SD-CPS: Taming the Challenges of Cyber-Physical Systems with a Software-Defined Approach

Pradeeban Kathiravelu
INESC-ID / Instituto Superior Técnico, Universidade de Lisboa
Université catholique de Louvain
Place Sainte-Barbe 2, 1348 Ottignies-Louvain-la-Neuve, Belgium
Tel: (+32) 10 47 87 18
Fax: (+32) 10 45 03 45
pradeeban.kathiravelu@tecnico.ulisboa.pt

Luís Veiga
INESC-ID / Instituto Superior Técnico
Universidade de Lisboa
Rua Alves Redol 9, Lisboa 1000-029, Portugal
Tel: (+351) 21 310 0300
Fax: (+351) 21 314 5843
luis.veiga@inesc-id.pt

Abstract—**Cyber-Physical Systems (CPS) revolutionize various application domains with integration and interoperability of networking, computing systems, and mechanical devices. Due to its scale and variety, CPS faces a number of challenges and opens up a few research questions in terms of management, fault-tolerance, and scalability. We propose a software-defined approach inspired by Software-Defined Networking (SDN), to address the challenges for a wider CPS adoption. We thus design a middleware architecture for the correct and resilient operation of CPS, to manage and coordinate the interacting devices centrally in the cyberspace whilst not sacrificing the functionality and performance benefits inherent to a distributed execution.**

I. Introduction

While the Internet of Things (IoT) [1] motivates for a scenario where there are many smart devices that are all connected together and are accessible pervasively in the Internet, reality is still far from this. We do have several networks of things, where the smart devices (or the "things") are interconnected to form network of devices, or connected to an existing enterprise network. However, one needs not to have *the Internet of Things* literally, as it is not necessary to connect everything to the Internet, the single unified network of networks. It is not only unnecessary, but also counter-intuitive to have everything connected and open beyond what is necessary, due to security and privacy reasons.

Cyber-Physical Systems (CPS) fix the shortcomings and limitations in the definition of IoT and similar terms, in clearly defining the common larger ground of theories and practice where the physical/mechanical systems intersect and deeply intertwine with the cyber/computer systems [2]. While sharing the core architecture with IoT [3], CPS is defined as a pure interdisciplinary mechanism, with applications ranging from smart homes [4] to smart cities [5]. Though CPS is a term that is coined relatively in recent times, there have been research and implementation efforts on the topic even before the inception of the term [6].

Due to the scale and variety in its implementation and devices, CPS faces several challenges in design and practice [7], in terms of: (i) unpredictability of the execution environments [8], (ii) communication and coordination within the system [9], (iii) security, distributed fault-tolerance, and recovery upon system and network failures [10], (iv) decision making in the large-scale geo-distributed execution environments, (v) modelling and designing the systems [11], and (vi) management and orchestration of the intelligent agents.

The challenges are imposed from both the core domains of CPS: physical space consisting of networking, distributed systems, and the physical systems involved, and cyberspace. Further challenges manifest due to the co-existence and interdependencies of the cyberspace and physical devices in CPS. A unified approach is necessary to address the challenges that prevent or hinder the seamless adoption, applicability, and reusability of the CPS principles and constructs pervasively.

Software-Defined Networking (SDN) offers programmability and management capabilities, among many other improvements, to networks by separating the control layer as a unified controller, away from the distributed data forwarding elements of the network. There have been recent researches on leveraging SDN in the implementation of CPS. SDN has been proposed to improve the resilience of multinetworks in CPS [12]. SDN has been leveraged to secure the CPS networks through SDN-assisted emulations [13] and improve the resilience [14] of CPS.

We propose to tackle the current and foreseen future challenges of CPS through a middleware architecture following a software-defined approach. We call the proposed approach for CPS, "Software-Defined Cyber-Physical Systems (SD-CPS)". Designed as a middleware framework inspired by the logically centralized control offered by the SDN controllers, *SD-CPS* aims to tackle the core challenges of CPS as an architectural enhancement.

Through its novel architecture for CPS, *SD-CPS* offers a set of enhancements to building and executing CPS. The major contributions of *SD-CPS* are: (i) A dual execution environment of physical and virtual/cyber execution spaces. This is aimed to mitigate the challenges faced by the unpredictable nature of the physical execution environment. (ii) Enhanced communication and coordination of autonomous agents, through virtually separated control and data flows, offering a clear logical separation of devices from their execution. (iii) Resilience for critical flows with a differentiated quality of service (QoS) with end-to-end delivery guarantees, and (iv) Enhanced CPS modelling.

In the upcoming sections, we further discuss the challenges faced in developing and executing CPS and how *SD-CPS* addresses them. Section II introduces how SDN can be adopted for CPS, and illustrates an architecture for the approach. Section III elaborates how *SD-CPS* tackles the design and operational challenges of CPS. Section IV discusses the prototype implementation and presents a feasibility assessment. We continue to discuss the related work on the CPS and software-defined systems in Section V. Finally, Section VI concludes the paper with the current state of the research and future research directions.

II. Solution Architecture

SD-CPS tackles the major challenges of CPS by adopting SDN and a software-defined approach for CPS. In this section, we will discuss the design and architecture of *SD-CPS* in detail.

A. SD-CPS *Approach*

Current software-defined approaches can broadly be categorized into: (i) approaches that extend or use SDN and SDN controllers, and (ii) approaches that follow a similar architecture or motivation of SDN typically employing logically centralized approach to control, and distributed approach to data - but without addressing networking directly. *SD-CPS* employs a hybrid approach. It leverages the SDN when SDN is already a part of the deployment architecture of the CPS. Maintaining backward compatibility with existing system such as legacy switches, *SD-CPS* uses a software-defined approach to data and control, when SDN is absent in the existing system. Thus *SD-CPS* does not make SDN a prerequisite, to ensure a wider adoption.

