IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

SPECIAL SECTION ON CYBER-PHYSICAL SYSTEMS

Received May 27, 2018, accepted July 2, 2018, date of publication July 17, 2018, date of current version August 15, 2018.

Digital Object Identifier 10.1109/ACCESS.2018.2856509

PsCPS: A Distributed Platform for Cloud and Fog Integrated Smart Cyber-Physical Systems

JAMEELA AL-JARODI[1], (Member, IEEE), AND NADER MOHAMED[2], (Member, IEEE)
[1]Department of Engineering, Robert Morris University, Moon, PA 15108, USA
[2]Middleware Technologies Laboratory, Pittsburgh, PA 15057, USA

Corresponding author: Nader Mohamed (nader@middleware-tech.net)

ABSTRACT Smart cyber-physical systems (sCPS) extend the traditional CPS by introducing intelligent and autonomous capabilities to these systems. sCPS provide smart interactions, smart controls, and smart enhancements for the physical world. These smart features can enhance the operations, efficiency, safety, utilization, reliability, quality, and cost-effectiveness of the physical world. These systems are usually highly distributed, real-time, deal with huge data sets, implement intelligent algorithms, and need powerful computation power and large-scale storage capacity. Some of the promising approaches to achieve the sCPS objectives include the use of a combination of cloud computing and fog computing to enable developing and operating them. Cloud computing can provide scalable and powerful computation platforms, large storage capacities, and advanced and intelligent software services, while fog computing can provide more optimized real-time controls for sCPS. Although cloud and fog computing can provide many advantages for sCPS, developing and integrating all these systems is challenging. This is due to the strict requirements of sCPS on one hand and the types of distributed and heterogeneous environments these systems support on the other. This paper proposes a distributed platform for cloud and fog integrated sCPS, named PsCPS. This platform can be distributed among multiple clouds, multiple fog nodes, and sCPS subsystems to provide services to relax many challenges of such integration. The proposed platform includes system and application agents that can be deployed on participating nodes to provide different services for cloud and fog integrated sCPS. These agents can be developed, implemented, controlled, and managed as a set of single agents, as multi-agent systems, or as hierarchical multi-agent systems. A prototype of the proposed platform is implemented and evaluated as well.

INDEX TERMS Cyber-physical systems, smart CPS, distributed platform, cloud computing, fog computing, software agent.

I. INTRODUCTION

In the last decade, there have been numerous research and development activities studying and implementing Cyber-Physical Systems (CPS) that offer valuable interactions between the physical and cyber worlds [1]. The physical world includes the machines, environments, infrastructures and humans and the cyber world is the control software that executes on computers and microcontrollers and implement useful algorithms to enable the interactions with the physical world. CPS generally utilize and link numerous technologies and ideas from software, networks, distributed systems, embedded systems, and control systems. In addition, it involves various hardware such as microcontrollers, sensors and actuators. Moreover, they involve other fields such as mechanical, biomedical, civil, and electrical engineering to deliver added values to applications in the physical world [2].

A special and more advanced type of CPS are the smart CPS (sCPS). As the name indicates, sCPS provide smart interactions, controls, and enhancements for the physical world. These smart features can enhance the operations, efficiency, safety, reliability, quality, and cost-effectiveness of the physical world. sCPS have numerous applications in smart buildings, smart cities, smart manufacturing, smart grids, intelligent transportation systems, smart infrastructure monitoring, smart healthcare, and smart logistics. For example, sCPS in healthcare applications can provide useful real-time services for patients monitoring and treatments. sCPS in smart buildings can improve energy efficiency and living and

working conditions. In addition, it can be used in transportation systems to enhance safety and efficiency.

There are high systems requirements for successful implementations and operations of sCPS. These include managing and analyzing large-scale data sets, operating in large environments with complex processes, offering intelligent and smart operations and optimized decisions. Cloud computing can provide powerful computational resources, scalable storage capacity, and advanced software services that are needed by sCPS. Examples of such advanced services are knowledgebase implementations, machine leaning, data mining, data analytics, optimizations, and simulation services. Cloud computing has been proposed to be used for various sCPS applications such as robotics and automation [3], smart buildings [4], smart healthcare [5], and smart transportation systems [6]. However, cloud computing cannot handle some technical and performance requirements needed by sCPS like providing low latency services, offering location-aware services, providing better scalability support for geographically widely distributed applications, supporting streaming communication and processing, supporting mobility and localized access control, offering better Quality of Services (QoS) support, and providing context-aware processing, communication, and decisions.

Fortunately, fog computing has been proposed to provide solutions for these requirements [7]. Fog computing can utilize edge devices such as routers and dedicated computers to operate cloud-like services to support different sCPS operations. The services can be control, communication, storage, processing, configuration, monitoring, measurement, and management services to support a sCPS application. The Fog facilitates executing services geographically close to the components of the sCPS applications and at the same time it can access and use services provided by the cloud. Due to its great advantages, many projects have been started to utilize fog computing for sCPS applications in healthcare [8], energy management [9], and smart cities [10], to name a few.

Although both cloud computing and fog computing can provide many advantages for sCPS applications, developing such applications that can effectively and efficiently utilize both is not trivial. This is due to several technical challenges including how to develop, manage, control, monitor, and secure this type of integration and operations. In this paper, we propose a distributed platform, named PsCPS, to help address these challenges. This platform can be used to implement and operate cloud and fog integrated sCPS.

In the rest of the paper we offer background information about CPS and sCPS in Section II. In Section III, we discuss cloud and fog integrated sCPS and their requirements. Section IV discusses the architecture of PsCPS, its components, functions, and operations and Section V provides a discussion of a PsCPS prototype implementation. Section VI discusses example scenarios of utilizing PsCPS for sCPS applications and their benefits. Experimental evaluations are discussed in Section VII and Section VIII offers a discussion of related work. Concluding remarks and future directions are offered in Section IX.

II. CYBER-PHYSICAL SYSTEMS AND SMART CYBER-PHYSICAL SYSTEMS

CPS are embedded systems, characterized by strong and continuous interactions between the physical and cyber components [2]. CPS are being increasingly used everywhere, featuring physical domains such as energy, manufacturing, healthcare, civil infrastructures, automotive, transportation, aerospace, entertainment, and consumer appliances. A great portion of CPS is designed to support smart and context-aware mission-critical applications [1]. Predefined goals of the relevant application domain are achieved through the monitoring and control processes, as provided by the CPS. The control decisions are usually done by the cyber world using specific algorithms implemented by software. Unlike conventional embedded systems, CPS are multifaceted embedded systems that feature distributed components and processing capabilities. Microcontrollers, sensors and actuators are examples of embedded computing devices that are integrated within the CPS. These devices are usually connected using wired or wireless networks and are tightly coupled with their physical environment.

The controls can be divided to three main tasks:

1) Monitoring the status of the physical system or environment using different types of sensors attached to the elements of the physical system or environment.
2) Making decisions to control the operations or conditions of the physical system or environment to meet the predefined application objectives.
3) Initiating actions through actuators connected to the different elements of the physical system or environment.

These three main tasks are linked in a feedback loop known as the closed-loop control (see Figure 1) to allow the CPS to provide full monitoring and control functions to meet the predefined objectives.

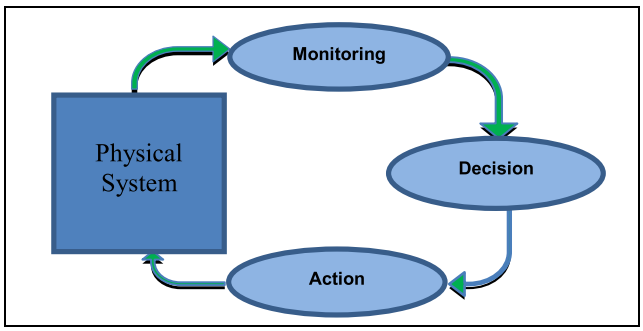

FIGURE 1. Closed-loop control steps of CPS.

The three tasks and the closed-loop control represent the main functions of basic CPS. As software solutions used for CPS become more complex and utilize intelligent mechanisms and advanced analysis and decision-making processes,

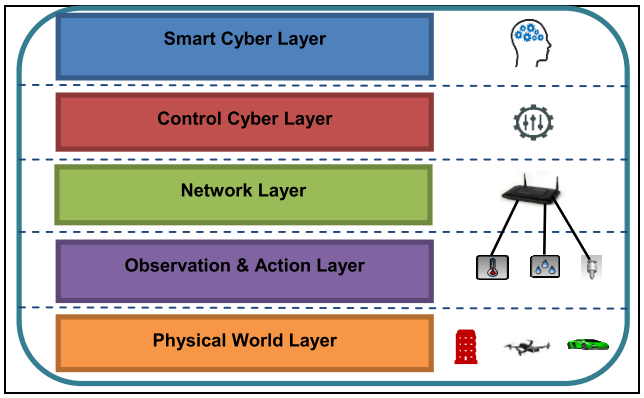

FIGURE 2. sCPS layers.

it is more accurate to refer to these systems as smart CPS (sCPS). sCPS being embedded in physical environments through intelligent mechanisms, they enable smart interactions between the physical and cyber elements. The functions of many sCPS will also include the required features and operations for the application domain it represents. Therefore, the overall functions of sCPS are more complicated due to the various technical challenges and application domains requirements. In addition, sCPS have smart features for different enhancements. In general, it is possible to organize the different components of sCPS in five layers as shown in Figure 2.

1) **Physical World Layer:** This includes any physical environment or system such as buildings, aircrafts, vehicles, or humans in addition to some intangible physical conditions such as temperature, sound, smell etc.
2) **Observation and Action Layer:** This layer includes the sensors that monitor the associated environment or system. In additions, it includes the actuators used to act upon or initiate actions in the environment or system in response to certain decisions or monitored conditions.
3) **Network Layer:** The components of CPS are usually distributed and connected via communication networks. In this layer, the communication networks can be wired, wireless, or a combination of both. They can have varying scales from a nanoscale network to a global wide area network (like the Internet). This is usually determined by the CPS applications and their required coverage areas. Using the network connectivity, it is possible to convert a set of individual sensors into a wireless or wired sensor network. Likewise, we can convert individual actuators to a wireless or wired actuator network. In both cases, the network layer will provide better mechanisms to work with these sensors and actuators as a system rather than individual components. Examples of these mechanisms are addressing, routing, and forwarding schemes for efficient communication.
4) **Control Cyber Layer:** This layer is part of the cyber world. It collects current status information from CPS sensors and sensor networks to make real-time decisions based on pre-defined objectives. These decisions then trigger action controls that are sent to actuators or actuator networks to be executed. This layer is implemented by centralized or distributed software. In a centralized approach the software executes on one microcontroller or computer/server and it is usually suitable for small-scale CPS. A distributed control software uses multiple distributed compute nodes to execute programs and provide the necessary operations across the CPS. This approach is generally more complex, yet more reliable and suitable for medium and large-scale CPS. In addition, using a distributed approach provisions for requirements like scalability and real-time support.
5) **Smart Cyber Layer:** This is a software layer used to build a knowledgebase about the corresponding physical environments and CPS. It is an advanced sophisticated layer that helps introduce smart features and optimization models for CPS, thus it is only available in advanced CPS applications. The knowledgebase is built over time as sensors collect environmental and system information and also monitor and record the effects of the actions taken on the environment or system. The collected information is organized and refined over time to help provide advanced smart features to the CPS applications such as prediction models, optimizations and smart decisions. Part of this layer includes data mining and learning capabilities to discover new knowledge from gathered information.

