a larger count of explorers helps to reach stability faster. Interestingly, even with reasonably short values for R (e.g., R = 100 m), the system shows a pretty high reachability (the lower the count of isolated devices, the higher the reachability). As expected, for any given value of R, the more virtual devices get generated, the higher is the reachability.

Of course, as shown in Figure 8 (cost in computational resources, using the count of virtual devices R_v as a proxy metric) and Figure 9 (communication cost, measured as the sum of messages received per round on all real devices $\sum_{j \in H} M_j$), it is not a free lunch. In particular, the system is very sensible (both performance-wise and cost-wise) to the way virtual devices are generated, in this case tuned via the parameter R_v (regulated along with the communication range R by tuning $R_v R$). Figure 8 shows that the count of generated virtual devices N_v grows sub-linearly with time, and approximately linearly with shorter ranges. Deploying more explores leads to more virtual devices generated more quickly in the initial phase of the simulation, and then to a flatter generation curve, as most of the map gets covered. Finer strategies to deploy devices may present further trade-offs, especially in case virtual devices can be removed dynamically.

Finally, we remark that the *augmented CDT approach* can be useful whenever there is the need to affect the behaviour of collective and self-organising systems. Example scenarios might include swarm robotics [13], WSNs [38] and crowd computing [21], as well as large-scale CPS ecosystems. For instance, a CDT can be developed to represent and control the state and motion of a fleet of drones; at this point, virtual devices may be spawned to *dynamically* regulate the overall distribution and direction of the fleet, using techniques like those presented in [13,38]. The idea is that such a soft control technique can help when (i) we cannot change the behaviour of the components of the CPS, (ii) to avoid the specification of an overly complex behaviour (e.g., whereby decisions are based on several non-local situations or to previsions about future events), or (iii) to apply perturbations or simulate conditions that do not exist in the real world (e.g., to test how the CPT reacts to crowding situations). In other words, the approach provides an *additional* mechanism for self-organisation design (steering). Whether it is the most suitable approach depends on the problem at hand, but some general observations can be made. Indeed, the approach appears most applicable when the application depends on spatial structures (e.g., to propagate information, to move in space, or to make decisions based on spatial arrangements). More generally, it is a matter of decentralisation and partial observability: in our and most approaches to self-organisation engineering, the behaviour of individuals is organised in terms of local environment information (e.g., sensed data) and messages (e.g., neighbour data); augmentations through virtual devices, then, allow changes in the local context of the individual devices, hence affecting the overall emergence of global results. The main issue, which may affect applicability, lies in the ability to determine reasonable local contexts for the virtual devices (or, similarly, defining their ad-hoc interactive behaviour).