The core of *SD-CPS* is a controller deployment, that controls the "cyber" of the CPS and centrally orchestrates the CPS elements. *SD-CPS* controller typically consists of a deployment of multiple SDN controllers and further software components to manage the CPS. For the ease of discussion, we assume SDN controller to be the core of the *SD-CPS* controller, though *SD-CPS* controller can be implemented without the SDN controller or protocol implementations. The control plane communicates with the underlying network through implementations of SDN southbound protocols such as OpenFlow, while communicating with the devices that are non-compliant with OpenFlow through a similar approach inspired by OpenFlow. Various other southbound protocols are incorporated into *SD-CPS* controller to communicate with the devices, to support the systems that are not SDN-native. This ensures that while *SD-CPS* has SDN at its core, it is not limited to software-defined networks with SDN switches that are still far from widespread in IoT and CPS settings (outside data centers). *SD-CPS* devises its APIs, adapting that of SDN [15] for the extended distributed controller deployment for CPS. Figure 1 depicts the *SD-CPS* controller in the larger *SD-CPS* ecosystem.

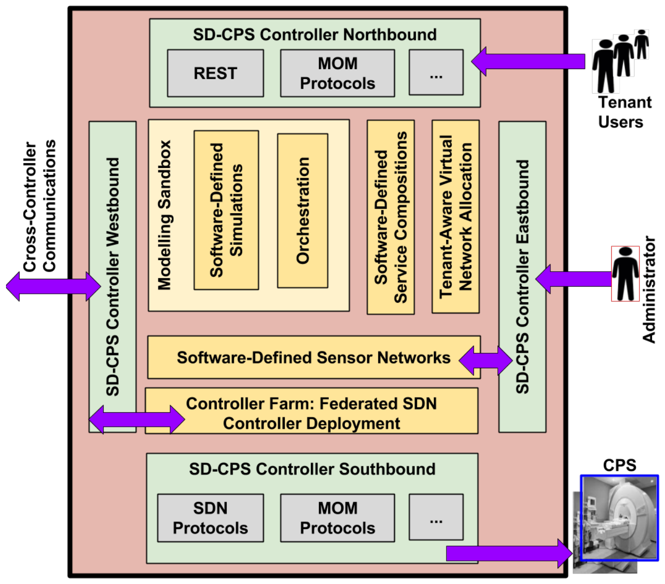
SD-CPS Controller Northbound
REST
MOM Protocols
...
Tenant Users
Cross-Controller Communications
SD-CPS Controller Westbound
Modelling Sandbox
Software-Defined Simulations
Orchestration
Software-Defined Service Compositions
Tenant-Aware Virtual Network Allocation
SD-CPS Controller Eastbound
Administrator
Software-Defined Sensor Networks
Controller Farm: Federated SDN Controller Deployment
SD-CPS Controller Southbound
SDN Protocols
MOM Protocols
...
CPS

Fig. 1. Controller of the "Cyber" of *SD-CPS*

Developed in high-level languages, SDN controller mediates the management and control of the networks through its APIs. *SD-CPS* controller is an extension of an SDN controller with various APIs, exposing its functionalities to its users, controlling and coordinating multiple devices and networks. The APIs consists of the implementation of various integration protocols, and connection points. **The Controller Farm** refers to a federated deployment of multiple clusters of controllers orchestrating networks in a wide area. The controllers communicate to each other to collaborate in a controlled manner, by passing messages, leveraging the public or dedicated/private connectivity between the domains. **The Software-Defined Sensor Networks** are built on top of the controller farm, representing each mobile terminal or a smart device as a sensor or an actuator.

The execution space of controller is virtually isolated to be owned by the multiple tenants, each managing the virtual network space allocated to the tenant. This allows a virtual network allocation to each tenant inside the controller, hence partitioning the rules and policies per tenant, minimizing the complexity of routing tables. The executions are passed between the components inside the controller as well as between the controllers as messages. The execution logic is passed to the data, rather than sending data remotely to the execution node. This minimizes the communication delays and bandwidth overheads. To ensure this data locality-aware execution, each middlebox action is designed as a web service. Thus, the network functions are virtualized into service compositions.

A **Modelling Sandbox** is developed inside the controller. Networks are modelled as graphs inside this sandbox. The **Orchestration** module either passes the execution to either simulate or execute in the devices according to the state of the execution, as specified by the tenant. While the application logic including middlebox actions are deployed in the controller space, the execution is either simulated or executed as services compositions. Due to such separation of execution space from the data plane, these actions are defined as (i) **Software-Defined Simulations** and (ii) **Software-Defined Service Compositions**, respectively. The colocation of the simulation and service composition bundles inside the controller ensures easy incremental migration without repeated implementation and manual deployment efforts.

SD-CPS controller can be deployed as a cluster, with a logically centralized view. The clustering of *SD-CPS* controller leverages the clustering mechanism of the underlying SDN controller. The *SD-CPS* controller operates as an orchestrator for the CPS cyberspace rather than just an SDN controller with extensions.

B. SD-CPS *Controller APIs*

SD-CPS controller is invoked by its users through its APIs: northbound, eastbound, southbound, and westbound. The various users that communicate with the controller through the applications or firmware hosted on their smart devices, or mobile terminals are called the tenant users of *SD-CPS*. The physical devices themselves are controlled by the controller through its southbound. The administrators manage the *SD-CPS* ecosystem consisting of the controller at its core.

The Northbound API consists of the typical SDN northbound protocols including REST and Message-Oriented Middleware (MOM) [16] protocols such as Advanced Message Queuing Protocol (AMQP) [17] or MQTT(formerly, Message Queue Telemetry Transport) [18] for the tenant processes to interact with the controller. As MOM protocols are long researched for use with networks of wireless sensors and actuators [19], [20], extending SDN with MOM increases its applicability, in addition to scalability.

