Hindawi
Mobile Information Systems
Volume 2021, Article ID 3231626, 12 pages
https://doi.org/10.1155/2021/3231626

Research Article

Hybrid Resource Modeling and Scheduling Platform Based on Multisource Cooperation for Cyber-Physical-Human System

Jianyong Zhu,[1] **Yang Wang**,[2] **and Xiaoqiang Yu**[1]

[1]*Beijing Advanced Innovation Center for Big Data and Brain Computing, Beihang University, Beijing 100191, China*
[2]*Beijing Huatech Trusted Computing Information Technology Co.,Ltd., Beijing 100195, China*

Correspondence should be addressed to Jianyong Zhu; zhujy@buaa.edu.cn

Received 2 August 2021; Revised 17 October 2021; Accepted 20 October 2021; Published 3 November 2021

Academic Editor: Floriano Scioscia

Cyber-Physical-Human System (CPHS) is receiving increasing attention as an interrelated system that integrates computing, physical, and human resources. Different from existing computing paradigms, CPHS generally comprises heterogeneous software and hardware resources from multiple resource providers and allows interaction among them. The existing research on resource scheduling either focuses on traditional computing resources without consideration of physical environmental factors or is limited to specific application fields, making it difficult to promote and apply across domains. In this paper, we design and implement an application-independent CPHS resource scheduling platform based on a self-defined hybrid resource model. The platform provides a complete instantiation environment for all kinds of CPHS resources and their interactions and supports flexible customization of various resource scheduling strategies. We have injected several different resource scheduling policies into this platform, and the execution results are in line with expectations. The platform is extended to be implemented based on mainstream technologies that have been widely practiced, and the code is open source.

1. Introduction

Cyber-Physical-Human System (CPHS) is a new computing paradigm whose essence lies in "full-stack and whole network information on demand." The active human interaction and participation in CPS processing foster the emergence of CPHS, in which multiple participants work together to achieve the system goals [1]. CPHS typically consists of interconnected different resource providers (RPs) that communicate across time and space and provides an ideal paradigm for the design and construction of intelligent environments with command and control [2–5]. Different from the traditional computing paradigm, CPHS needs to realize the collaborative work of resources in the open environment that includes humans, computers, and the physical world and realize the resource management, scheduling, interactive communication, and dynamic evolution at runtime. Therefore, the resource scheduling platform for CPHS not only needs to realize access control of resource types provided by clouds, networks, terminals, and humans, but also needs to develop a set of rules to realize abstraction and collaboration of open resources.

Unfortunately, resource scheduling for CPHS remains a challenging task due to the complex and heterogeneous hardware and software entities, as well as cross-space-time interactions between entities. First, resource scheduling in existing computing paradigms usually focuses on a single supply-side computing resource and lacking consideration of environmental and social factors, as well as cross-layer resource coordination. In cloud computing, resource scheduling only abstracts and quantifies computing resources in large-scale data centers. The process is mainly cost-aware based on QoS constraints with considering the profit of the homogeneous service providers [6–9], leading the data generated by applications to be more tightly coupled on underlying hardware. Resource scheduling in edge computing (EC) providers low latency services by connecting computing resources and services from cloud to the nearby end-users. Decentralized computing resources are aggregated at the edge of the network to process data from

closer devices [10, 11]. Compared with EC, although mobile edge computing (MEC) takes into account the abstraction of spatiotemporal attributes when applications offload workloads from mobile devices to the cloud, it seldom considers the linkage between the cloud and the terminal, nor does it take into account the initiative of humans in production loop [12–15]. Secondly, there is no general resource scheduling platform for CPHS, resulting in the high cost of customized scheduling strategy, which also limits the existing CPHS scheduling research to specific domains. Xue and Yu [16] make conceptual assumptions about resource management of future energy systems and propose to incorporate environmental and social attributes as well as human influences into resource scheduling. Han [17] realized resource modeling of moving vehicles and traffic and guided driving by building a virtual scheduling system. References [18, 19], respectively, put forward the idea of using CPHS to build a dispatching platform for medical diagnostics and aviation fields. Obviously, the above-mentioned studies of CPHS scheduling are only for specific application scenarios, and most of them are still in the theoretical stage.