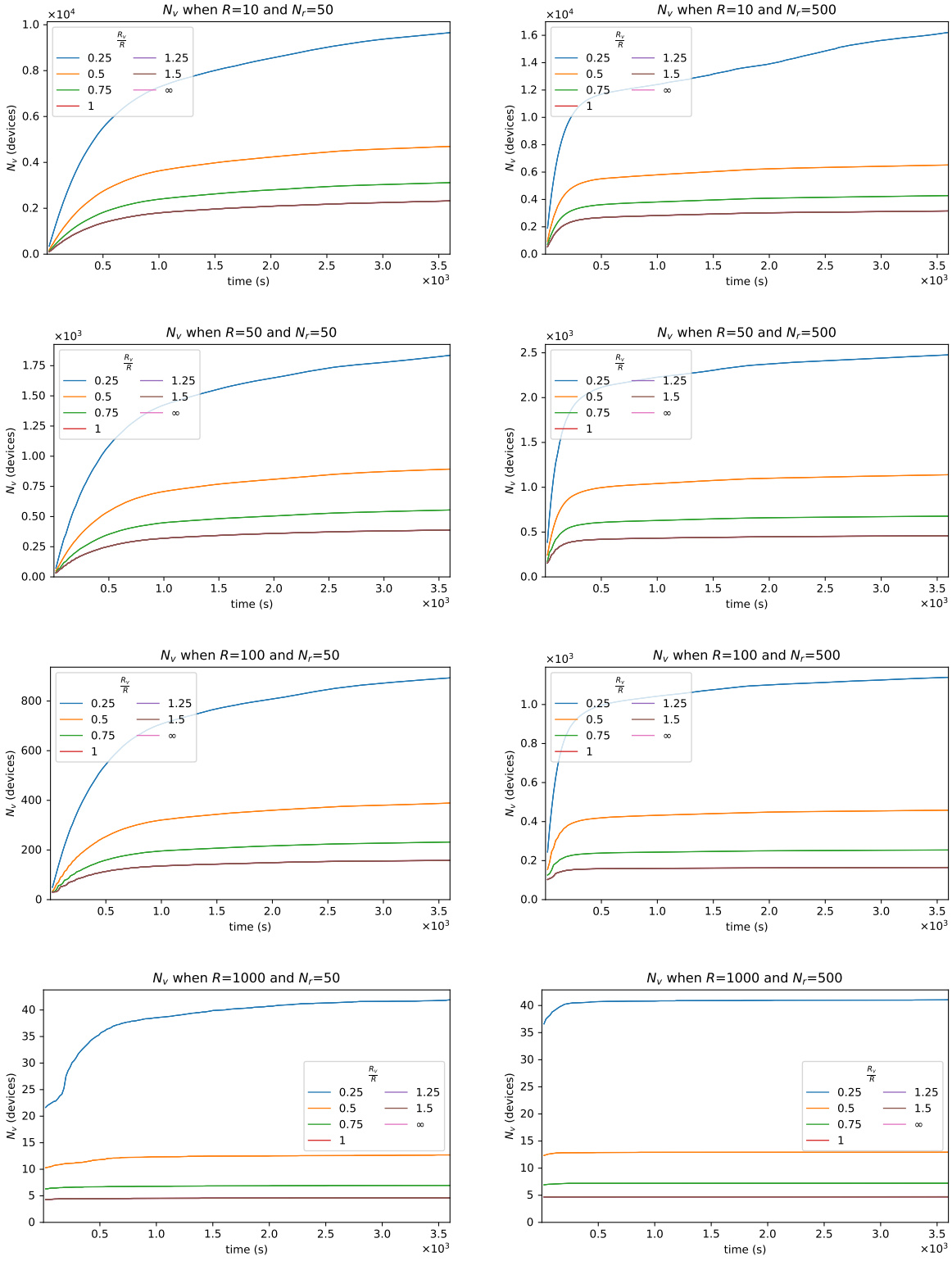

Figure 8. System cost in computational resources, using the count of virtual devices N_v as a proxy metric, for 50 explorers (**left column**) and 500 explorers (**right column**), and increasing logical communication range R (**top** to **bottom**). The system cost grows more than linearly with shorter R_v values. Considering the performance seen in Figures 6 and 7, it is clear that the value of N_v needs to be selected carefully, as values too small do not improve performance substantially, but increase the computation cost considerably.