The northbound API allows management of smart devices in a tenant-aware manner respecting the tenant intents (or preferences) and system policies as defined from the application layer. It also lets the tenants configure their user spaces in the controller based on the application layer inputs, or through their devices. Hence, the northbound ensures user involvement and human interaction in CPS through direct tenant input or by collecting operational data from the devices.

The Southbound API consists of implementations of SDN southbound protocols such as OpenFlow [21] and additional light-weight protocols such as MOM protocols for the communication with the physical devices. Based on the defined policies, rules, and the tenant inputs from northbound, controller defines the actions. The actions are propagated back to the data plane consisting of the SDN switches or network fabric and the system of physical devices through the southbound. Thus, the southbound API handles the communication, coordination, and integration of the network data plane consisting of the CPS devices with the control plane.

The Westbound API enables inter-control communication among the controllers in *SD-CPS*, as well as inter-domain communications across multiple *SD-CPS* controller deployments, through their westbound. The controller farm provides a federated deployment of controllers in each *SD-CPS* control plane consisting of multiple SDN controllers that have protected access to the internal storage of each other. Hence multi-domain networks can be controlled in a centralized, yet multi-tenanted manner, i.e. without sharing the single (logically centralized) controller. This offers multi-tenancy and tenant isolation in the CPS networks which typically must share the network for the data and control flows unlike the traditional data center networks that can have dedicated bandwidth for each. Thus, the controller farm and the westbound API facilitate the execution and interoperability of various entities in *SD-CPS*, coordinated by SDN controllers, legacy network controllers, and the other controllers of physical devices and cyberspace. The cross-controller communication supported by messaging between the controllers enable a clustered deployment in a wide area network. This approach enables dynamic subscription-based communication channels rather than predefined topologies or static links dedicated for controller communication.

The Eastbound API is leveraged by the administrators to configure and manage the controller deployment. By offering a restricted access to the tenant space in the internal data store of the controller, sensors and actuators in a sensor network can efficiently collaborate and communicate with one another and with the controller. This produces a Software-Defined Sensor Network, that can control sensor networks and heterogeneous smart devices, extending a controller farm of SDN controllers with lightweight southbound MOM protocols. Equipped with (i) the global view of the system from the SDN controller, and (ii) scalability of the control plane from the controller farm, the Software-Defined Sensor Network is capable of forwarding/routing decisions based on the tenant preferences and system policies from the cyberspace application layer.

C. SD-CPS *Core Enablers*

SD-CPS architecture adheres to the functions and attributes of CPS while not sacrificing the capabilities of CPS and to maintain backward compatibility with existing CPS architectures. This can be articulated in a **5C level** architecture [22] in a bottom-up approach: (i) Software-Defined Sensor Networks representing the **Smart Connection Level** stays the core bottom-most element in the 5C level architecture which is responsible for plug & play of sensor networks. *SD-CPS* further leverages the controller farm to offer a teather-free communication for the network. (ii) **Data-to-Information Conversion Level** handles multi-dimensional data analytics. *SD-CPS* Software-Defined Service Composition visualizes the analytics as microservices and executes the multi-dimensional data correlation as service compositions. (iii) **Cyber Level** supported by the *SD-CPS* modelling sandbox offers a twin representation for the physical devices and their cyber counterpart with identification and memory across time, offering data mining capabilities in the cyber representation for decision making. (iv) **Cognition Level** targets the human aspects with modelling, simulation, and visual aspects of CPS. The

Software-Defined Simulations enable integrated visualization and synthesis for the Cognition Level. and (v) **Configuration Level** offers self-configuration and adjustment for resilience, optimization, and healing capabilities as the top-most layer.

*1) **Software-Defined Service Composition**:* CPS networks typically share the data and control flows over the same network bandwidth, despite the heterogeneity in data flow of various CPS and devices. *SD-CPS* virtually isolates and allocates bandwidth to the tenants, leveraging the controller farm to offer a tenant-aware virtual network allocation. This provides differentiated QoS for various applications and devices sharing the network. The execution is broken into sub executions to enable parallel and independent executions. This Software-Defined Service Composition enables executions as microservices in the control plane. Multiple execution paths for each computation exist either physically across the connected devices or are virtually created as tenant execution spaces by the virtual tenant network allocation. Leveraging these potential multiple alternative execution paths, Software-Defined Service Composition enables breaking down complex computations into distributed service executions that can be executed in parallel in controller and CPS devices' firmware with differentiated priority and control.

Through a common API and an SDN-based approach, Software-Defined Service Composition enables web services to be composed through various distributed execution paradigms such as MapReduce [23] and Dryad [24], in addition to the traditional web services engines to fit the requirements of the CPS. It further allows the services detection and execution to be dynamic, to balance the load across various services nodes. It does so by leveraging the network load information readily available to the SDN controller, as well as the service-level information such as requests on the fly and the requests in the queue that are available to the web services engine, and the services deployment information available to the web services registry.

*2) **Modelling Sandbox**:* The *SD-CPS* modelling sandbox offers modelling and orchestrating capabilities, thus using the controller as a sandbox in modelling the complex CPS in real world. Software-Defined Simulations bring the simulations of SDN systems close to the systems that they model, where the system being simulated is separated from the simulated application logic. Following a software-defined approach, Software-Defined Simulation lets the developers model and continuously and iteratively design the CPS. Thus, the simulation in cyberspace will be closer to the execution in the cyber-physical deployment. The modelling sandbox offers dynamic management capabilities to heterogeneous systems by providing a software-defined approach to orchestrate various stages of development, from simulations, emulations, to physical deployments, minimizing repeated development efforts and manual deployment overheads.

III. *SD-CPS*: Addressing the Challenges of CPS

In this section, we will look at how *SD-CPS* tackles the core challenges of CPS [7], including modelling, incremental building and testing, seamless execution in a sandbox and production environments, scalability, reusability of services through service compositions, fault-tolerance, and resilience.