Table 1 provides a summary of some sCPS applications in terms of the physical and cyber parts they include and the benefits of utilizing sCPS.

III. CLOUD AND FOG INTEGRATED sCPS

Cloud and fog computing can provide services to effectively implement and improve some of the sCPS layers shown in Figure 2 and discussed in the previous section. For example, the network layer in sCPS can be enhanced using services provided by fog computing such as communication filtering, streaming, and data fusion for large-scale sCPS. In addition, fog computing can provide services to completely or partially implement both the control cyber layer and the smart cyber layer. Alternatively, it can provide support services for both layers. At the same time, cloud computing can provide powerful resources to implement and operate the smart cyber layer.

Cloud and fog integrated sCPS can be developed using the three-layer architecture shown in Figure 3. In this architecture, the cloud computing layer will provide the necessary large scale and resource intensive services such as powerful processing, scalable data storage, intelligent and complex algorithms for machine learning, data analytics, decision-making, and optimization, in addition to integration services with other systems, and development tools. The fog

TABLE 1. The cyber and physical worlds in sCPS applications and their benefits.

sCPS	The Physical World	The Cyber World	sCPS Applications' Benefits
Smart Healthcare	Patients, healthcare staff, illnesses, diagnostics and monitoring devices, medications and treatment procedures	Software to monitor and control patient health status and data	Timely and accurate patient monitoring and treatments
Smart Buildings	Buildings, appliances, HVAC systems, temperature, lighting, air quality and residents	Software/devices to monitor environmental conditions and energy use and implement algorithms to control equipment	Reduced energy consumption and enhanced quality of life
Smart Grids	Electricity, fuel, power generators, distribution networks, workers, consumer devices and consumers	Software enabling real-time monitoring and control of energy production, distribution and consumption	Optimized energy utilization, reduced overload risks and energy waste
Smart Water Networks	Water, distribution networks, storage systems, pumps, workers, consumer devices, and consumers	Software/devices to monitor and control the process of processing, transferring and storing water and its quality and usage	Reduced water loss, optimized water production and utilization, and enhanced water quality
Smart Manufacturing	Factories, machines, raw material, products, processes, workers, and warehouses	Real-time algorithms to monitor/control production processes, equipment and material utilization, and product quality	Optimized production and maintenance and enhanced product quality
Self-Driving Vehicles	Vehicles, onboard equipment, energy, location, roads, signs, traffic lights, traffic laws, other vehicles and passengers	Smart algorithms to automate vehicle driving and to maintain driving safety	Reduced transportation costs, optimized traffic flow and enhanced safety
Smart Traffic Lights	Traffic lights, vehicles and status data like speed, numbers and locations, roads, pedestrians and intersections	Real-time algorithms to monitor traffic status and control traffic lights to enhance traffic flow	Reduced traffic delays, minimized vehicles' travel times, and increased vehicles' average velocity.
Smart Greenhouse Control	plants, climatic conditions, soil, ventilation, carbon dioxide, water and HVAC equipment	Algorithms to regulate greenhouse climate, optimize resources utilization and maximize production	Enhanced plants growth and produce quantity and quality, and optimized resources utilization

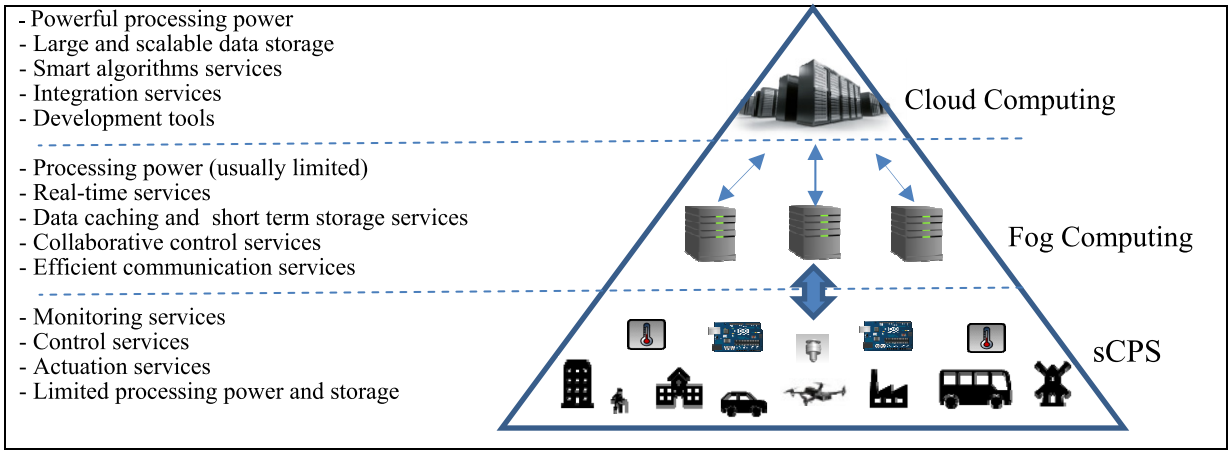

FIGURE 3. Cloud and fog integrated sCPS layered architecture.

computing layer will provide, limited processing power for local use, real-time services, data caching and limited data storage services, collaborative control services, streaming processing and communication services, in addition to other localized and context sensitive services. The sCPS layer has the actual sCPS components including the physical systems, sensors, actuators, and control devices. While this architecture is similar to the three-tier client/server architecture, there is a major difference. Unlike the client/server architecture, where service requests are one-directional from the clients to the servers; in this model the service requests can be multi-directional. The sCPS layer can issue requests to both the fog and cloud layers, the fog layer can issue requests to the cloud layer above and the sCPS layer below, and the cloud layer can issue requests to the fog and sCPS layers. For example, the limited resources sCPS components can issue a request to offload computation to the fog layer for some tasks, while the computing service in the fog layer can request to retrieve some data stored on the cloud. At the same time, the cloud can ask the fog layer to filter some information, which the fog layer will ask for from the connected sCPS components.

There are two models of integrating and utilizing cloud and fog computing for sCPS:

1) Integration with a single sCPS: cloud and fog computing can provide several services for a single sCPS such as offloading computations, data storage, advanced smart algorithm services such as machine leaning, data mining, simulation, streaming, and caching. In addition, the cloud and fog can be used to enable the integration among components of a large-scale smart system that extends over a large geographic area such as a smart grid and a smart water network. Cloud and

fog computing can provide integration infrastructures for these large-scale smart systems to smoothly link the components reliably and securely. The compute components in such systems can be all integrated together through cloud and fog computing.

2) Integration with a system of multiple sCPS: some systems consist of multiple sCPS. Each of these sCPS can be considered an individual cyber-physical system unit or node. Example of these nodes can be robots, vehicles, traffic lights, and smart buildings. Several of these nodes can be integrated together in a single large sCPS. Cloud and fog computing can provide, in addition to the services offered for a single sCPS, collaborative services among these nodes to build applications involving multi-robot systems [13], collaborative drones [16], cooperative vehicles [17], and collaborative smart buildings [18].

As implementing and operating cloud and fog integrated sCPS can be challenging, the availability of a special platform for developing and operating such systems can provide many advantages. However, there are some requirements that such platform should provide for cloud and fog integrated sCPS:

- **Cloud, fog, and sCPS components integration:** The main goal for the platform is to allow sCPS applications to gain access to the capabilities and services available on fog and cloud nodes to enhance their operations and performance. The platform should enable the integration among sCPS components, fog computing, and cloud computing. This integration will facilitate the exchange of services and better utilization of these services at the different levels as needed.
- **Fog functions development and deployment:** When new fog functions or services are needed for the sCPS applications, development and deployment capabilities are necessary for uniform implementation of these services and proper integration with already available services. The platform should enable the development and deployment of new fog functions as needed.
- **Dynamic loading for fog functions:** Fog computing provides many functions for different sCPS applications. These functions need to be made available on the integrated fog nodes for use by the sCPS applications. Yet fog nodes usually have limited resources including memory, processing, and storage. Therefore, the platform should provide mechanisms to dynamically load needed functions and remove unneeded ones.
- **Communication mechanisms support:** There are different communication mechanisms needed by different distributed functions and services of cloud and fog integrated sCPS. One example is the remote function/service calls which provide one-to-one synchronous request/response communication. Another example is publish/subscribe communication which provides asynchronous one-to-many communication and is used to send notifications. In addition, there is message passing and multicasting. Different cloud and fog integrated sCPS applications need to use different combinations of these communication mechanisms to achieve the required connectivity for integration among the components of the cloud and fog integrated sCPS. Therefore, the platform needs to have the ability to support these mechanisms and provide easy access to them for the different services.
- **Collaboration among multiple fog nodes:** While for some cloud and fog integrated sCPS applications each fog node can efficiently operate independently from the others to satisfy the requirements of the applications; there are applications that require collaboration and communication among some fog nodes to optimize their operations. For example, in a smart traffic light system, multiple fog nodes at different intersections will need to exchange traffic patterns information to optimize and synchronize the lights operations. Furthermore, multiple robots tasked with one large operation will need to coordinate sub tasks and create an efficient workflow among them. In such cases, the platform must offer mechanisms for collaboration and communication.
- **Hierarchical fog nodes organization support:** Some cloud and fog integrated sCPS applications with a large number of fog nodes need to virtually organize the fog nodes in a hierarchical structure for efficient communication and operations [11], [12]. Fog nodes can be organized in multiple levels in a hierarchical structure such that each intermediate fog node has a single parent node and one or more child nodes. In this structure, integrated cloud and fog sCPS will be formed as n-tier architectures where there will be multiple intermediate fog level layers. In this structure, each fog node mainly communicates and provides/uses functions/services to and from its parent and children. An example highlighting the need for this structure is a large-scale sCPS such as that used for smart cities [11]. In smart cities, different infrastructure components such as smart buildings, smart traffic systems, and smart pipeline monitoring need to be monitored and controlled. Cloud and fog services can provide many advantages for such monitoring and control. However, it is inefficient to use one level of fog nodes with similar functions [11]. It is better to use multiple levels of fog nodes such that each level provides different functions for the applications. For example, the fog nodes in the lower level can provide services for individual components like a single smart building or a single smart traffic light. The fog nodes in the intermediate level can provide services for a number of components together. In this level, each fog node can be responsible for a number of smart components in a neighborhood. The fog nodes in the upper level can provide services for all smart components at the level of communities, while the cloud can be used to provide services at the level of a whole city. In this structure, the fog nodes at the lower levels can provide very low latency services for smart components but at very

small scale, while the fog nodes at the upper level can provide large-scale services that need more powerful resources but also can tolerate high latency. As fog nodes have different characteristics at different levels in terms of scale and latency, different services can be distributed among these levels in such way that the requirements of smart applications are satisfied in a more efficient way.