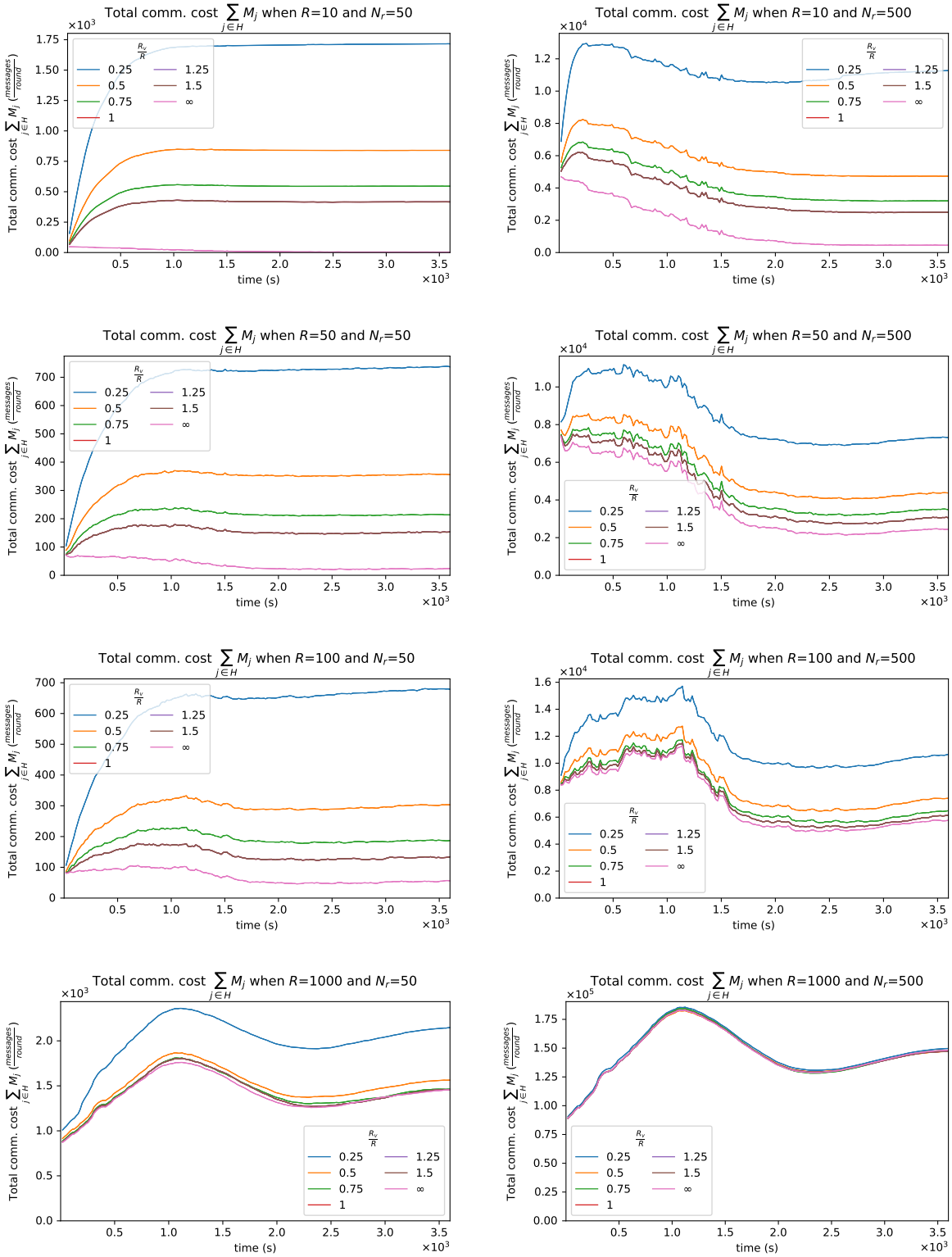

Figure 9. System communication cost, measured as the sum of messages received per round on all real devices $\sum_{j \in H} M_j$, for 50 explorers (**left column**) and 500 explorers (**right column**), and increasing logical communication range R (**top** to **bottom**). Similarly to the results depicted in Figure 8, the cost grows more than linearly with $R_v R$ The cost is lower for intermediate values of R, suggesting that a trade-off in the logical communication range could be found between performance and use of network resources.

6. Conclusions and Future Work

In this manuscript, we present a conceptual and modelling framework supporting the modular abstraction and design of self-organising Cyber-Physical Systems. The idea is to use multiple notions of logical devices, based on two typical constructs found in the literature—virtual devices and digital twins, which we distinguish on the basis of an identity relationship between logical and physical devices—and the adoption of a collective viewpoint. Indeed, the novel concept of the *collective digital twin* is introduced, referring to a collection of digital twins that can be considered as a whole. This notion can further be augmented with virtual devices in order to define a richer logical system, which we call an *augmented collective digital twin*, which can be exploited to provide an augmented reality to a collection of actual, physical twins. Additionally, we perform experiments to prove that this is not just an abstraction exercise, but an enabler mechanism to improve and steer certain kinds of self-organisation processes.

Arguably, the ideas presented in this paper can pave the path for more research and investigations.