A. *Resilience in* SD-CPS

SD-CPS leverages the global knowledge of the entire CPS network of its controller deployment to ensure a resilient and high performant execution. Computation power is often rare physically at the mobile terminals to perform complex computations. Hence, computation-intensive algorithms of the mobile terminals and CPS physical devices are delegated to the cyberspace and executed as a composition of microservices, choosing virtual execution spaces in the controller environment. The microservice-based execution avoids repeated computation efforts. The data flow goes through various intermediaries in a traditional workflow. The workflows can be sent through the potential alternatives to ensure load balancing and fair and efficient resource utilization. Redundancy in execution alternatives enables workflows to be executed in a distributed and parallel manner when possible.

Furthermore, the path redundancy makes CPS fault-tolerant and ready to handle unexpected failures and congestion. The enhanced redundant execution paths and the global awareness of the controller on their existence ensures resilience, load balancing, and congestion control among the underlying network paths. Figure 2 models a wireframe of the underlying system of CPS with data flow between two smart devices, with multiple potential paths. The origin and destination nodes are the start and the end nodes of a communication initiated by a distributed computation. In a data center network, these nodes are hosts or servers, while the intermediate nodes are traditionally switches that connect the large underlying network. However, due to the heterogeneous nature of CPS, origin and/or destination can be smart mobile devices/terminals or virtual execution spaces in the controller, while intermediate and/or destination nodes can be surrogate nodes such as computer servers or switches as in a data center network. In mobile cyber-physical systems, these end nodes can be dynamic with an ever-changing geographic location, such as automobile devices, moving robots, or smart phones on the go. Without sacrificing the details, *SD-CPS* views this system as a connected network.

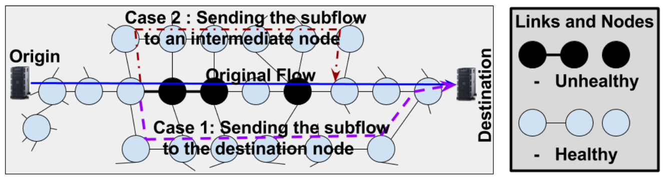
Case 2 : Sending the subflow to an intermediate node
Origin
Original Flow
Destination
Case 1: Sending the subflow to the destination node
Links and Nodes
- Unhealthy
- Healthy

Fig. 2. Execution as a Service Composition and Alternative Execution Paths

With the dynamic traffic of network flows, a few service or network nodes and links may become congested. Moreover, some nodes may be prone to failures. *SD-CPS* attempts to identify the congested, malfunctioning, or malicious nodes and links (that are highlighted and differentiated as unhealthy in Figure 2 for the ease of reference) through its southbound API, to alter the execution or data transfer path towards a healthy alternative dynamically.

When an intermediary is identified as failed or slow, *SD-CPS* enforces a partial redundancy in the data flows to ensure correctness and end-to-end delivery. *SD-CPS* approach creates subflows by diverting or cloning parts of the flows, and sends them towards a node known as the clone destination. In case 1, the clone destination is same as the original destination. However, case 2 has a clone destination that differs from the original. Here the cloned subflow is sent towards an intermediate node on the original path connecting the origin and destination. The flow is recomposed afterwards. The case 2 approach minimizes unnecessary redundancy when it is possible to recompose the flow at the clone destination or an intermediate node. When such a recompose of flows is impossible at an intermediate node due to the technical difficulties or due to the nature of the congestion or network failure itself, the flow is eventually recomposed when it reaches the destination node as in the case 1.

Network fabrics unify network devices such as switches to offer connectivity, centrally managed as a single unit. Fabrics allow mobility of nodes inside the network with centralized controller dynamically determining the network topology, hence is used in conjunction with SDN to support mobility in CPS. As the geographical shortest paths between the nodes change rather frequently in mobile CPS, the weight and cost of each path should be considered dynamic. This is specifically true to the edges or path segments that connect the mobile terminals to the network fabric or the edge switches in the network topology. Hence *SD-CPS* exploits the network fabrics in supporting dynamic network topologies for mobile CPS.

While an ongoing flow is served following the previous route, any subsequent flow should be routed considering the new geographic location of the terminal. For instance, a moving automobile can pass through various control stations, and hence the proximity of nodes differs. The flow tables are dynamically changed to reflect the changes in the cost of each path. Even existing elephant flows can be rerouted with minimal overhead, if the previously chosen path is not the best-fit any more. Thus *SD-CPS* supports the uncertainty and dynamic paths in CPS.

B. *Security in* SD-CPS

It is essential to secure the controller in *SD-CPS* for a correct execution, as an unprotected controller will become a vulnerability on its own. General researches on improving the SDN security are and will be relevant and applicable here, with further extensions for the southbound API for the CPS.

The centralized control avoids the potentials for a network segmentation. Thus, with the global knowledge of the CPS, the *SD-CPS* controller mitigates the risks of resource scarcity or external attacks in the intermediate nodes in the underlying network and system. Moreover, the awareness of the application and network enables the controller to differentiate the QoS offered to the tenant applications based on the importance or service-level agreements (SLA).

Distributed fault-tolerance and recovery upon system and network failures are handled efficiently using the controller as a centralized arbiter in the network. As reported for the traditional networks, threats on confidentiality, integrity, availability, and consistency are inherent to the network, and are not introduced by SDN itself [25]. The vulnerability in privacy due to the co-existence and shared space of tenants, and issues in scale are caused by poor implementation than the design of SDN. *SD-CPS* avoids these pitfalls through the highly available, multi-tenanted, federated controller deployment, designed as the controller farm.