- **Location Identification and Awareness:** One or more fog nodes are available in an area to provide services for certain smart sCPS components in that area. Therefore, requests for services can be associated with a specific location instead of a specific fog node with specific IP address. As an example, a request can come from a cloud application to check the status of a specific smart traffic light or smart building. Furthermore, a request can come to check all smart buildings in a neighborhood. In addition, some sCPS components that can provide services may be mobile such as robots. These components can only provide useful services if they are at a certain location. For example, a mobile robot will only transmit useful information when it arrives at the designated location. As a result, the platform should provide support for location identification and awareness.
- **Security:** Like any networked distributed system, sCPS will have to face the security and privacy challenges. The integration with fog and cloud computing further magnify this requirement. The platform should provide security mechanisms to protect all integrated functions of cloud computing, fog computing, and sCPS from attacks, data compromises and unauthorized access.
- **Multiple applications and users support:** Fog and cloud nodes and the services they provide will be accessed and used concurrently by many sCPS applications owned by different users or clients. The platform should provide control and protection mechanisms to separate these applications and users to ensure independent operations of different applications and complete isolation of users' access.
- **Management:** Cloud and fog integrated sCPS applications are usually highly distributed spanning large areas, include high volume of components and services, and incorporate various types of resources. Therefore, the platform should provide some mechanisms to easily and efficiently manage the whole environment.
- **Integration with other systems:** Many cloud and fog integrated sCPS applications cannot work in isolation. They need to be integrated with other systems such as wireless sensor networks (WSN), enterprise systems, and privately-owned computing resources. The platform should provide the right tools and methods to facilitate integrating the sCPS applications with external systems.

IV. PsCPS ARCHITECTURE AND FUNCTIONS

PsCPS is designed to support the implementation and operations of cloud and fog integrated sCPS applications. PsCPS is a distributed architecture that consists of a PsCPS manager, multiple node platforms (nodePlatform) installed on fog, cloud, and sCPS nodes with compute capabilities. Each installed nodePlatform offers an execution environment for software agents that implement different sCPS, fog and cloud functions. These agents will provide services to other agents and use services provided by other agents. The PsCPS Manager provides overall control for the whole environment including loading, deploying, suspending, and removing agents for all nodePlatforms. There are also different types of software agents in PsCSP including single agents, multi-agent systems, and hierarchical multi-agent systems.

A. NODEPLATFORM

Each fog node in the environment is equipped with a nodePlatform to be part of the PsCPS. In addition, some nodePlatforms can be available on some sCPS nodes with compute components and a connected cloud node or multiple connected cloud nodes based on the characteristics and requirements of the sCPS application. The main function of the nodePlatform is to provide a runtime environment to securely execute different agents' services on the corresponding nodes and to enable communication among different components, agents and services within and outside the node. Each nodePlatform needs to be started to enable the corresponding fog, cloud, or sCPS node to be integrated with the environment. As soon as any nodePlatform is started it sends a secure login request to the PsCPS Manager. If this request is accepted, the nodePlatform starts to download the needed system agents and application agents to support different environment and applications functions.

The nodePlatforms in PsCPS are identified by unique nodePlatform IDs. These IDs can be flat, hierarchical or a combination of both. Each nodePlatform ID is associated with the node IP address. Flat nodePlatform IDs can be used when there are a few fog nodes in the environment and they have the same functionality. An example of flat nodePlatform IDs can be SBuilding1, SBuilding2, SBuilding3, and SBuilding4, where a single fog node is responsible for one of the four smart buildings in the system. Hierarchical nodePlatform IDs can be used to associate a specific set of nodes to a specific area or to identify a related group of nodes together. An example of hierarchical nodePlatform IDs is N1.SBuilding1, N1.SBuilding2, N1.SBuilding3, N2.SBuilding1, and N2.SBuilding2. Here we have N1 as the ID of the first neighborhood and N2 as the ID of the second. Each neighborhood may have multipole smart buildings, hence the SBuildingx IDs in the next level. As we can see from the example, there are two neighborhoods each having multiple smart buildings and each smart building has a nodePlatform. A combination set of nodePlatform IDs can be N1, N2, N1.SBuilding1, N1.SBuilding2, N1.SBuilding3, N2.SBuilding1, and N2.SBuilding2. In this example, in addition to the five nodePlatforms responsible for the five smart buildings, there are two more nodePlatforms solely responsible for two neighborhood, N1 and N2. In addition, each

nodePlatform can have more than one unique ID. This type of identification can provide features to easily manage the whole PsCPS environment for different applications.

Each nodePlatform has a Node Service Broker (NSB) and several system and application agents. The system agents provide functions and services to support the system operations of the node while application agents are available to support the sCPS applications. NSB is responsible for services registration, advertisement, and discovery. The agents of a nodePlatform are either preloaded with the platform or loaded from PsCPS Manager when needed. Each agent implements some functions and can be a service provider, service consumer, or both. The provided services can be control functions for sCPS, filtering services for some communications, storage services, caching services, etc. Any service provider agent needs to register its services with the node's NSB to enable other agents to discover and access their services. An agent needs to specify the services available and the types of access to these services when registering these services. NSB registers remotely accessible services with PsCPS to be known for the whole environment.

Local client agents can access some services provided by other local and remote agents by looking up these services with the NSB that provides information to access the requested services. This access information is directly provided by the NSB for locally available services, while it forwards any request for remote services to the PsCPS Manager, which maintains access information about all available remote services on all nodePlatforms.

B. AGENTS

An agent contains programs that can be executed on nodePlatforms. Each agent implements certain functions or services for other agents within the environment including agents from other nodePlatforms. In addition, it can provide services for other programs outside the environment, to facilitate access to external systems for example. There are two kinds of agents: system agents and application agents. System agents have functions to enable the operations and management of a single nodePlatform or the whole PsCPS. The system agents can provide services for other system agents and application agents. Examples of system agents are a fog storage agent that provides services to use temporary storage in fog nodes. Another example is a caching agent that provides data caching services for other agents. While system agents generally implement functions for the nodePlatforms, application agents are developed to support application domain specific functionalities. An example of application agents is agents that provide control service (e.g. monitoring the timing of a specific traffic light) for a smart traffic light sCPS. These agents will differ from one application to another; however, they will be able to use existing services within the nodePlatforms.

One of the main differences between system agents and application agents is that while system agents can directly access the node resources such as a node's storage, operating systems, and network ports and interfaces; application agents cannot directly access these resources. However, they can access them through the available system agents or through some interfaces provided by the nodePlatform. The main reason of this restriction is to enable the nodePlatform to enforce required security and access control policies. For example, using this distinction, it will be possible for a nodePlatform to prevent an application from unauthorized access to resources and also prevent one application from accessing resources or features of another application. Agents available within the same nodePlatform or within different nodePlatforms can communicate using two mechanisms:

- Service calls, where one agent can communicate with another agent by calling a service provided by that agent. This provides one-to-one synchronized request/response communication.
- Publish/subscribe communication services, where agents can publish notifications while other agents can subscribe to receive these notifications. This provides many-to-one and one-to-many communication across multiple agents.

Agents that provide services for other agents need to register their services with the nodePlatform NSB. This is needed to allow other agents to know about the availability of these services. With the registration of these services, the agent should specify the type of access on these services. In PsCPS, access to any service provided by any agent can be restricted with four options:

- Services publicly available to all agents and other programs that have access to the environment.
- Services only available for the agents within the PsCPS environment.
- Services only available for agents within the same nodePlatform.
- Services only available for a group of agents in a multi-agent system, as discussed next.

C. MULTI-AGENT SYSTEMS (MAS)

A multi-agent system is a group of related agents that can be distributed on multiple nodePlatforms to implement certain functions for certain applications or system operations. A MAS can be controlled and managed as a single system by the PsCPS Manager and sCPS applications. Although, a MAS can consist of multiple agents that will be installed on different nodePlatforms, all of these agents can be easily managed and controlled as a single system within the whole environment. Like a single agent, which can be system agent or application agent, a MAS can also be either a system MAS, or an application MAS. Similarly, system MAS have more access to the resources of the nodes and application MAS have controlled access to the nodes resources.

MAS can be either homogenous where all agents have the same programs and perform the same type of tasks or heterogeneous where agents have different programs and perform different tasks. A homogenous example is when a MAS is used to provide the same set of services and functions on

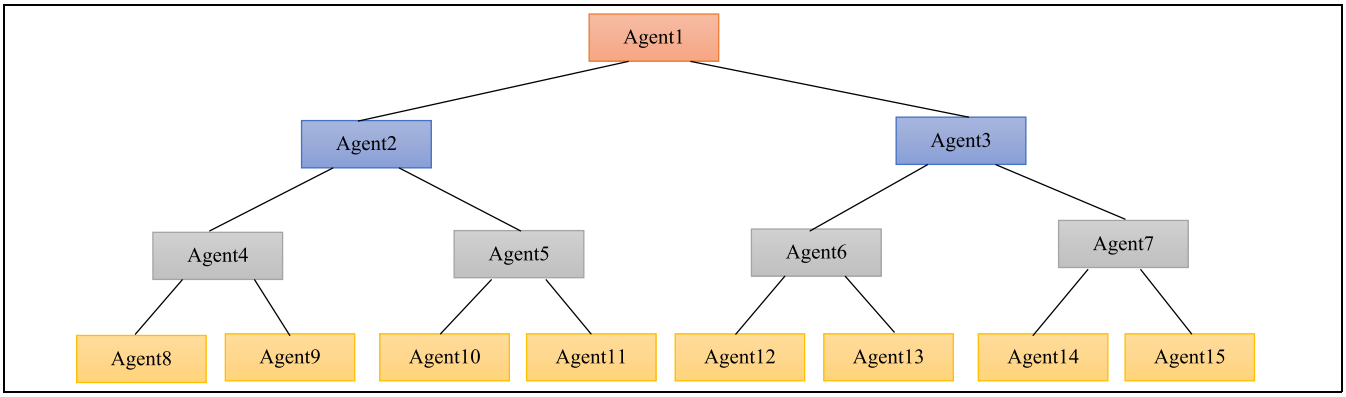

FIGURE 4. A 4-level hierarchical multi-agent system.

multiple fog nodes. In contrast, a heterogeneous MAS is used to provide services on fog nodes, compute sCPS components, and cloud nodes. For example, in a MAS, agents on the fog nodes provide communication services, agents in the sCPS components provide interfaces and controls for the sCPS application, and agents in the cloud nodes provide services to allow external applications such as web applications to access the environment.

Furthermore, MAS can be either static or dynamic. A static MAS has a fixed number of agents from the instantiation to the end of the life of the system, while a dynamic MAS can have a dynamic number of agents based on the current situation of the application. In the dynamic MAS, agents can be dynamically added and removed based on the needs of the application. For example, in an application that involves multiple mobile components such as collaborative mobile robots [13], special agents to support the mobile components such that only the nodePlatforms that are geographically close to these components will have the services related to them. In such applications, an agent is created and started on a fog nodePlatform if the application's mobile components are within its location. However, more agents will be created and started on other nodePlatforms when these mobile components move to their locations. At the same time, agents created earlier on a certain nodePlatform for these components will be removed when the mobile components move out of their locations.

While agents in a MAS can communicate through service calls and publish/subscribe services like single agents, they can also communicate directly through message passing to exchange and broadcast messages among themselves. Message passing is only allowed among agents within the same MAS. Message passing can provide fast communication among the agents of a MAS. Each agent instantiation in a multi-agent system receives a unique ID when it starts. This ID can be used within the programs of the agents to control the logical flow in the agent instantiation. In this formation, all agents in a single MAS are at the same level, allowing direct access to any and all of them.