- *Logical abstractions for engineering of self-organising Cyber-Physical Systems.* This work discusses how purely virtual devices can be used to steer existing self-organising applications. We believe that defining logical constructs and formalising the relationship between these and physical counterparts can provide benefits for the engineering of Cyber-Physical Systems. In particular, collective-level notions such as CDTs (and strategies for implementing the corresponding digital threads) are still to be explored in depth.
- *Self-adaptive deployment of digital thread components.* The location of the software components responsible for implementing the digital thread can be optimised w.r.t. non-functional metrics such as bandwidth and energy usage. Moreover, for highly dynamic systems subject to changes affecting connectivity and available computing infrastructure, (re-)deployment decisions may be taken at runtime.
- *Self-adaptive and opportunistic management of virtual devices.* In the case study of Section 5, the system can dynamically spawn virtual nodes in order to support connectivity and self-organised exploration in a dynamic environment. In general, the trade-off between the functional benefit (current and future) and the virtualisation overhead can be analysed in order to take the best decisions for the situation at hand. This could be supported by an autonomic platform where opportunities for virtualisation, execution cost, and user preferences are altogether considered for the efficient spawning or removal of virtual devices in the system. This could be seen as a form of self-improving integration [15] of virtual and physical devices.

Author Contributions: Conceptualization, R.C. and M.V.; data curation, D.P.; formal analysis, D.P.; funding acquisition, M.V.; investigation, R.C. and D.P.; methodology, R.C. and D.P.; project administration, R.C.; resources, M.V.; software, D.P.; supervision, D.W.; validation, R.C. and D.P.; visualization, R.C. and D.P.; writing—original draft, R.C. and D.P.; writing—review and editing, R.C., D.P., D.W. and M.V. All authors have read and agreed to the published version of the manuscript.

Funding: This work has been supported by the MIUR PRIN Project N. 2017KRC7KT "Fluidware".

Data Availability Statement: Data can be found at the following public archived repository: https://zenodo.org/record/5809103, accessed on 29 December 2021.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Bures, T.; Weyns, D.; Schmer, B.; Tovar, E.; Boden, E.; Gabor, T.; Gerostathopoulos, I.; Gupta, P.; Kang, E.; Knauss, A. Software Engineering for Smart Cyber-Physical Systems: Challenges and Promising Solutions. *SIGSOFT Softw. Eng. Notes* **2017**, *42*, 19–24. [CrossRef]
2. Weyns, D.; Andersson, J.; Caporuscio, M.; Flammini, F.; Kerren, A.; Lowe, W. A Research Agenda for Smarter Cyber Physical System. *J. Integr. Des. Process. Sci.* **2021**, 1–21. [CrossRef]
3. Kephart, J.O.; Chess, D.M. The Vision of Autonomic Computing. *IEEE Comput.* **2003**, *36*, 41–50. [CrossRef]