C. *Modelling and Scaling CPS with* SD-CPS

The scale and complexity of the CPS increase due to either the larger number of devices and components, or their heterogeneity. The controller is the element with the highest processing power in the *SD-CPS* ecosystem. It manages the communication and coordination across all the entities, including the CPS, humans, and the tenant applications. The federated controller deployment ensures smooth scaling and decision making in large-scale execution environments. As the controller is multi-tenanted with protected access to multiple domains or tenant spaces, management and orchestration of the intelligent agents and their data in the cyberspace are handled seamlessly with scale.

Through Software-Defined Simulations, the designed systems are initially modelled as simulations that are still coordinated by the centralized controller in the same way the physical system that it models is coordinated. Hence, the simulations function as a virtual proxy for the system that is being designed. The systems are in practice implemented once in simulation, and then in physical deployment, reusing the same single effort, having controller as a unified execution space. As the modelling sandbox functions as a controlled modelling space of the designed CPS, unpredictability of the execution environment is significantly reduced.

Figure 3 represents how the systems are modelled in the sandbox environment of *SD-CPS* controller. The controller farm of *SD-CPS* orchestrates both the physical systems and their simulated counterparts in the cyberspace. With a one-to-one mapping between the simulated virtual intelligent agents and interdependent components of the physical system, the interactions are modelled and closely monitored in the controlled sandbox environment before the decisions are loaded into the physical space.

The model follows the Software-Defined Simulations and orchestration approach: it attempts to minimize the code duplication by executing the real code from the controller, instead of having a simulation or model running custom code, thus independent of the real execution. As the controller is developed in a high-level language such as Java, *SD-CPS* enables deployment of custom applications as controller plugins to alter or reprogram the behaviour of CPS. The physical system loads the decisions from the cyberspace. A

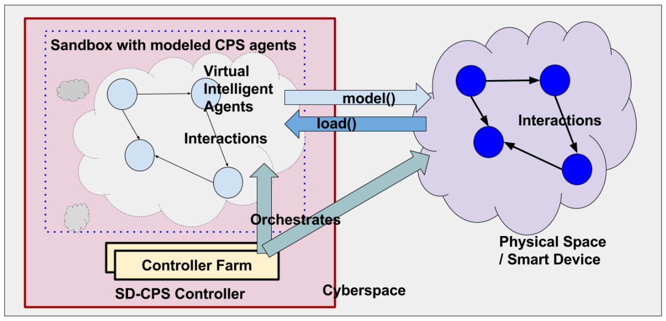

Fig. 3. Modelling with *SD-CPS* Approach

multi-tenanted execution space ensures modelling of multiple CPS in parallel.

IV. Current Prototype

We prototyped *SD-CPS* with OpenDaylight Beryllium [26] as the SDN controller, Oracle Java 1.8.0 as the programming language, and ActiveMQ 5.14.2 [27] as the message broker of MOM protocols. *SD-CPS* extends and leverages our previous work as the enablers of the software-defined approach for CPS.

Smart Connection and Data-to-Information Conversion Levels: CHIEF [28] designs the controller farm, a federated deployment of SDN controllers, to manage scalable multi-domain cloud networks. Initially designed for community network clouds, CHIEF was exploited as the *SD-CPS* controller farm for any large-scale network composed of multiple tenants with heterogeneous devices and access. In addition to the network management, CHIEF offers auxiliary services such as throttling and network monitoring through its event-based extended SDN architecture. *SD-CPS* extends Mayan [29] to offer Software-Defined Service Composition for microservices representing the CPS executions. Cassowary [30] designs Software-Defined Sensor Networks for smart buildings leveraging SDN and MOM protocols. We extend Cassowary to facilitate a wider adoption of SDN with loose coupling to the underlying network or SDN switches.

Cyber and Cognition Levels: SDNSim [31] offers Software-Defined Simulations. Built on top of SDNSim, SENDIM [32] enables systems to be designed and deployed seamlessly across various realizations and deployments. Originally developed for cloud and data centers, SENDIM is extended for CPS, IoT, or any software-defined systems and networks, as the modelling sandbox of *SD-CPS*. Thus, following the same descriptive language, a system can be modelled and executed in the cyberspace as a simulation, and later can be executed in the CPS with logic executing in the cyberspace while the physical device executes accordingly.

Configuration Level: Core configuration data is stored in the *SD-CPS* controller data store by extending the OpenDaylight controller data tree. *SD-CPS* exposes its persistent in-memory data store through the REST and MOM protocol implementations of the northbound API. The northbound manages the configuration of the internal data. The *SD-CPS* user interface can be integrated into the existing CPS interface seamlessly, hence avoiding learning curves and enabling easy adoption while ensuring backward compatibility.

Feasibility Assessment: Through a few simulations and microbenchmarks, we demonstrated that *SD-CPS* increases the potential scale of the CPS. *SD-CPS* controller performance was increased through the deployment of controller farm [28]. A near-linear performance growth with the number of controller instances up to a maximum value followed by a near-logarithmic growth was observed [29]. The reduced performance gain is due to idling controllers for each service execution. Hence, the performance gain depends on the problem size and its distribution as services in service composition. Furthermore, the modelling sandbox reduces the time in modelling as it offers a dual reality of cyber-physical spaces for simulations and designs.

V. Related Work

Use cases of SDN have been steadily spanning beyond the traditional networks, from sensor networks to smart buildings. Wireless Sensor Networks (WSN) [33] have the requirement to be context-aware. They need to handle a larger control traffic due to their dynamic nature compared to data center networks, while having to share the bandwidth among control and data traffic. Sensor OpenFlow (SOF) [34] identifies the benefits of a Software-Defined WSN, leveraging SDN for WSN. SOF increases manageability of WSN and adapts to policy changes of wireless networks and mobile devices.