D. HIERARCHICAL MULTI-AGENT SYSTEMS (HMAS)

A hierarchical multi-agent system is a special type of MAS, where the agents are organized in hierarchical (tree) structure across a set of nodePlatforms on multiple nodes. These nodes can be cloud, fog, and sCPS nodes. The agents in a HMAS are organized in multiple levels, where the top level can be on a cloud node or on a node that is relatively close to a cloud node. The lowest levels can be a set of sCPS compute components and fog nodes that are relatively close to the sCPS compute components. The intermediate levels can be on several fog nodes with different capabilities. The nodes in the upper intermediate levels usually have more resources compared to the nodes in the lower levels. Several agents at one of the lower or intermediate levels of the HMAS will have a parent agent at the upper level.

A HMAS can have different types of agents in the different levels. In other words, all agents in each level provide similar services but are different from the services provided by agents in the other levels. For example, a HMAS with 4 levels can have 4 different agent types as shown in Figure 4. The communication and services utilization among these agents are done through the defined hierarchy. Agent2 will communicate with its parent: Agent1, and will provide services for it, while the parent, Agent1, will also provide services for Agent2. At the same time, Agent2 will communicate with its child agents, Agent4 and Agent5, and provide services for them. The child agents, Agent4 and Agent5, will also communicate and provide services to Agent2. The root agent of a hierarchical multi-agent systems can be available in a cloud node to provide high level and complex services for several hierarchical multi-agent systems for multiple applications available on the cloud. These services can be information collection from distributed sCPS components or common configuration and control services for a large-scale sCPS. The communication and service calls in a hierarchical multi-agent system can take two directions:

- **Bottom-up**: A child agent requests a service provided by its parent agent. The parent agent can complete the service and return the result by itself, or it initiates

another service call to its parent agent. It will wait for a response to generate a response to the child agent. This process will continue until the initial request is satisfied.

- **Top-down**: A parent agent requests a service provided by its child agent. Then either the child agent can satisfy the request, or it calls its child agents. The process continues until the request is satisfied and a response is returned to the requester.

E. PsCPS MANAGER

PsCPS Manager provides overall control and management services for all PsCPS components. It keeps information about the nodePlatforms that are part of the environment. This includes their unique IDs and login information to authenticate and identify the nodePlatforms and their node. It also keeps information about the nodes types, which identify them as fog, sCPS compute components, or cloud nodes. In addition, it associates these nodes with their IP addresses if they have fixed IP address. Only nodePlatforms with valid credentials can access and connect with the PsCPS Manager. As soon as a nodePlatform is authenticated and connected, the PsCPS Manager downloads the necessary system and applications agents to the nodePlatform. Although, the nodePlatform can be preloaded with some system agents, the PsCPS Manager can provide the newly started nodePlatform with unloaded and new system agents. In addition, it can provide some application agents that are needed to provide specific functions for the applications. The PsCPS Manager maintains information about all active and connected nodePlatforms, their current IP addresses, ports, and types.

PsCPS Manager has a service broker to register available services that can be remotely accessed either by external programs or other agents in the environment. Named the Global Service Broker (GSB), it maintains information about all remotely accessible services. Furthermore, the PsCPS Manager provides services to instantiate agents in different nodePlatforms including:

- **Agent as a Service (AGENTaaS):** A service that enables users, administrators or programs to load, start, integrate, monitor, control, and remove an instance of a single agent in a specific nodePlatform. We will use arrow "→" to formally specify loading and starting an agent on a nodePlatform. Examples are Agent1 → FogNode1, indicating that an instance of Agent1 is loaded and started on the nodePlatform on FogNode1.
- **MAS as a Service (MASaaS):**A service that enables users, administrators, or programs to load, start, integrate, monitor, control, and remove instances of the agents in a MAS in a selected group of nodePlatforms. This service can basically use multiple requests for AGENTaaS to load and start multiple agents on multiple nodePlatforms. An example of using this service can be starting a static heterogeneous MAS named MAS1 with requests: Agent1 → CloudNode and Agent2 → FogNode1, FogNode2, and FogNode3. In this example, we are instantiating four agents of two different types. An Agent1 instance is loaded and started on the nodePlatform on CloudNode, while three instances of Agent2 are loaded and started on the nodePlatforms on FogNode1, FogNode2, and FogNode3. Another example of using this service is to load and start a dynamic heterogeneous MAS named MAS2 with the requests: Agent1→ CloudNode and Agent2 → CloudNode.*. Here, an instance of Agent1 is loaded started on the nodePlatform of CloudNode and an instance of Agent2 is loaded and started on each connected nodePlatform with the 2-level hierarchical ID that starts with CloudNode like CloudNode.FogNode1, CloudNode.FogNode2, etc. If a new nodePlatform with ID CloudNode.FogNode9 is started and connected later, the system will load and start an instance of Agent2 on CloudNode.FogNode9 and add it to MAS2.
- **HMAS as a Service (HMASaaS):** A service that enables users, administrators, or programs to load, start, integrate, monitor, control, and remove an instance of a HMAS on a selected group of hierarchical nodePlatforms. This service can also use AGENTaaS to instantiate multiple agents in multiple nodes based on the defined nodes hierarchy. An example can be starting a dynamic 4-level HMAS named HMAS1 with the requests: Agent1 → CloudNode, Agent2 → CloudNode.* , Agent3 → CloudNode._.*, and Agent4 → CloudNode._._.*. Like in the MASaaS, if a new nodePlatform with an ID that fits one of the IDs defined in the hierarchy (e.g. CloudNode. FogNode13.FogNode24 or CloudNode.FogNode4. FogNode25.sCPSNode7) is started and connected later, the system will load and start an instance of the required agent at the proper location in the hierarchy and add it to HMAS1.

The PsCPS Manager maintains information including the status and locations of all instantiated agents, MAS, and HMAS in the environment. In addition, it handles adding and removing agents to dynamic MAS and HMAS as needed.

V. PsCPS IMPLEMENTATION

The PsCPS can be implemented in different ways. However, for this paper we experimented with the features of the proposed platform through a prototype implementation using Java. We selected Java due to its portability, performance, security, network programming, and reflection advanced features. In this section, we discuss the implementation of the agents, MAS, and HMAS. We also discuss the nodePlatform and PsCPS Manager implementations in addition to the implementation of message passing and publish/subscribe notification services.

A. AGENTS IMPLEMENTATION

Agents are completely implemented in Java including agents that provide local and remote services. These services are provided in the form of public Java methods that can be accessed by Remote Method Innovation (RMI) mechanisms [22]. Different agents' classes can be implemented by extending

the sCPSAgent class. This class provides an extra layer on top of RMI to add PsCPS features that are not available in the original implementation of RMI. Examples of these features are the support for different security mechanisms, location awareness, MAS, HMAS, and interfaces connecting with the corresponding nodePlatform. This layer deals with NSB and GSB to provide the extra value-added services in addition to services already provided by the RMI registry. Examples of the methods provided by this layer are:

- `NodePlatformIDs[]getNodePlatformID()` to retrieve nodePlatforms' IDs list
- `void bind(String AgentName, Object obj)` to register an agent object that provides services for other agents available locally or remotely.
- `boolean secureService(Object obj, String methodName, int securityLevel)` to define a security feature for accessing different services provided by an agent.
- `Remote lookup(String AgentName)` to search for an agent with a specified name and return its stub's reference.
- `Remote lookup(String AgentName, NodePlatformID nodeID)` to search for a remote agent with a specified name at a specified nodePlatform ID and return its stub's reference.
- `int agentType()` to check the type of agent. This method returns 0 for single agent deployment, 1 for MAS deployment, and 2 for HMAS deployment.
- `int hierarchicalLevel()` to return the level of the agent in a HMAS. It will always return 0 if the deployment is a single agent or a MAS.
- `Remote getParent()` to get the stub's reference of the parent agent in a HMAS.
- `int getNoChildren()` to retrieve the number of child agents associated with an agent.
- `Remote getChild(int childNo)` to get the stub's reference of the child agent number childNo in a HMAS.

Different agents that provide diverse services for accessing CPS devices can be also developed and deployed in the environment. These services can be used by other agents in PsCPS to access different CPS devices. For our prototype implementation and for CPS side, we used the Arduino board [23] which is open source hardware for embedded systems. For this prototype, the Arduino was used as the CPS payload subsystem that is the onboard device requesting services. Some sensors were connected to the Arduino such as DHT11 sensor [24] for temperature and humidity measurements. Furthermore, some LEDs and a buzz were installed to represent actuators. In addition, we installed an Adafruit CC3000 Wi-Fi board [25] to connect the Arduino to a local area network that has a fog node. We developed an agent, named the ArduinoAgent, to provide access services to the Arduino devices. For this agent, we used the Arduino IDE [26] with the Adafruit CC3000 library [27] to develop it.

B. NODEPLATFORM IMPLEMENTATION

The main functions of a nodePlatform are to deploy, schedule, and support the execution of different agents in a fog, cloud, or sCPS node. When a nodePlatform receives a request to deploy an agent from the PsCPS Manager, the nodePlatform performs the following tasks:

1) If the agent' programs, their stubs and skeletons are not available on the nodePlatform, it requests them from the PsCPS Manager.
2) The nodePlatform starts the agent and registers all available services to the local NSB and registers all remotely accessible services with the GSB available with the PsCPS Manager.
3) The nodePlatform enables local and remote clients to use the available registered service.

For high throughput, the nodePlatforms are designed to be multithreaded, where each thread serves a client's service request. Each nodePlatform consists of several components that implement different functions:

1) The Agent Classes Loader retrieves specific agent classes' code from the PsCPS Manager.
2) The Request Manager handles client requests to call a service provided by a local agent in the nodePlatform.
3) The Resource Manager provides methods to manage, schedule, and maintain the resources of the machine where the nodePlatform resides. It keeps records of executing threads, machine and communication resources' utilization, and performance information. In addition, it is responsible for reclaiming system resources after each service's completion or termination.
4) The Security Manager provides security measures for the system as discussed next.

Executing user services on remote fog, cloud, and sCPS nodes exposes these nodes to many "alien" threats, raising security and integrity worries. Therefore, these machines must be protected to ensure safe execution. Java's default security manager offers some security mechanisms by checking execution requests against certain defined security policies before execution. However, the security manager in Java has some limitations, thus additional security features were added to our nodePlatforms. More specifically, two modes of operation are used to offer a secure and reliable execution environment:

1) The System Mode, in which no limitations are enforced. This mode is used only by the system agents to enable full access and control of all the operations, services, and resources in the corresponding node. This is necessary as system agents need to provide access to resources and services to other agents.
2) The Application Mode, in which limitations are enforced to limit users' access to the node operations, services, and resources. This mode is used for application agents. Some operations, such as removing files, initiating a process, using system calls, and changing system properties are inactive. However, application

agents can use services provided by system agents to get some node related services when needed and with proper authentication.

The security manager also ensures the proper execution of services within the restrictions required. Thus, it will provide full access to public services, and restrict access to other types of services depending on their availability and access policies. For example, some services are only available for agents on the same nodePlatform, some are available for agents within the same MAS or HMAS, while others are available to agents within one PsCPS environment.