4. Rasheed, A.; San, O.; Kvamsdal, T. Digital Twin: Values, Challenges and Enablers From a Modeling Perspective. *IEEE Access* **2020**, *8*, 21980–22012. [CrossRef]
5. Singh, V.; Willcox, K.E. Engineering Design with Digital Thread. *AIAA J.* **2018**, *56*, 4515–4528. [CrossRef]
6. Bose, R.; Helal, A.; Sivakumar, V.; Lim, S. Virtual Sensors for Service Oriented Intelligent Environments. In Proceedings of the 3rd IASTED International Conference: Advances in Computer Science and Technology, Phuket, Thailand, 2–4 April 2007; pp. 165–170.
7. Chatterjee, S.; Misra, S. Optimal composition of a virtual sensor for efficient virtualization within sensor-cloud. In Proceedings of the 2015 IEEE International Conference on Communications, London, UK, 8–12 June 2015; pp. 448–453. [CrossRef]
8. Khansari, M.E.; Sharifian, S.; Motamedi, S.A. Virtual sensor as a service: A new multicriteria QoS-aware cloud service composition for IoT applications. *J. Supercomput.* **2018**, *74*, 5485–5512. [CrossRef]
9. Brown, M.; Gilbert, S.; Lynch, N.A.; Newport, C.C.; Nolte, T.; Spindel, M. The virtual node layer: A programming abstraction for wireless sensor networks. *SIGBED Rev.* **2007**, *4*, 7–12. [CrossRef]
10. Gershenson, C. Guiding the Self-Organization of Cyber-Physical Systems. *Front. Robot. AI* **2020**, *7*, 41. [CrossRef]
11. Casadei, R.; Pianini, D.; Placuzzi, A.; Viroli, M.; Weyns, D. Pulverization in Cyber-Physical Systems: Engineering the Self-Organizing Logic Separated from Deployment. *Future Internet* **2020**, *12*, 203. [CrossRef]
12. Aguzzi, G.; Casadei, R.; Pianini, D.; Salvaneschi, G.; Viroli, M. Towards Pulverised Architectures for Collective Adaptive Systems through Multi-Tier Programming. In Proceedings of the 2021 IEEE International Conference on Autonomic Computing and Self-Organizing Systems Companion (ACSOS-C), Washington, DC, USA, 27 September–1 October 2021; pp. 99–104. [CrossRef]
13. Han, J.; Li, M.; Guo, L. Soft Control on Collective Behavior of a Group of Autonomous Agents By a Shill Agent. *J. Syst. Sci. Complex.* **2006**, *19*, 54–62. [CrossRef]
14. Li, W.; Shen, W. Swarm behavior control of mobile multi-robots with wireless sensor networks. *J. Netw. Comput. Appl.* **2011**, *34*, 1398–1407. [CrossRef]
15. Bellman, K.L.; Botev, J.; Diaconescu, A.; Esterle, L.; Gruhl, C.; Landauer, C.; Lewis, P.R.; Nelson, P.R.; Pournaras, E.; Stein, A. Self-improving system integration: Mastering continuous change. *Future Gener. Comput. Syst.* **2021**, *117*, 29–46. [CrossRef]
16. Casadei, R.; Placuzzi, A.; Viroli, M.; Weyns, D. Augmented Collective Digital Twins for Self-Organising Cyber-Physical Systems. In Proceedings of the 2021 IEEE International Conference on Autonomic Computing and Self-Organizing Systems Companion (ACSOS-C), Washington, DC, USA, 27 September–1 October 2021; pp. 160–165. [CrossRef]
17. Fuller, A.; Fan, Z.; Day, C.; Barlow, C. Digital Twin: Enabling Technologies, Challenges and Open Research. *IEEE Access* **2020**, *8*, 108952–108971. [CrossRef]
18. van der Valk, H.; Haße, H.; Möller, F.; Arbter, M.; Henning, J.; Otto, B. A Taxonomy of Digital Twins. In Proceedings of the 26th Americas Conference on Information Systems, Salt Lake, UT, USA, 15–17 August 2020.