SDN for Distributed Systems: As OpenDaylight leverages Akka[1] for a clustered deployment, it has also been leveraged for the distributed clusters of *SD-CPS* with a logically centralized view. OpenDaylight Federation[2] is an incubation project currently under development. OpenDaylight Federation allows multiple distributed clusters of OpenDaylight controllers to communicate and coordinate, offering a protected and limited access to remote clusters. Hence, it enables a super-cluster of controllers in wide area or even the Internet scale. It exploits the inter-cluster state sharing, following the same approach taken by other OpenDaylight projects such as OpenDaylight Conceptual Data Tree[3] and Messaging4Transport[4]. While *SD-CPS* exploits Messaging4Transport in building a controller farm as designed by CHIEF, OpenDaylight Federation can be used as an alternative to the controller farm. Conceptual Data Tree project also aims to offer federation, replication, and high availability beyond single controller deployments.

Albatross [35] is a membership service that addresses the uncertainty of distributed systems. Albatross tends to be 10 times more efficient than the previous membership services by exploiting the standard interface offered by SDN to monitor and configure network elements. It addresses common network failures, while avoiding interfering with working processes and machines, and maintaining a quick response time and

[1]http://akka.io/
[2]https://wiki.opendaylight.org/view/Federation:Main
[3]https://wiki.opendaylight.org/view/MD-SAL:Boron:Conceptual_Data_Tree
[4]https://wiki.opendaylight.org/view/Messaging4Transport:Main

high overall availability. The challenges such as split-brain scenarios and violations in consistency and availability that are addressed by Albatross are relevant for CPS too. However, while CPS is a distributed system, it has its own peculiar challenges due to its diverse nature in implementation and devices as we discussed earlier. While Albatross aims to address the uncertainty and challenges in ensuring reachability of distributed systems, *SD-CPS* aims to mitigate a list of challenges inherent to CPS by innovating a logically centralized control plane and management architecture. Albatross approach narrows itself to data centers equipped with software-defined networks, leaving wide area networks as a future work. *SD-CPS* is designed for CPS to scale up to the Internet scale.

Smart Environments and CPS: Software-Defined Environment (SDE) [36], [37] focuses on factors such as, (i) resource abstraction based on capability, (ii) workload abstraction and definition based on policies, goals, and business/mission objectives, (iii) workload orchestration, and (iv) continuous mapping and optimization of workload and the available resources. SDN controller and physical and virtual SDN switches remain the heart of SDE. The control of compute, network, and storage is built atop a virtualized network. By leveraging Network Function Virtualization (NFV) [38], middlebox actions typically handled by hardware middleboxes such as firewalls and load balancers are replaced by relevant software components.

Software-Defined Buildings (SDB) [39] envision a Building Operating System (BOS) which functions as a sandbox environment for various device firmwares to run as applications atop it. The BOS spans across multiple buildings in a campus, rather than confining itself to a single building. The scale of SDB and SDE can be increased through collaboration and coordination of the controllers of the buildings or environments. While SDB and SDE architectures can be extended for CPS, they cannot cater for CPS by themselves due to the variety and heterogeneity in the architecture and requirements of CPS. The increased dynamics and mobility of CPS compared to the environments controlled by SDB and SDE further hinder adopting them for CPS.

Software-Defined Internet of Things (SDIoT): SDIoT [40] proposes a software-defined architecture for IoT devices by handling the security [41], storage [42], and network aspects in a software-defined approach. SDIoT proposes an IoT controller composed of controllers of software-defined networking, storage, security, and others. This controller operates as an orchestrating middleware between the data-as-a-service layer consists of end user applications, and the physical layer consists of the database pool and sensor networks. Software-Defined Industrial Internet of Things [43] extends SDIoT to Industrial Internet of Things (IIoT), to offer more flexible networks to Industry 4.0. As CPS is a core enabler of Industry 4.0 [22], adopting SDN for IIoT is highly related to *SD-CPS* approach, though unlike *SD-CPS* this work is focused more on safety, reliability, and standardization aspects.

Multinetwork INformation Architecture (MINA) self-observing and adaptive middleware [44] has been extended with a layered SDN controller to implement a controller architecture for IoT [45]. Various research and enterprise use cases are proposed and implemented, including SDIoT for smart urban sensing [46], and end-to-end service network orchestration [47]. While sharing similarities with IoT, CPS is set to address a larger set of problems with more focus on ground issues on interoperability of cyber and physical spaces and dimensions in a CPS. Hence, *SD-CPS* differs in scope to that of SDN for IoT researches such as SDIoT, though they share similar motivation.

VI. Conclusion and Future Work

In this paper, we presented *SD-CPS*, an approach and architecture that aims to mitigate the application and design challenges faced by CPS. *SD-CPS* leverages the SDN switches and controllers when available, while employing an approach motivated by SDN even during the absence of SDN switches. Hence it remains compatible with and applicable to existing CPS deployments that do not have SDN. *SD-CPS* opens many research avenues on envisioning and improving SDN for CPS architectures and evaluating implementation alternatives.

We observe that by extending and adopting an architecture inspired by SDN, CPS can be offered a unified control. By exploiting the cyberspace of CPS for modelling the systems, repeated effort to model the system in cyber and physical spaces is avoided. Furthermore, a unified control space supports a fine-grain management of devices with minimal communication and coordination overhead. Resilience of selected critical flows can be improved by categorically specifying rules in the control plane, hence offering SLA-awareness to higher priority processes originated by the heterogeneous devices.

As a future work, *SD-CPS* approach should be incorporated with implementations of various networking and integration protocols for multiple use case scenarios. Furthermore, the *SD-CPS* deployments should be tested against baseline implementations of various CPS for their efficiency in addressing the identified shortcomings. We identify a few potential future research avenues that can extend the *SD-CPS* approach for CPS in the presence of a centralized controller architecture: (i) securing CPS in the presence of malicious participants, (ii) increasing the resource efficiency of CPS by using the locality data in the Internet scale, and (iii) metering and billing the tenant resource consumption for a fair resource allocation.