To reach a consensus on the concepts of CPHS and achieve the management and scheduling of full-stack resources in CPHS, as well as customize the scheduling strategy by providing an application-independent resource scheduling platform for CPHS, we propose a unified hybrid resource scheduling framework for CPHS in this paper. First, by designing a hybrid resource model, HMIModel, we implement an abstraction of resources including clouds, networks, terminals, and humans as well as their interactions in CPHS. The properties of the schedulable resource units of each layer are defined to facilitate the semantic description and resource scheduling of CPHS applications. We highlight the natural heterogeneity of resource units at different layers in the model and emphasize cross-layer interactions among different RPs. Moreover, human activities are abstracted from the perspective of resources provided to analyze the contribution of humans as service providers to CPHS. We then design and implement an HMIModel-based resource orchestration tool, HMIEditor, that prototypes CPHS application requirements through the built-in elements and rules of HMIModel and quickly customizes resource requirements by formatting storage and parsing. Finally, using Kubernetes [20], the current mainstream container orchestration system, we design and implement an application-independent CPHS resource scheduling and management platform. The platform features a loosely coupled but highly interactive design that supports the customization of scheduling strategies, as well as monitoring and tracking of application execution status. We have injected typical CPHS scheduling strategies into the platform and analyzed the scheduling results, and the expected effect was achieved.

The current submission is an extension of our earlier work "Hybrid Resource Orchestration and Scheduling for Cyber-Physical-Human Systems" [21] that appeared in the IEEE 18th International Conference on Smart City. The work incorporates the most recent progress we have made, which greatly improves the conference version in several aspects. The main contributions of our work are as follows:

(i) We have designed a hybrid resource model with a set of semantic specifications to abstract the various heterogeneous resources and their interactions in CPHS.

(ii) We have designed a resource orchestration tool based on the hybrid resource model to realize the rapid customization of resource requirements for CPHS applications.

(iii) We have designed and implemented a general CPHS resource scheduling and management platform, which can be used to quickly customize scheduling strategies and simulate their execution.

To the best of our knowledge, this paper is the first to propose a complete domain-independent resource scheduling framework for CPHS application. Our code is available as open source in Mulan [22], an open-source community in China.

The rest of this paper is organized as follows. We firstly depict the background and challenges in Section 2. Hybrid resource modeling is detailed in Section 3. Section 4 shows the design of the resource orchestration tool. In Section 5, we describe in detail the design principles and system implementation of the application-independent resource scheduling and management platform. Experiments are shown in Section 5.3. Related work is discussed in Section 6, before we finally draw the conclusions and discuss the future work in Section 7.

2. Background

2.1. Resource Requirement in Existing Computing Paradigm. Existing computing paradigms have focused on improving resource provision efficiency in the last decade. Cloud computing brings computing and storage resources in different geographic locations together through virtualization for on-demand use [23]. However, centralized resource management causes serious transmission latency for interactive applications [24, 25]. Edge computing provides location-aware resources to bridge the latency gap between the terminal devices with limited resources and the cloud servers.

Considering the available network resources and QoS constraints of latency-sensitive applications, edge servers, and terminals, resources are allocated reasonably for applications [26, 27]. For applications in the mobile edge computing and the Internet of things (IoT), connectivity and location information between entities also become an important reference for resources offloading and scheduling [28, 29]. In general, existing intelligent systems mainly provide computing resource-oriented services. They pay attention to the improvement of the efficiency of data-intensive tasks on the single resource supply-side during resource scheduling, but they lack consideration of the physical environmental factors and the interaction among cross-layer resource providers.