C. PsCPS MANAGER IMPLEMENTATION

The PsCPS Manager was also implemented in Java as a multithreaded server. The threads allow the PsCPS to process requests from both external user programs, nodePlatforms, and system administrators. The PsCPS Manager consists of several components that implement different functions:

1) Agents' Repository stores all classes' codes of all agents, stubs, and skeletons used in the environment.
2) Agents Manager handles the operations of AGENTaaS, MASaaS and HMASaaS.
3) Global Service Registry handles services' registration and lookup processes.
4) Security Manager handles login processes and access monitoring for user programs, nodePlatforms, and system administrators.

AGENTaaS, MASaaS and HMASaaS are implemented as Java classes. These classes can be used as part of Java programs to instantiate single agents and agents' groups on single or multiple nodePlatforms. Examples of using these classes are:

- `AGENTaaS a = new AGENTaaS("Agent1", NodePlatformID)` to instantiate an Agent1 object on the nodePlatform with ID NodePlatformID.
- `MASaaS mas = new MASaaS(0, Agents, NodePlatformIDs)` to instantiate a static MAS object. Agents is an array of agent objects and NodePlatformIDs is an array of nodePlatform IDs where the agents will be instantiated. For example, if Agents = {"Agent1", "Agent2", "Agent2", "Agent2"} and NodePlatformIDs = {"CloudNode", "FogNode1", "FogNode2", "FogNode3"}. This will instantiate an instance of Agent1 object on CloudNode and 3 instances of Agent2 object one on each of FogNode1, FogNode2, and FogNode3.
- `MASaaS mas = new MASaaS(1, Agents, NodePlatformIDs)` to instantiate a dynamic MAS object where an array of the agents is defined in Agents and an array of nodePlatform IDs is in NodePlatformIDs. Here, if Agents[1] = "Agent2" and NodePlatformIDs[1] = "CloudNode.*" then an object of Agent2 will be instantiated on any connected nodePlatform with an ID matching "CloudNode.*" as soon as it starts. The new agent will also be added to the defined MAS.
- `HMASaaS hmas = new HMASaaS(0, Agents, NodePlatformIDs)` to instantiate a static HMAS object, where an array of agents is defined in Agents and an array of nodePlatform IDs is in NodePlatformIDs. For example, Agents = {"Agent1", "Agent2", "Agent3"} and NodePlatformIDs = {"CloudNode", "CloudNode.*", "CloudNode._.*"}, will result in creating a static HMAS object of 3 levels. In the first level, an object of Agent1 is instantiated on CloudNode, several objects of Agent2 are instantiated on all active and connected nodePlatforms with IDs matching "CloudNode.*", and several objects of Agent3 are instantiated on all active nodePlatforms with IDs matching "CloudNode._.*".

There are several additional methods in these classes that support managing and controlling the created single agents, MAS, and HMAS. These include methods to suspend, remove, or query the agents.

D. MESSAGE PASSING SERVICES

A facility for message passing services was provided as a system MAS, where its agents are deployed on the nodePlatforms that need to use message passing services in the PsCPS. The agents of this MAS are developed to provide all services and supporting functions for message passing operations. We used the Java Object Passing Interface (JOPI) [14], [15] to offer the message passing services for the PsCPS. JOPI provides an MPI-like (Message Passing Interface) interface that can be used to exchange objects among distributed agents. JOPI provides Java programmers with the necessary services to write object-passing distributed and parallel programs in distributed systems. JOPI provides point-to-point blocking (synchronous) communication services, non-blocking communication services, broadcast communication services, and a group synchronization service. All these services can be used for distributed and parallel operations for other application and system related MAS. This allows for utilizing multiple closely located nodePlatforms to achieve some distributed and parallel tasks needed for some applications.

E. PUBLISH/SUBSCRIBE COMMUNICATION SERVICES

Publish/subscribe communication services are implemented in PsCPS either by a single agent, MAS or HMAS. In a small environment with a few nodes, it is usually enough and efficient to use a single agent, while in a large-scale environment with many nodes, it is more efficient to use a HMAS. The implementation of the publish/subscribe services in a small system using a single agent will utilize one designated agent as a manager of the services. The managing agent is responsible for keeping track of notifications, subscriptions and publication. This agent should reside on a powerful node such as a cloud node to provide these services for all other agents in the system. Other agents in the system will contact the managing agent to subscribe to be notified for the occurrence of a specific event, unsubscribe from previous subscriptions, and publish a notification of the occurrence of

a specific event to interested agents. When a certain event occurs, the managing agent finds and notifies all agents who subscribed for that event.

In large-scale systems a single agent will not be able to efficiently manage these services. Therefore, it is more efficient to use HMAS to achieve better performance. In such system, multiple homogenous agents can be used in the PsCPS to provide distributed management for the notification, subscription and publication services. All these agents will provide the needed services to subscribe to be notified for the occurrence of a specific event, to unsubscribe from previous subscriptions, and to publish a notification of the occurrence of a specific event. These services are the same as those provided by the single agent solution. However, in the hierarchy, these services are provided only for the agent's parent and for the agent's children. If the subscription request comes to AgentX from a child agent, then AgentX will subscribe for that notification with its other child agents and with its parent agent. However, if the subscription request comes to AgentX from its parent agent, then AgentX will subscribe for that notification with all its child agents. This process will progressively continue in the whole environment. A notification for an event will only be published through the hierarchy and only to the subscribed agents. If an agent in this hierarchy receives a notification from its parent agent, then it will forward this notification to only its child agents that are already subscribed to receive that notification. If an agent receives a notification from one of its child agents, it will forward this notification to its parent agent and other child agents if and only if they subscribed to receive this notification.

Using HMAS for publish/subscribe services provides many advantages. One of these advantages is that a notification for an event will be sent over the network only to agents that need it. Another advantage is that the notification process can be faster and more scalable as the hierarchical structure is used to publish the notifications. Furthermore, more optimizations can be added to the publish/subscribe system that can further enhance the services performance by taking advantage of the decentralized management approach. One possible optimization is the ability to add a location attribute to subscriptions to receive notifications of an even only if it occurs at a specific location or only if the subscribing agent is at a specific location. For example, there is an interest to know about a certain event if it occurred in a specific building (e.g. a manufacturing facility, a hospital, or a school). In this case, there is no need to subscribe everywhere with all agents. Only agents residing on nodePlatforms at these locations will subscribe and get notified, while agents in other locations do not need to be part of this.

VI. APPLICATION SENARIOS

This section demonstrates how to utilize PsCPS for some sCPS applications. Using two examples from smart buildings and smart traffic lights systems, we show how the agents can be developed and deployed to assist in integrating specific applications with services on fog and cloud nodes for best possible results.

A. CLOUD-BASED FAULT DETECTION AND DIAGNOSIS IN SMART BUILDINGS

This example demonstrates how to utilize PsCPS for enhancing energy efficiency in smart building applications. The example application is a collaborative cloud-based fault detection and diagnosis in smart buildings. Accurate and timely detection and diagnosis of faults in Building Energy Management Systems (BEMS), which are CPS has the prospective to save 15 to 30 percent of the overall building energy consumption. Finding and learning about faults in a single building can take significant time and can be a very costly process. However, utilizing experiences from a number of similar buildings and smart cloud services for fault detection and diagnosis can reduce the time and cost needed to find the common faults in these buildings [18].

PsCPS can be used to enable developing and operating a collaborative cloud-based fault detection and diagnosis application for similar smart buildings. Each building has its own BEMS that connects to all devices in the building. Within the PsCPS context each of these agents will be active on a nodePlatform for a building. This may be a fog node nearby or a computing facility within the building. All similar buildings within a city will have similar nodePlatforms and agents representing the buildings' BEMS. These can be connected and integrated with the Cloud using PsCPS and the services its agents offer. PsCPS will integrate all services provided by the buildings BEMS in the smart buildings and the services provided by the Cloud. Two types of agents can be used for this application: cloud agent and building agent. The cloud agent will provide services to store, validate, process, and learn from the collected information from the participating buildings. This will enable the discovery of unknown common faults faster based on the collected data from multiple buildings. This agent will utilize services available on the cloud such as high-performance processing, scalable data storage, machine learning, data mining, and simulation services. Building agents provide services to enable collecting data from the BEMSs and configure the BEMS with new rules to appropriately react to the discovered faults in the energy systems of these buildings. Depending on the number of buildings covered, the building agents can be configured as single agents or as MAS. PsCPS will connect all agents and allow for adding more services to facilitate the operations in the smart buildings and seamless integration with cloud services. More information about this application are in [19].

B. SMART TRAFFIC LIGHTS

The second example is using PsCPS for implementing smart traffic lights in a city. PsCPS can be used to enable smart traffic light controls, which include monitoring devices across multiple locations to accurately determine current traffic conditions and software models to predict possible traffic patterns. This information is then used to adjust traffic lights

to optimize flow. This type of application can also benefit from global positioning systems and vehicle-to-vehicle and vehicle-to-infrastructure communication systems to collect a more detailed view of the traffic situation. The collected information can be used to achieve traffic flow optimizations at different levels within a city including, a single intersection, a neighborhood, a community, or the whole city. Learning and adaptation algorithms are usually used in such applications for decision making and optimizations [20], [21].

Smart traffic light controls can be developed and operated using PsCPS. A hierarchical fog computing structure with a HMAS can be used to provide controls and optimizations at different levels in the whole city. This HMAS has different types of agents that provide different services for controlling and optimizing city traffic:

- Traffic light agents (TLA), where each agent provides services to monitor, control and optimize traffic in one intersection.
- Neighborhood traffic agents (NTA), where each agent provides services to control and optimize traffic in one neighborhood through its direct connections with all TLAs within that neighborhood.
- Community traffic agents (CTA), where each agent provides services to control and optimize traffic in one community. As a community is comprised of multiple neighborhoods, the hierarchy allows a CTA to directly connect with the NTAs in its area. This also creates the indirect links to all TLAs in all neighborhoods involved.
- A city traffic agent that is available on a cloud node to provide services to control and optimize traffic in the city. This represents the top level of the hierarchy as it will connect to all CTAs in the city.

The hierarchy looks a lot like the example in figure 4, which can be easily interpreted as Agent1 being the city traffic agent, Agent 2 and Agent 3 are CTAs and each has two neighborhoods represented by two NTAs and each NTA connects to two TLAs in the lowest level. All these agents work together enabling the PsCPS to reduce traffic delays, minimize vehicles travel times, increase vehicles average velocity, and enhance the prioritization for emergency vehicles' movements in the city area. For example, in normal traffic conditions, each TLA will deploy a local traffic light sequencing policy and may also locally adjust it as traffic patterns fluctuate lightly. Information about flow patterns are sent to the corresponding NTA, which monitors conditions across all traffic lights within the neighborhood. If traffic flow changes significantly, the NTA will notify all TLAs in the neighborhood to adjust their light sequences to help reduce congestion. In addition, it will notify its CTA about the changes and wait for further instructions. The CTA, aware of traffic situations across all neighborhoods, can make more informed decisions about how to handle the traffic. In addition, it will consult with the top level, the city traffic agent, to utilize the available intelligent optimization techniques to find the best way to handle the traffic across the community. Another example could be when emergency vehicles need to travel a certain path from one part of a city to another. In this case, the city traffic agent will determine the optimal route, notify the CTAs with a new traffic lights sequences to be propagated to all NTAs, which in turn will send them to their TLAs to ensure the emergency vehicles will not hit any red lights as they travel and also to quickly clear the path ahead of them.