Acknowledgements: This work was supported by national funds through Fundação para a Ciência e a Tecnologia with reference UID/CEC/50021/2013 and a PhD grant offered by the Erasmus Mundus Joint Doctorate in Distributed Computing (EMJD-DC).

References

[1] F. Xia, L. T. Yang, L. Wang, and A. Vinel, "Internet of things," *International Journal of Communication Systems*, vol. 25, no. 9, p. 1101, 2012.

[2] E. A. Lee, "The past, present and future of cyber-physical systems: A focus on models," *Sensors*, vol. 15, no. 3, pp. 4837–4869, 2015.

[3] L. Atzori, A. Iera, and G. Morabito, "The internet of things: A survey," *Computer networks*, vol. 54, no. 15, pp. 2787–2805, 2010.

[4] S. Munir and J. A. Stankovic, "Depsys: Dependency aware integration of cyber-physical systems for smart homes," in *Cyber-Physical Systems (ICCPS), 2014 ACM/IEEE International Conference on*. IEEE, 2014, pp. 127–138.

[5] J. Pacheco, C. Tunc, and S. Hariri, "Design and evaluation of resilient infrastructures systems for smart cities," in *Smart Cities Conference (ISC2), 2016 IEEE International*. IEEE, 2016, pp. 1–6.

[6] P. Leitão, A. W. Colombo, and S. Karnouskos, "Industrial automation based on cyber-physical systems technologies: Prototype implementations and challenges," *Computers in Industry*, vol. 81, pp. 11–25, 2016.

[7] E. A. Lee, "Cyber physical systems: Design challenges," in *2008 11th IEEE International Symposium on Object and Component-Oriented Real-Time Distributed Computing (ISORC)*. IEEE, 2008, pp. 363–369.

[8] ——, "Computing foundations and practice for cyber-physical systems: A preliminary report," *University of California, Berkeley, Tech. Rep. UCB/EECS-2007-72*, 2007.

[9] M. Persson and A. Håkansson, "A communication protocol for different communication technologies in cyber-physical systems," *Procedia Computer Science*, vol. 60, pp. 1697–1706, 2015.

[10] A. Cardenas, S. Amin, B. Sinopoli, A. Giani, A. Perrig, and S. Sastry, "Challenges for securing cyber physical systems," in *Workshop on future directions in cyber-physical systems security*, 2009, p. 5.

[11] P. Derler, E. A. Lee, and A. S. Vincentelli, "Modeling cyber–physical systems," *Proceedings of the IEEE*, vol. 100, no. 1, pp. 13–28, 2012.

[12] Z. Qin, N. Do, G. Denker, and N. Venkatasubramanian, "Software-defined cyber-physical multinetworks," in *Computing, Networking and Communications (ICNC), 2014 International Conference on*. IEEE, 2014, pp. 322–326.

[13] D. Antonioli and N. O. Tippenhauer, "Minicps: A toolkit for security research on cps networks," in *Proceedings of the First ACM Workshop on Cyber-Physical Systems-Security and/or PrivaCy*. ACM, 2015, pp. 91–100.

[14] X. Dong, H. Lin, R. Tan, R. K. Iyer, and Z. Kalbarczyk, "Software-defined networking for smart grid resilience: Opportunities and challenges," in *Proceedings of the 1st ACM Workshop on Cyber-Physical System Security*. ACM, 2015, pp. 61–68.

[15] M. Jarschel, T. Zinner, T. Hoßfeld, P. Tran-Gia, and W. Kellerer, "Interfaces, attributes, and use cases: A compass for sdn," *IEEE Communications Magazine*, vol. 52, no. 6, pp. 210–217, 2014.

[16] E. Curry, "Message-oriented middleware," *Middleware for communications*, pp. 1–28, 2004.

[17] S. Vinoski, "Advanced message queuing protocol," *IEEE Internet Computing*, vol. 10, no. 6, p. 87, 2006.

[18] D. Locke, "Mq telemetry transport (mqtt) v3. 1 protocol specification," *IBM developerWorks Technical Library], available at http://www. ibm. com/developerworks/webservices/library/ws-mqtt/index. html*, 2010.

[19] U. Hunkeler, H. L. Truong, and A. Stanford-Clark, "Mqtt-s—a publish/subscribe protocol for wireless sensor networks," in *Communication systems software and middleware and workshops, 2008. comsware 2008. 3rd international conference on*. IEEE, 2008, pp. 791–798.

[20] M. Collina, G. E. Corazza, and A. Vanelli-Coralli, "Introducing the qest broker: Scaling the iot by bridging mqtt and rest," in *2012 IEEE 23rd International Symposium on Personal, Indoor and Mobile Radio Communications-(PIMRC)*. IEEE, 2012, pp. 36–41.

[21] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "Openflow: enabling innovation in campus networks," *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 2, pp. 69–74, 2008.

[22] J. Lee, B. Bagheri, and H.-A. Kao, "A cyber-physical systems architecture for industry 4.0-based manufacturing systems," *Manufacturing Letters*, vol. 3, pp. 18–23, 2015.

[23] J. Dean and S. Ghemawat, "Mapreduce: simplified data processing on large clusters," *Communications of the ACM*, vol. 51, no. 1, pp. 107–113, 2008.

[24] M. Isard, M. Budiu, Y. Yu, A. Birrell, and D. Fetterly, "Dryad: distributed data-parallel programs from sequential building blocks," in *ACM SIGOPS Operating Systems Review*, vol. 41, no. 3. ACM, 2007, pp. 59–72.

[25] L. Schehlmann, S. Abt, and H. Baier, "Blessing or curse? revisiting security aspects of software-defined networking," in *10th International Conference on Network and Service Management (CNSM) and Workshop*. IEEE, 2014, pp. 382–387.