2.2. Resource Scheduling in CPHS. Although there have been some studies on CPHS resource scheduling for specific domains, there is a lack of conceptual consensus in terms of system basis, let alone research on domain-independent resource scheduling platforms. In this paper, we divide heterogeneous resources from multiple sources into three categories from the perspective of resource providers, namely, sensing units, computing units, and control units. The sensing unit required by information consumers mainly comes from the environmental information collected by terminal devices. Computing units in the CPHS loop are derived from multisource RPs including clouds, networks, and terminals. The control units are responsible for the execution logic of CPHS applications after obtaining the computing and sensing units, including interactions and collaborations between RPs as well as abstraction of human activities. Although the existing work has conducted certain research on the supply efficiency of the above resources in specific application fields, the following problems still exist:

(i) Traditional computing paradigms are designed for computing resource services, but the full-stack resource information is not included in the scheduling strategy.

(ii) Existing research on CPHS resource scheduling focuses on a specific domain and lacks domain-independent conceptual consensus and resource abstraction methods.

(iii) Due to the lack of a general scheduling platform, the existing scheduling strategy is limited to the dedicated field, and the iteration cost is high.

In this paper, we aim to build a hybrid resource model covering clouds, networks, terminals, and environmental factors (such as human capacity) and build an application-independent CPHS resource scheduling platform on this basis to achieve resource scheduling strategy customization, injection, and evolution analysis, so as to improve the effectiveness of CPHS resource scheduling.

3. Hybrid Resource Modeling

Different from the existing description specifications for CPHS which focus on specific domains, we propose HMIModel, a hybrid resource abstraction model for CPHS application; by establishing a unified description specification, we can realize the abstraction of computing, physical, and human resources in CPHS and their interactions across time and space.

We first define the RPs in CPHS to facilitate understanding of HMIModel as follows:

(i) Edge: RPs that are on the edge of the management domain and close to the data source, terminal devices, or users include devices that provide network connectivity and edge servers that provide computing capacities.

(ii) Terminals: These are RPs with limited resources, including sensors, wearable devices, and mobile phones. They capture sensing data, either spontaneously or driven by humans, and interact with other RPs through network connections to achieve information aggregation or feedback.

(iii) Clouds: These are centralized server clusters with high computing and storage capacity. They are responsible for data-intensive task calculation, information storage and backup, and completion of tasks partition and coordination between RPs of the same layer or different layers.

(iv) Humans: As participants in the CPHS production loop, their interactions with other RPs can trigger, drive, or interfere with the execution of the CPHS application.

The architecture of HMIModel is shown in Figure 1. The hierarchical distribution of sensing units, computing units, and control units discussed in Section 2.2 is given, and resource abstraction and pooling are conducted for different layers of RPs. HMIModel defines the semantic rules of resource description for CPHS applications. Depending on the resource units and the services they provide, we classify RPs into four layers of entities in HMIModel, namely, clouds, networks, terminals, and humans, and give the attribute definitions of these resource units and semantic rules of interaction among them.

3.1. Cloud. It consists of data centers with clusters of servers that provide computing and storage services on demand using dedicated high-speed network connections. Specifically, multiple cloud servers form a cluster with high computing and storage capabilities, while one or more clusters form a data center according to their location and the collaborative computing ability. To facilitate the description of the hierarchy of RPs in the cloud layer, we present a three-layer structure, namely, data center–server–container structure, where a data center consists of clusters of physical or virtual machines (i.e., servers), with several containers running on each server based on its resource supply capacity. The container is the executor of each partitioned task and is the smallest object for resource resizing at runtime. We define the main properties of the three layers as follows:

(i) Data center: blockip (BIP) refers to the public access address of the data center; location refers to the geographic location of the data center; capacity refers to the supply capacity of various resources in the data center; scale refers to the number of servers in the data center.

(ii) Server: Position is the server address composed of the cabinet-rack-IP triplet, used to identify servers and meet data localization requirements; health indicates whether the server is in normal operation; resource_status refers to the used and available resources; parallelism is used to indicate the number of containers the server allows to run in parallel.