19. Casadei, R.; Pianini, D.; Viroli, M.; Natali, A. Self-organising Coordination Regions: A Pattern for Edge Computing. In *LNCS*; Springer International Publishing: Berlin/Heidelberg, Germany, 2019; pp. 182–199. [CrossRef]
20. Almobaideen, W.; Qatawneh, M.; AbuAlghanam, O. Virtual node schedule for supporting QoS in wireless sensor network. In Proceedings of the 2019 IEEE Jordan International Joint Conference on Electrical Engineering and Information Technology (JEEIT), Amman, Jordan, 9–11 April 2019; pp. 281–285.
21. Beal, J.; Pianini, D.; Viroli, M. Aggregate Programming for the Internet of Things. *IEEE Comput.* **2015**, *48*, 22–30. [CrossRef]
22. Viroli, M.; Beal, J.; Damiani, F.; Audrito, G.; Casadei, R.; Pianini, D. From distributed coordination to field calculus and aggregate computing. *J. Log. Algebr. Methods Progr.* **2019**, *109*, 100486. [CrossRef]
23. Bosch, J.; Eklund, U. Eternal Embedded Software: Towards Innovation Experiment Systems. In Proceedings of the Leveraging Applications of Formal Methods, Verification and Validation. Technologies for Mastering Change—5th International Symposium, ISoLA 2012, Heraklion, Greece, 15–18 October 2012; Proceedings, Part I; Margaria, T., Steffen, B., Eds.; Springer: Berlin/Heidelberg, Germany, 2012; Volume 7609, pp. 19–31. [CrossRef]
24. Fiorentino, G.; Forte, A.; Pagano, E.; Sabetta, F.; Baggio, C.; Lavorato, D.; Nuti, C.; Santini, S. Damage patterns in the town of Amatrice after August 24th 2016 Central Italy earthquakes. *Bull. Earthq. Eng.* **2017**, *16*, 1399–1423. [CrossRef]
25. Azzaro, R.; Tertulliani, A.; Bernardini, F.; Camassi, R.; Mese, S.D.; Ercolani, E.; Graziani, L.; Locati, M.; Maramai, A.; Pessina, V.; et al. The 24 August 2016 Amatrice earthquake: Macroseismic survey in the damage area and EMS intensity assessment. *Ann. Geophys.* **2016**, *59*. [CrossRef]
26. Puzrin, A.M.; Faug, T.; Einav, I. The mechanism of delayed release in earthquake-induced avalanches. *Proc. R. Soc. Math. Phys. Eng. Sci.* **2019**, *475*, 20190092. [CrossRef] [PubMed]
27. Huang, M.H.; Fielding, E.J.; Liang, C.; Milillo, P.; Bekaert, D.; Dreger, D.; Salzer, J. Coseismic deformation and triggered landslides of the 2016 M_w 6.2 Amatrice earthquake in Italy. *Geophys. Res. Lett.* **2017**, *44*, 1266–1274. [CrossRef]
28. Braun, T.; Frigo, B.; Chiaia, B.; Bartelt, P.; Famiani, D.; Wassermann, J. Seismic signature of the deadly snow avalanche of January 18, 2017, at Rigopiano (Italy). *Sci. Rep.* **2020**, *10*, 18563. [CrossRef]
29. Valcarce, A.; Rasheed, T.; Gomez, K.M.; Sithamparanathan, K.; Reynaud, L.; Hermenier, R.; Munari, A.; Mohorcic, M.; Smolnikar, M.; Bucaille, I. Airborne Base Stations for Emergency and Temporary Events. In Proceedings of the Personal Satellite Services—5th International ICST Conference, PSATS 2013, Toulouse, France, 27–28 June 2013; Revised Selected Papers; Dhaou, R., Beylot, A., Montpetit, M., Lucani, D.E., Mucchi, L., Eds.; Springer: Berlin/Heidelberg, Germany, 2013; Volume 123, pp. 13–25. [CrossRef]