[26] J. Medved, R. Varga, A. Tkacik, and K. Gray, "Opendaylight: Towards a model-driven sdn controller architecture," in *Proceeding of IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks 2014*, 2014.

[27] B. Snyder, D. Bosnanac, and R. Davies, *ActiveMQ in action*. Manning, 2011, vol. 47.

[28] P. Kathiravelu and L. Veiga, "Chief: Controller farm for clouds of software-defined community networks," in *2016 IEEE International Conference on Cloud Engineering Workshop (IC2EW)*, April 2016, pp. 1–6.

[29] P. Kathiravelu, T. G. Grbac, and L. Veiga, "Building blocks of mayan: Componentizing the escience workflows through software-defined service composition," in *Web Services (ICWS), 2016 IEEE International Conference on*. IEEE, 2016, pp. 372–379.

[30] P. Kathiravelu, L. Sharifi, and L. Veiga, "Cassowary: Middleware platform for context-aware smart buildings with software-defined sensor networks," in *Proceedings of the 2nd Workshop on Middleware for Context-Aware Applications in the IoT*. ACM, 2015, pp. 1–6.

[31] P. Kathiravelu and L. Veiga, "Software-defined simulations for continuous development of cloud and data center networks," in *OTM Confederated International Conferences" On the Move to Meaningful Internet Systems"*. Springer, 2016, pp. 3–23.

[32] ——, "Sendim for incremental development of cloud networks: Simulation, emulation and deployment integration middleware," in *Cloud Engineering (IC2E), 2016 IEEE International Conference on*. IEEE, 2016, pp. 143–146.

[33] K. Romer and F. Mattern, "The design space of wireless sensor networks," *IEEE wireless communications*, vol. 11, no. 6, pp. 54–61, 2004.

[34] T. Luo, H.-P. Tan, and T. Q. Quek, "Sensor openflow: Enabling software-defined wireless sensor networks," *IEEE Communications Letters*, vol. 16, no. 11, pp. 1896–1899, 2012.

[35] J. B. Leners, T. Gupta, M. K. Aguilera, and M. Walfish, "Taming uncertainty in distributed systems with help from the network," in *Proceedings of the Tenth European Conference on Computer Systems*. ACM, 2015, p. 9.

[36] C. Dixon, D. Olshefski, V. Jain, C. DeCusatis, W. Felter, J. Carter, M. Banikazemi, V. Mann, J. M. Tracey, and R. Recio, "Software defined networking to support the software defined environment," *IBM Journal of Research and Development*, vol. 58, no. 2/3, pp. 3:1–3:14, 2014.

[37] C.-S. Li, B. Brech, S. Crowder, D. Dias, H. Franke, M. Hogstrom, D. Lindquist, G. Pacifici, S. Pappe, B. Rajaraman *et al.*, "Software defined environments: An introduction," *IBM Journal of Research and Development*, vol. 58, no. 2/3, pp. 1:1–1:11, 2014.

[38] J. Batalle, J. F. Riera, E. Escalona, and J. A. Garcia-Espin, "On the implementation of nfv over an openflow infrastructure: Routing function virtualization," in *Future Networks and Services (SDN4FNS), 2013 IEEE SDN for*. IEEE, 2013, pp. 1–6.

[39] S. Dawson-Haggerty, J. Ortiz, J. Trager, D. Culler, and R. H. Katz, "Energy savings and the "software-defined" building," *IEEE Design & Test of Computers*, vol. 29, no. 4, pp. 56–57, 2012.

[40] Y. Jararweh, M. Al-Ayyoub, E. Benkhelifa, M. Vouk, A. Rindos *et al.*, "Sdiot: a software defined based internet of things framework," *Journal of Ambient Intelligence and Humanized Computing*, vol. 6, no. 4, pp. 453–461, 2015.

[41] M. Al-Ayyoub, Y. Jararweh, E. Benkhelifa, M. Vouk, A. Rindos *et al.*, "Sdsecurity: a software defined security experimental framework," in *2015 IEEE International Conference on Communication Workshop (ICCW)*. IEEE, 2015, pp. 1871–1876.

[42] A. Darabseh, M. Al-Ayyoub, Y. Jararweh, E. Benkhelifa, M. Vouk, and A. Rindos, "Sdstorage: a software defined storage experimental framework," in *Cloud Engineering (IC2E), 2015 IEEE International Conference on*. IEEE, 2015, pp. 341–346.

[43] J. Wan, S. Tang, Z. Shu, D. Li, S. Wang, M. Imran, and A. V. Vasilakos, "Software-defined industrial internet of things in the context of industry 4.0," *IEEE Sensors Journal*, vol. 16, no. 20, pp. 7373–7380, 2016.

[44] Z. Qin, L. Iannario, C. Giannelli, P. Bellavista, G. Denker, and N. Venkatasubramanian, "Mina: A reflective middleware for managing dynamic multinetwork environments," in *2014 IEEE Network Operations and Management Symposium (NOMS)*. IEEE, 2014, pp. 1–4.

[45] Z. Qin, G. Denker, C. Giannelli, P. Bellavista, and N. Venkatasubramanian, "A software defined networking architecture for the internet-of-things," in *2014 IEEE network operations and management symposium (NOMS)*. IEEE, 2014, pp. 1–9.

[46] J. Liu, Y. Li, M. Chen, W. Dong, and D. Jin, "Software-defined internet of things for smart urban sensing," *IEEE Communications Magazine*, vol. 53, no. 9, pp. 55–63, 2015.

[47] R. Vilalta, A. Mayoral, D. Pubill, R. Casellas, R. Martínez, J. Serra, C. Verikoukis, and R. Muñoz, "End-to-end sdn orchestration of iot services using an sdn/nfv-enabled edge node," in *Optical Fiber Communication Conference*. Optical Society of America, 2016, pp. W2A–42.