VII. RESULTS AND DISCUSSION

To evaluate the performance of the proposed distributed platform, PsCPS, several sets of experiments were conducted using the prototype implementation. We used several computers to represent cloud nodes, fog nodes, and sCPS nodes. Furthermore, we used the Arduino board as a compute sCPS component. In addition, we used a wide area network (WAN) emulator to connect the cloud nodes with fog nodes to include the effects of WAN such as high return trip time (RTT), limited bandwidth, and packet drops.

The first set of experiments was conducted to find the response time of using services provided by devices connected to the Arduino. Here, we used three computers; one as a cloud node and the other two as fog nodes. We installed three nodePlatforms on the nodes and used the WAN emulator to connect them. Experiments were conducted for a local sCPS service call (SC1) within the corresponding fog nodePlatform (from a local client agent to ArduinoAgent), a remote sCPS service call from the cloud nodePlatform (SC2) (from a client cloud agent to remote ArduinoAgent on a fog nodePlatform), and a remote sCPS service call from another fog nodePlatform (SC3) (from a client fog agent to a remote ArduinoAgent on a different fog nodePlatform). The experiments were repeated for two types of services. The first service is to get the current temperature (CurTemp()) while the second is to turn on the LED (LEDon()). The average results of 10 runs of multiple service calls are shown in Figure 5. The recoded times for these calls do not include service lookup times. The response time for a service call from fog to fog and from cloud to fog are similar as the service call is directly done between the client fog node and the server fog node without involving the cloud. The cloud is only involved during the service lookup process. After that, the service can be directly called.

The second set of experiments was conducted to test the effect of using an agent at a local fog node to provide computation services for sCPS applications. This agent provides a matrix multiplication service for sCPS applications if it can provide that service faster than the cloud; otherwise it will send the request to the cloud as the cloud can provide parallel matrix multiplication using 4 nodes.Generally, there are many computation problems that can be solved locally with limited resources if the data set is small. However, large-scale computational problems need higher system resources that can be provided by the cloud. An agent to provide a matrix multiplication service is implemented on the local fog nodePlatform, while a service that provides parallel matrix multiplication is provided by 4 powerful cloud nodes.

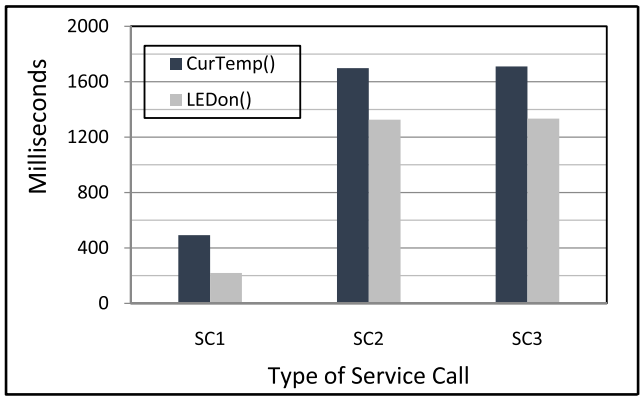

FIGURE 5. sCPS evices service call response times.

The main function of this agent is to provide integrated matrix multiplication service that will allow it to decide to perform the process locally for small matrices or forward the process request to the cloud for large matrices. Experiments were conducted using a local fog node (Fog), cloud nodes (Cloud), and both fog and cloud nodes (Integrated) for matrix multiplication. The machines used for this experiment have Intel CoreTM i5 CPUs running at 2.20 GHz and 8.00 GB memory. The average results of 10 runs are shown in Figure 6. The results are reported for square matrices of sizes 1000, 1500, 2000, 2500, and 3000. The fog node provides good performance for small computational problems (e.g. matrix size 1500 or smaller); however, the cloud provides better results for larger problems. The integrated solution provided by the developed agent can combine the advantages of both in a single solution.

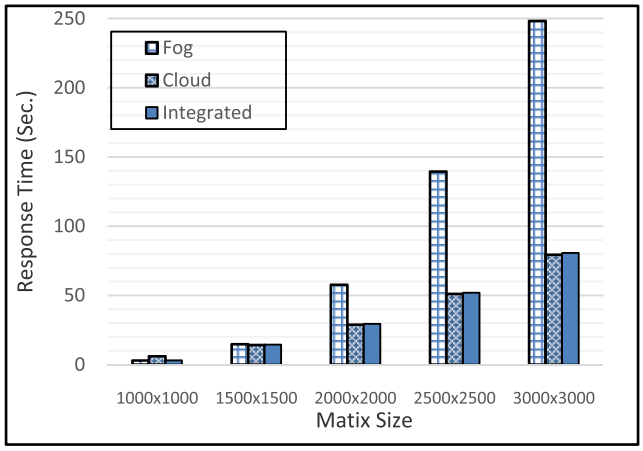

FIGURE 6. fog, cloud, and integrated matrix multiplication performance.

The third set of experiments was conducted to test the performance of the MAS in PsCPS. If there are multiple fog nodes available within the same area, they can collaborate on some tasks. For example, if we have 3 fog nodes in one area all 3 nodes can provide local parallel matrix multiplication. To demonstrate this approach with PsCPS, a MAS that provides parallel matrix multiplication was developed using PsCPS. Direct message passing mechanism using JOPI was used. The same problem used in the second set of experiments is used. However, now we have the MAS that provides parallel matrix multiplication on multiple fog nodes instead of a single fog node. Figure 7 shows that with this approach, matrix multiplication with size less than 2500 can be achieved faster locally using multiple fog nodes compared to less than 1500 square matrices using a single fog node.

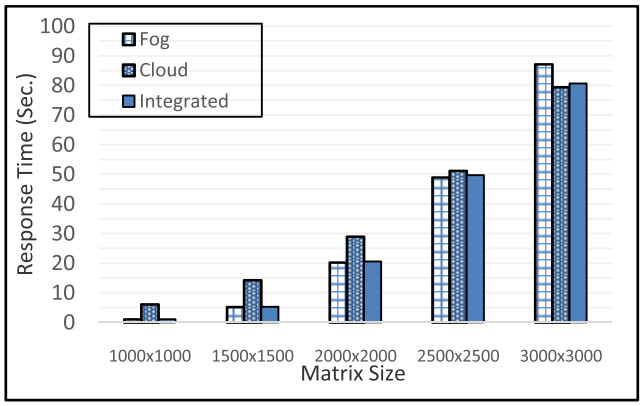

FIGURE 7. Multi-agent fog-based and cloud matrix multiplication performance.

The fourth set of experiments was conducted to investigate the benefits of using PsCPS for one of the important smart environments, smart buildings. We selected this smart environment as cloud computing can provide many services and advantages for smart buildings to enhance energy efficiency, reduce carbon emissions, and enhance quality of living for the occupants [4], [28]. PsCPS can be used to implement different integration models between cloud and fog computing and smart buildings. We will evaluate possible integration models in terms of communication traffic and delay. These models are:

- 1-Level Model: Cloud computing is used to provide BEMS services. All sensors and actuators of smart buildings are connected directly to the cloud.
- 2-Level Model: In this model, the top level is the cloud to provide advanced services needing powerful resources such as fault detection, while the second level is the fog where each fog node provides other BEMS services for each smart building. These services include communication and data aggregation services for the collected information needed at the cloud level.
- 3-Level Model: In this model, the top level is the cloud to provide advanced services needing powerful resources such as fault detection, the next level is a fog level where each fog node provides some communication and data aggregation services to link smart buildings in a community with the cloud, while the lower level is also a fog level where fog nodes provide other BEMS services like communication and data aggregation for individual building.

To evaluate these models, we used PsCPS to develop a single agent solution for 1-Level Model, a 2-level HMAS for the 2-Level Model, and a 3-level HMAS for the 3-Level Model. We also developed a smart building software emulator with some functions for sensors and actuators to control different devices. We instantiated multiple smart buildings and linked them using the integration models. We used WAN emulators between the smart buildings and the cloud for the 1-Level Model, between the building fog level and the cloud for the 2-Level Model, and between the community fog level and the cloud for the 3-Level Model.

For these experiments, we considered schools as smart buildings and two types of applications. The first application is that each room in a smart school building is equipped with a temperature sensor to monitor the temperature and a controller that links with the classroom's schedule and the number of students in each classroom maintained on the cloud to control air-conditioning to maintain convenient temperatures in the classrooms and at the same time regulate and optimize energy consumption. The second application is cloud-based fault detection and diagnosis (as discussed in Section V1.A) to discover unknown faults in the air-conditioning control system. We configured the environment to have 5 communities, each with 4 smart school buildings. There are 20 classrooms in each school. Each temperature sensor measures and sends temperature value every one minute. We configured the system to count the number of messages that the cloud needs to receive or send for both monitoring and controlling tasks to correctly execute the closed-loop control. The results are shown in Figure 8. Both the 2-Level and 3-Level Models produce less traffic to the cloud compared to the 1-Level Model as they transfer the control process to the lower levels. Both the 2-Level and 3-Level Models use data aggregation to transfer collected information. The 3-Level Model provides the best results as it uses two levels of data aggregation, the smart building level and community level. We also measure the average control response times for all models as shown in Figure 9. As we can see both the 2-Level and 3-Level Models provide better control response times compared to the 1-Level Model. This can enhance the control processes for better energy efficiency and occupants' comfort. The 3-Level Model generally provides better results in both the number of messages and average control response time. However, this depends on the number of connected communities, schools, and classrooms, and targeted smart buildings applications.

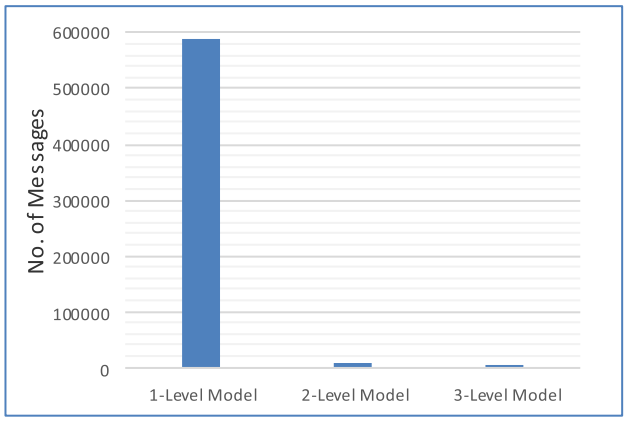

FIGURE 8. No. of messages using the different integration models.

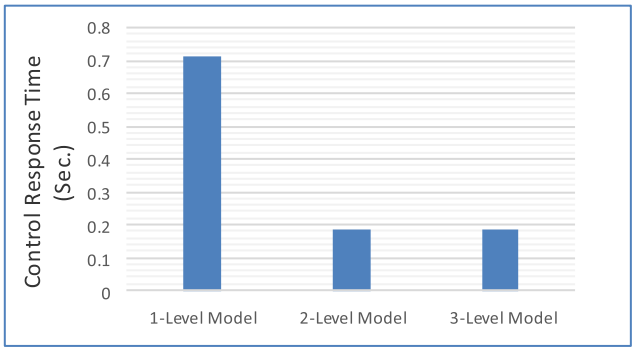

FIGURE 9. Average control response time of the different integration models.