30. Audrito, G.; Casadei, R.; Damiani, F.; Viroli, M. Compositional Blocks for Optimal Self-Healing Gradients. In Proceedings of the 2017 IEEE 11th International Conference on Self-Adaptive and Self-Organizing Systems (SASO), Tucson, AZ, USA, 18–22 September 2017.
31. Pianini, D.; Casadei, R.; Viroli, M.; Mariani, S.; Zambonelli, F. Time-Fluid Field-Based Coordination through Programmable Distributed Schedulers. *Log. Methods Comput. Sci.* **2021**, *17*, 13:1–13:48. [CrossRef]
32. Arellanes, D. Self-Organizing Software Models for the Internet of Things: Complex Software Structures that Emerge without a Central Controller. *IEEE Syst. Man Cybern. Mag.* **2021**, *7*, 4–9. [CrossRef]
33. Berger, S.; Häckel, B.; Häfner, L. Organizing Self-Organizing Systems: A Terminology, Taxonomy, and Reference Model for Entities in Cyber-Physical Production Systems. *Inf. Syst. Front.* **2021**, *23*, 391–414. [CrossRef]
34. Ballouli, R.E.; Bensalem, S.; Bozga, M.; Sifakis, J. Four Exercises in Programming Dynamic Reconfigurable Systems: Methodology and Solution in DR-BIP. In Proceedings of the Leveraging Applications of Formal Methods, Verification and Validation, Distributed Systems—8th International Symposium, ISoLA 2018, Limassol, Cyprus, 5–9 November 2018; Proceedings, Part III; Margaria, T., Steffen, B., Eds.; Springer: Berlin/Heidelberg, Germany, 2018; Volume 11246, pp. 304–320. [CrossRef]
35. Nicola, R.D.; Maggi, A.; Sifakis, J. The DReAM framework for dynamic reconfigurable architecture modelling: Theory and applications. *Int. J. Softw. Tools Technol. Transf.* **2020**, *22*, 437–455. [CrossRef]
36. Serugendo, G.D.M. Spatial Edge Services—From Coordination Model to Actual Applications. In Proceedings of the Coordination Models and Languages—19th IFIP WG 6.1 International Conference, COORDINATION 2017, Held as Part of the 12th International Federated Conference on Distributed Computing Techniques, DisCoTec 2017, Neuchâtel, Switzerland, 19–22 June 2017; Jacquet, J., Massink, M., Eds.; Springer: Berlin/Heidelberg, Germany, 2017; Volume 10319, pp. 3–17. [CrossRef]
37. Diaconescu, A.; Felice, L.J.D.; Mellodge, P. Exogenous coordination in multi-scale systems: How information flows and timing affect system properties. *Future Gener. Comput. Syst.* **2021**, *114*, 403–426. [CrossRef]
38. Sartoretti, G. Leader-based versus soft control of multi-agent swarms. *Artif. Life Robot.* **2016**, *21*, 302–307. [CrossRef]
39. Kim, H.; Valentini, G.; Hanson, J.; Walker, S.I. Informational architecture across non-living and living collectives. *Theory Biosci.* **2021**, *140*, 325–341. [CrossRef] [PubMed]
40. Kroher, C.; Schmid, K.; Paasche, S.; Sauer, C. Combining Central Control with Collective Adaptive Systems. In Proceedings of the 2021 IEEE International Conference on Autonomic Computing and Self-Organizing Systems Companion (ACSOS-C), Washington, DC, USA, 27 September–1 October 2021; pp. 56–61. [CrossRef]
41. Genter, K.; Zhang, S.; Stone, P. Determining Placements of Influencing Agents in a Flock. In Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems, Istanbul, Turkey, 4–8 May 2015; pp. 247–255.
42. Leonard, N.; Fiorelli, E. Virtual leaders, artificial potentials and coordinated control of groups. In Proceedings of the 40th IEEE Conference on Decision and Control (Cat. No.01CH37228), Orlando, FL, USA, 4–7 December 2001; Volume 3, pp. 2968–2973. [CrossRef]
43. Zheng, Z.; Li, J.; Li, J.; Tan, Y. Avoiding decoys in multiple targets searching problems using swarm robotics. In Proceedings of the IEEE Congress on Evolutionary Computation, CEC 2014, Beijing, China, 6–11 July 2014; pp. 784–791. [CrossRef]
44. Pianini, D. Digital Twins, Virtual Devices, and Augmentations for Self-Organising Cyber-Physical Collectives. In Zenodo, DanySK/Experiment-2021-Virtual-Devices: 1.1.0, December 2021. Available online: https://zenodo.org/record/5809103 (accessed on 29 December 2021). [CrossRef]
45. Gilbert, S.; Lynch, N.A. Perspectives on the CAP Theorem. *Computer* **2012**, *45*, 30–36. [CrossRef]
46. Conti, M.; Kumar, M. Opportunities in Opportunistic Computing. *Computer* **2010**, *43*, 42–50. [CrossRef]
47. Weyns, D. *Introduction to Self-Adaptive Systems: A Contemporary Software Engineering Perspective*; Wiley: Hoboken, NJ, USA, 2020.
48. Browning, R.C.; Baker, E.A.; Herron, J.A.; Kram, R. Effects of obesity and sex on the energetic cost and preferred speed of walking. *J. Appl. Physiol.* **2006**, *100*, 390–398. [CrossRef]
49. Pianini, D.; Viroli, M.; Beal, J. *Protelis: Practical Aggregate Programming*; ACM: New York, NY, USA, 2015; pp. 1846–1853. [CrossRef]
50. Pianini, D.; Montagna, S.; Viroli, M. Chemical-oriented simulation of computational systems with ALCHEMIST. *J. Simul.* **2013**, *7*, 202–215. [CrossRef]
51. Haklay, M.; Weber, P. OpenStreetMap: User-Generated Street Maps. *IEEE Pervasive Comput.* **2008**, *7*, 12–18. [CrossRef]