VIII. RELATED WORK

Smart CPS (sCPS) are becoming the focus of research both in industry and research communities. From a general perspective, governments and large organizations are considering a strong move towards applying sCPS to enhance operations in various sectors. The Road2CPS in Europe, for example aims to coordinate different sCPS projects across Europe to create a collaborative environment to advance the research and development in the field with the main target of digitizing the European industry. Topics like integration of tools for safety-critical systems development, interoperability, industrial automation, and maintenance are key. In addition, there is a strong focus on validation and verification to ensure safety and security. Other areas of interest include trusted computations, high performance computing, standardization, and recommendations for future work [29]. The SEsCPS workshop outcomes also highlighted the importance of finding ways to better facilitate the design and development of sCPS applications in light of the challenges and issues faced. These include aligning software engineering principles with other disciplines like mechanical and control engineering; including humans throughout the development life cycle; handling uncertainties; and managing large-scale interdisciplinary projects [30]. The following discussion further emphasizes this direction through the discussed examples of research and development in the field.

One example of sCPS design approaches is SMART-CPS, which starts with suitable and well-defined modeling tools for sCPS such as ECML (ETRI CPS Modeling Language) to create verifiable designs for sCPS applications [31]. Service-Oriented Middleware (SOM) can also play an important role as an enabler for the design, development, deployment and interoperability of sCPS applications. In the smart city context, the SOM approach was used to facilitate the development of smart city applications as a set of services [10].

CPSWare further enhances this approach by extending the SOM model for CPS applications. This facilitates integration across different platforms and service providers such as the cloud, fog computing and built in sCPS components [32]. This approach can also be easily integrated in our proposed model, PsCPS, as the base development unit.

More research efforts have emerged recommending integrating CPS with cloud computing to incorporate the benefits of large-scale computing and storage facilities with the CPS applications. One group recommends a twin architecture reference model (C2PS) that incorporates various degrees of computation-interaction modes in such applications [33]. This model incorporates a smart context-aware interaction controller and facilitates reconfiguration of applications. Another group tackles the issue of resource management when CPS is integrated with cloud computing to increase the sustainability of such systems [34]. The article identifies bottlenecks in accessing web server capabilities, service delays, and the torrent demand as major challenges that can be managed through predictive cloud capacities when assigning tasks to heterogeneous clouds.

Unfortunately, cloud computing imposes some constraints on such integration with CPS as it faces limitations when it comes to supporting real-time responses, mobility, and location- and context-awareness [35]. As a result, some work has emerged to incorporate edge-computing (fog, mist, or dew) in these systems as highlighted in [35]. Several examples of developing mist-, dew- fog-, cloud-computing integrated CPS are being developed or proposed. In [36], Frincu propose a model to add intelligence at the edges of the network (dew-computing) to support low latency data analysis and decision making, while keeping a centralized control at the cloud level. An efficient broker system to support publish/subscribe systems for large-scale CPS that integrates fog- and cloud-based data delivery model using a hierarchy of brokers across the environment is proposed in [37]. IFCIoT (Integrated Fog Cloud IoT) proposes an architecture to integrate clusters of IoT or CPS instances through fog computing, which are then integrated with the cloud for efficient management of locality, data collection and processing. It uses energy-efficient adaptive layered fog node architecture to facilitate abstraction and implementation and is illustrated using an intelligent transportation system [38].

In addition to these general approaches being proposed, work in specific application domains of sCPS has emerged. Many offer approaches, models and designs to integrate fog and cloud computing with application-specific CPS to achieve some level of smartness. In healthcare, work is underway to integrate fog computing with cloud-based IoT and CPS applications to overcome the limitations imposed by location, heterogeneity and uneven data loads. Examples include work in [39], where the fog, cloud and IoT integration issues and orchestration are studied and solutions are proposed and [40], where fog computing is integrated with the system as a way to overcome the instability and delay issues encountered when cloud computing is used alone. Smart transportation systems are also an area where CPS plays a vital role and [41] offers a detailed discussion on how cloud computing can be integrated for better performance. In the area of smart buildings, an example in [37] discusses a fog architecture to integrate CPS with fog nodes to account for human preferences as part of the building's system control and decision-making process. This approach allows for negotiation and conflict resolution for optimized energy management in smart buildings. Stack4Things [42] offers a framework for smart city CPS that migrates some of the control and processing from the cloud to fog nodes to allow developers to manage sCPS applications and scatter application logic closer to the CPS components. Another area where CPS becomes useful is the food industry as illustrated in [43], where CPS and fog computing are used together to facilitate value stream-based food traceability sCPS. In addition, an approach is proposed for developing food traceability systems using IoT, cloud and fog computing to track and control food products' conditions, quality and location [44].

As a major driver, manufacturing has also become a large domain where several approaches were introduced to incorporate cloud-computing, fog-computing or both into industrial and manufacturing sCPS applications. In this domain, it has been shown that such large-scale integration faces multiple challenges. For example, large amounts of data, security, privacy, reliability, scalability, flexibility, and sustainability are key concerns. However, the complexity also increases when we consider the priorities of these challenges with respect to specific applications [45]. In [46] a fog-enabled architecture is proposed for data collection, self-diagnosis and fault detection of manufacturing systems as a way to leverage some of the cloud-based CPS systems. de Brito *et al.* [47] also propose bringing analytics and processing features closer to the CPS in smart-factories using fog nodes through extending the machine-to-machine communication architecture using container-based orchestration. A discussion of how CPS and cloud computing can be integrated to optimize and enhance maintenance activities is available in [48].

As discussed above, various ways and models have been investigated and proposed to integrate one or more of the modern technologies in CPS, the cloud and fog computing to create more efficient and better ways to develop, operate and manage sCPS applications. The approaches discussed usually focus on a limited number of aspects or challenges, while many offer application domain-specific frameworks. In most cases, fog computing is introduced as an additional layer between the CPS and the cloud to leverage some of the constraints. However, the majority do not offer a complete platform for generic design and development of different sCPS applications. In addition, some proposed a hierarchical structure for their models; yet most are predefined within a specific application domain or a specific sCPS application. Our proposed PsCPS incorporates a hierarchical organization for the CPS, fog and cloud components that facilitates better communication and distribution of tasks as part of a generic

development and deployment platform. In our work, we focus on these issues and offer a generic platform, where multiple nodes in the system can assume different roles and collaborate using efficient distribution and communication mechanisms through their hierarchical structure. Moreover, PsCPS offers a structure and development environment, where different applications can be built by integrating sCPS, cloud and fog services and can be deployed across the whole environment.

IX. CONCLUSION

Smart CPS (sCPS) incorporate the basic elements and functionality of CPS with intelligent software features. These intelligent features enable CPS to make smart decisions through various algorithms and methods in analytics, self-x capabilities, smart control, machine learning and artificial intelligence. Therefore, it becomes necessary to have sufficient and sophisticated hardware resources that will enable these functionalities effectively and efficiently. Using the cloud as the support environment for sCPS provides many of these needed resources such as high-performance computing, large-scale storage capabilities, and distributed and parallel computations. However, the very nature of sCPS dictates certain constraints on the system such as timely responses, location and context awareness and mobility support. The cloud, being far from the sCPS components, may not be capable of efficiently satisfying these needs. The imposed latency, unstable connectivity, burstiness of data and loads cold lead to undesired effects on the sCPS performance. As a result, work has been directed towards integrating fog computing with these systems to overcome some of the limitations.

Our proposed platform, PsCPS, offers a framework to support this type of integration, thus enabling the development and deployment of complete sCPS applications utilizing the most suitable resources from all available resources within the sCPS components, on fog nodes and on the cloud. Unlike many other approaches, PsCPS offers a generic agent-based platform that facilitates the development and deployment of different sCPS applications. The PsCPS platform consists of nodePlatforms that can be initiated and deployed on any compute node available in the environment. Agents are instantiated and operate on these nodePlatforms to offer control and processing capabilities as needed by the applications. Furthermore, these agents can be organized in different ways based on the size and organization of the sCPS and its needs for recourses. For a small sCPS environment, a collection of single agents can be used to handle the required tasks of the application. However, as the environment grows bigger and more distributed, more nodePlatforms and accordingly more agents are deployed. In this case the agents are treated as a MAS where these agents collaborate together to achieve the sCPS application's objectives. Yet, as sCPS further integrates several levels of fog and cloud nodes, organization can shift to a HMAS where several levels of agents are deployed with specific tasks assigned at each level. This organization optimizes operations and allow for better ways to exchange information and perform tasks.

PsCPS allows for the development and integration of application services with platform services to create a complete sCPS application using all available resources on the fog and cloud nodes. Therefore, it becomes easier to distribute different tasks to where they can be best performed, optimize localized services such as mobility support and location awareness at the sCPS and fog levels, and utilize high performance resources when needed at the cloud level. PsCPS offers different communication models such as direct communication, message passing and publish/subscribe models to enhance information exchange across the different layers of the system. The prototype implementation and experimental results show good performance in terms of response times, processing performance, and data exchange. As future work, we plan to further extend the implementation of PsCPS to support more features, provide complete APIs for the platform for sCPS applications developers, and perform more complex measurements using examples from different sCPS applications to further illustrate the capabilities and functionalities of PsCPS.

REFERENCES

[1] E. A. Lee, "Cyber physical systems: Design challenges," in *Proc. 11th IEEE Int. Symp. Object Compon.-Oriented Real-Time Distrib. Comput. (ISORC)*, May. 2008, pp. 363–369.
[2] R. R. Rajkumar, L. Sha, I. Lee, and J. Stankovic, "Cyber-physical systems: The next computing revolution," in *Proc. ACM 47th Design Automat. Conf.*, Jun. 2010, pp. 731–736.
[3] B. Kehoe, S. Patil, P. Abbeel, and K. Goldberg, "A survey of research on cloud robotics and automation," *IEEE Trans. Autom. Sci. Eng.*, vol. 12, no. 2, pp. 398–409, Apr. 2015.
[4] N. Mohamed, J. Al-Jaroodi, and S. Lazarova-Molnar, "Energy cloud: Services for smart buildings," in *Sustainable Cloud and Energy Services*. Cham, Switzerland: Springer, 2018, pp. 117–134.
[5] C. Doukas, T. Pliakas, and I. Maglogiannis, "Mobile healthcare information management utilizing cloud computing and Android OS," in *Proc. Annu. Int. Conf. IEEE Eng. Med. Biol. Soc. (EMBC)*, Aug. 2010, pp. 1037–1040.
[6] Z. Li, C. Chen, and K. Wang, "Cloud computing for agent-based urban transportation systems," *IEEE Intell. Syst.*, vol. 26, no. 1, pp. 73–79, Jan./Feb. 2011.
[7] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog computing and its role in the Internet of Things," in *Proc. ACM 1st Ed. Mcc Workshop Mobile Cloud Comput.*, Aug. 2012, pp. 13–16.
[8] A. M. Rahmani *et al.*, "Exploiting smart e-health gateways at the edge of healthcare Internet-of-Things: A fog computing approach," *Future Gener. Comput. Syst.*, vol. 78, pp. 641–658, Jan. 2017.
[9] M. A. Al Faruque and K. Vatanparvar, "Energy management-as-a-service over fog computing platform," *IEEE Internet Things J.*, vol. 3, no. 2, pp. 161–169, Apr. 2016.
[10] N. Mohamed, J. Al-Jaroodi, I. Jawhar, S. Lazarova-Molnar, and S. Mahmoud, "SmartCityWare: A service-oriented middleware for cloud and fog enabled smart city services," *IEEE Access*, vol. 5, pp. 17576–17588, 2017.
[11] B. Tang, Z. Chen, G. Hefferman, T. Wei, H. He, and Q. Yang, "A hierarchical distributed fog computing architecture for big data analysis in smart cities," in *Proc. ASE BigData SocialInform.*, 2015, Art. no. 28.
[12] L. Tong, Y. Li, and W. Gao, "A hierarchical edge cloud architecture for mobile computing," in *Proc. 35th Annu. IEEE Int. Conf. Comput. Commun. (INFOCOM)*, Apr. 2016, pp. 1–9.
[13] L. E. Parker, D. Rus, and G. S. Sukhatme, "Multiple mobile robot systems," in *Springer Handbook of Robotics*. Berlin, Germany: Springer, 2016, pp. 1335–1384.
[14] N. Mohamed, J. Al-Jaroodi, H. Jiang, and D. Swanson, "JOPI: A Java object-passing interface," in *Proc. Joint ACM-ISCOPE Conf. Java Grande*, Nov. 2002, pp. 37–45.

[15] J. Al-Jaroodi, N. Mohamed, H. Jiang, and D. Swanson, "JOPI: A Java object-passing interface," *Concurrency Comput., Pract. Exper.*, vol. 17, nos. 7–8, pp. 775–795, 2005.
[16] N. Mohamed, J. Al-Jaroodi, I. Jawhar, and S. Lazarova-Molnar, "A service-oriented middleware for building collaborative UAVs," *J. Intell. Robot. Syst.*, vol. 74, nos. 1–2, pp. 309–321, 2014.
[17] S. Hallé and B. Chaib-draa, "A collaborative driving system based on multiagent modelling and simulations," *Transp. Res. C, Emerg. Technol.*, vol. 13, no. 4, pp. 320–345, 2005.
[18] S. Lazarova-Molnar and N. Mohamed, "Collaborative data analytics for smart buildings: Opportunities and models," *Cluster Comput.*, pp. 1–13, Nov. 2017. [Online]. Available: https://link.springer.com/article/10.1007/s10586-017-1362-x
[19] S. Lazarova-Molnar and N. Mohamed, "A framework for collaborative cloud-based fault detection and diagnosis in smart buildings," in *Proc. 7th Int. Conf. Modeling, Simulation, Appl. Optim. (ICMSAO)*, Apr. 2017, pp. 1–6.
[20] I. Arel, C. Liu, T. Urbanik, and A. G. Kohls, "Reinforcement learning-based multi-agent system for network traffic signal control," *IET Intell. Transp. Syst.*, vol. 4, no. 2, pp. 128–135, 2010.
[21] S. El-Tantawy, B. Abdulhai, and H. Abdelgawad, "Multiagent reinforcement learning for integrated network of adaptive traffic signal controllers (MARLIN-ATSC): Methodology and large-scale application on downtown Toronto," *IEEE Trans. Intell. Transp. Syst.*, vol. 14, no. 3, pp. 1140–1150, Sep. 2013.
[22] E. Pitt and K. McNiff, *Java.rmi: The Remote Method Invocation Guide*. Reading, MA, USA: Addison-Wesley, 2001.
[23] (Dec. 1, 2017). *Arduino*. [Online]. Available: https://www.arduino.cc/
[24] (Dec. 1, 2017). *DHT Sensor Library*. [Online]. Available: https://github.com/adafruit/DHT-sensor-library, viewed .
[25] (Dec. 1, 2017). *CC3000 Wi-Fi Board*. [Online]. Available: https://www.adafruit.com/products/1469
[26] (Dec. 1, 2017). *Arduino IDE*. [Online]. Available: http://arduino.cc/en/main/software
[27] (Dec. 1, 2017). *Adafruit CC3000 Library*. [Online]. Available: https://github.com/adafruit/Adafruit_CC3000_Library
[28] I. Hong, J. Byun, and S. Park, "Cloud computing-based building energy management system with ZigBee sensor network," in *Proc. IEEE 6th Int. Conf. Innov. Mobile Internet Services Ubiquitous Comput. (IMIS)*, Jul. 2012, pp. 547–551.
[29] H. Thompson, "The 'smart cyber-physical systems' cluster of EU projects," Rep. Road2CPS Clustering Commun. Event, Vienna, Austria, Event Rep., Apr. 2016.
[30] T. Bures *et al.*, "Software engineering for smart cyber-physical systems—Towards a research agenda: Report on the first international workshop on software engineering for smart CPS," *ACM SIGSOFT Softw. Eng. Notes*, vol. 40, no. 6, pp. 28–32, 2015.
[31] I. Chun, H. Lee, W. Kim, and E. Lee, "SMART-CPS: Self-managed reliable system development method for cyber-physical systems," *Int. Inf. Inst. Tokyo-Inf.*, vol. 17, no. 3, pp. 1025–1030, 2014.
[32] N. Mohamed, S. Lazarova-Molnar, I. Jawhar, and J. Al-Jaroodi, "Towards service-oriented middleware for fog and cloud integrated cyber physical systems," in *Proc. IEEE 37th Int. Conf. Distrib. Comput. Syst. Workshops (ICDCSW)*, Jun. 2017, pp. 67–74.
[33] K. M. Alam and A. El Saddik, "C2PS: A digital twin architecture reference model for the cloud-based cyber-physical systems," *IEEE Access*, vol. 5, pp. 2050–2062, 2017.
[34] K. Gai, M. Qiu, H. Zhao, and X. Sun, "Resource management in sustainable cyber-physical systems using heterogeneous cloud computing," *IEEE Trans. Sustain. Comput.*, vol. 3, no. 2, pp. 60–72, Apr./Jun. 2017.
[35] I. Stojmenovic, "Fog computing: A cloud to the ground support for smart things and machine-to-machine networks," in *Proc. IEEE Australas. Telecommun. Netw. Appl. Conf. (ATNAC)*, Nov. 2014, pp. 117–122.
[36] M. Frincu, "Architecting a hybrid cross layer dew-fog-cloud stack for future data-driven cyber-physical systems,"in *Proc. IEEE 40th Int. Conv. Inf. Commun. Technol., Electron. Microelectron. (MIPRO)*, May. 2017, pp. 399–403.
[37] A. Seitz, J. O. Johanssen, B. Bruegge, V. Loftness, V. Hartkopf, and M. Sturm, "A fog architecture for decentralized decision making in smart buildings," in *Proc. ACM 2nd Int. Workshop Sci. Smart City Oper. Platforms Eng.*, Apr. 2017, pp. 34–39.
[38] A. Munir, P. Kansakar, and S. U. Khan. (Jan. 2017). "IFCIoT: Integrated fog cloud IoT architectural paradigm for future Internet of Things." [Online]. Available: https://arxiv.org/abs/1701.08474
[39] R. Mahmud, F. L. Koch, and R. Buyya, "Cloud-fog interoperability in IoT-enabled healthcare solutions," in *Proc. 19th Int. Conf. Distrib. Comput. Netw. (ICDCN)*, 2018, Art. no. 32.
[40] L. Gu, D. Zeng, S. Guo, A. Barnawi, and Y. Xiang, "Cost efficient resource management in fog computing supported medical cyber-physical system," *IEEE Trans. Emerg. Topics Comput.*, vol. 5, no. 1, pp. 108–119, Jan./Mar. 2015.
[41] H. Song, Q. Du, P. Ren, W. Li, and A. Mehmood, "Cloud computing for transportation cyber-physical systems," in *Cyber-Physical Systems: A Computational Perspective*, vol. 15. London, U.K.: Chapman & Hall, 2015, pp. 351–369.
[42] D. Bruneo *et al.*, "Stack4Things as a fog computing platform for Smart City applications," in *Proc. IEEE Conf. Comput. Commun. Workshops (INFOCOM WKSHPS)*, Apr. 2016, pp. 848–853.
[43] R.-Y. Chen, "An intelligent value stream-based approach to collaboration of food traceability cyber physical system by fog computing," *Food Control*, vol. 71, pp. 124–136, Jan. 2017.
[44] M. Mededjel, G. Belalem, and A. Neki, "Towards a traceability system based on cloud and fog computing," *Multiagent Grid Syst.*, vol. 13, no. 1, pp. 47–68, 2017.
[45] M. Heiss, A. Oertl, M. Sturm, P. Palensky, S. Vielguth, and F. Nadler, "Platforms for industrial cyber-physical systems integration: Contradicting requirements as drivers for innovation," in *Proc. IEEE Workshop Modeling Simulation Cyber-Phys. Energy Syst. (MSCPES)*, Apr. 2015, pp. 1–8.
[46] D. Wu, J. Terpenny, L. Zhang, R. Gao, and T. Kurfess, "Fog-enabled architecture for data-driven cyber-manufacturing systems," in *Proc. ASME Int. Manuf. Sci. Eng. Conf. (MSEC)*, June. 2016, p. V002T04A032.
[47] M. S. de Brito, S. Hoque, R. Steinke, A. Willner, and T. Magedanz, "Application of the fog computing paradigm to smart factories and cyber-physical systems," *Trans. Emerg. Telecommun. Technol.*, vol. 29, no. 4, p. e3184, 2018.
[48] E. Jantunen, U. Zurutuza, M. Albano, G. di Orio, P. Maló, and C. Hegedus, "The way cyber physical systems will revolutionise maintenance," in *Proc. 30th Conf. Condition Monit. Diagnostic Eng. Manage.*, 2017, pp. 1–8.

JAMEELA AL-JAROODI received the B.Sc. degree in computer science from the University of Bahrain, the M.Sc. degree in computer science from Western Michigan University, the Ph.D. degree in computer science from the University of Nebraska–Lincoln, and the M.Ed. degree in higher education management from the University of Pittsburgh. She was a Research Assistant Professor with the Stevens Institute of Technology, Hoboken, NJ, USA, then an Assistant Professor with United Arab Emirates University, UAE. She was an independent Researcher in the computer and information technology. She is currently an Associate Professor and a Coordinator of software engineering concentration with the Department of Engineering, Robert Morris University, Pittsburgh, PA, USA. She is involved in various research areas, including middleware, software engineering, cyber-physical systems, and distributed and cloud computing, in addition to UAVs and wireless sensor networks.

NADER MOHAMED received the Ph.D. degree in computer science from the University of Nebraska–Lincoln, Nebraska, USA, in 2004. From 2004 to 2006, he was an Assistant Professor of computer engineering with the Stevens Institute of Technology, Hoboken, NJ, USA. He was with the College of Information Technology, United Arab Emirates University, Al Ain, UAE, as an Assistant Professor, from 2006 to 2009, and an Associate Professor from 2009 to 2015. He is currently an independent Computer and Information Research Scientist in Pittsburgh, PA, USA. In addition, he has eight years of industry experience in the information technology field. His current research interest focuses on middleware, cloud and fog computing, cyber-physical systems, unmanned aerial vehicles, and cyber security.