(iii) Container: appid is used to identify the application to which it belongs; resources_occupy represents

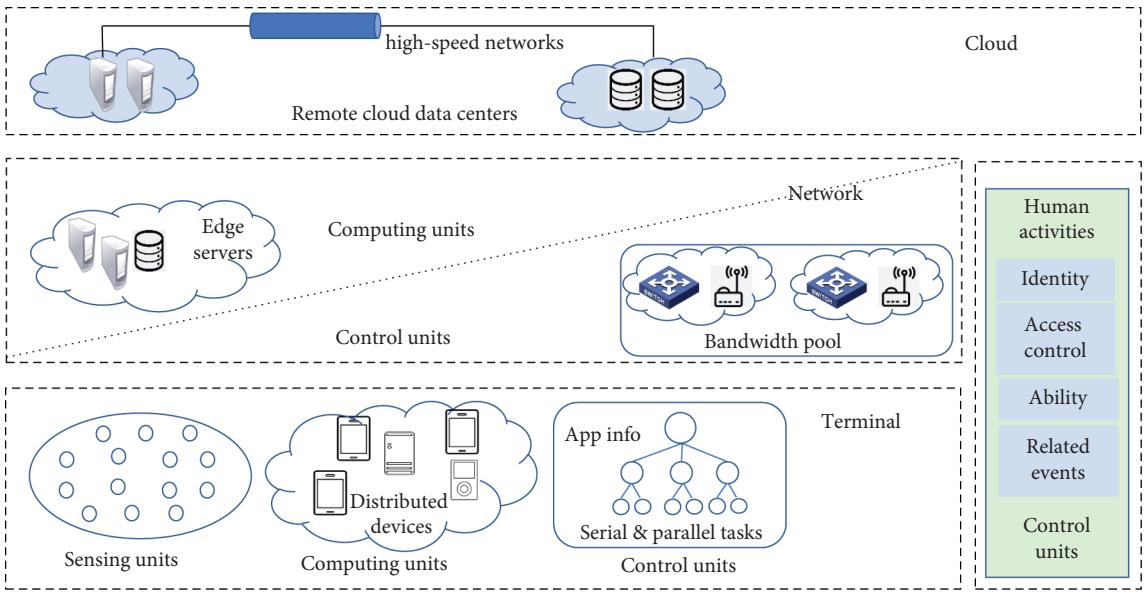

Figure 1: Architecture overview of HMIModel.

resource usage at runtime; status is an enumerated value that indicates the status of the container's execution, such as waiting, initializing, running, and finishing.

3.2. Network. In CPHS, network refers to a kind of RPs close to terminals. According to its roles in the CPHS loop and the difference of resources provided, it can be divided into two categories: one is the net node used for data forwarding and interconnection between RPs in application execution; the other is the edge server with computing and storage capacity, which is the key to application division and offloading and plays the role of linking the preceding and the following. CPHS applications reduce the cost of network bandwidth and time to communicate with the cloud by offloading the workloads from the terminal to nearby edge servers, and the cloud can also save cost by delegating multiple tasks to the edge servers through application partition. By offloading some real-time tasks to the edge of the network, applications reduce interactive latency and improve the QoS. We define the following main properties for these two types of RPs to indicate their resource capabilities:

(i) Net node: location describes its geolocation information; list details the list of RPs connected to it; bandwidth is used to indicate its network bandwidth capability

(ii) Edge server: location emphasizes its geographic coordinates, which are used to realize location-aware CPHS services and, on this basis, to realize the best solution for application partitioning and task offloading

3.3. Terminal. The terminal layer of CPHS contains a variety of heterogeneous RPs, including sensing devices to collect various environmental information, interactive devices that can provide control resources, and computing or storage devices with certain task processing capabilities. The terminal interconnects with the RPs in the network or cloud to launch tasks (including uploading, offloading, and partitioning), send and receive data, aggregate data, display results, etc. Terminal mobility, in addition to the driving effect of human social attributes on application execution, makes different RPs interactions more frequent and resource requirements more dynamic. We highlight the division of labor between different roles in terminals of HMIModel: sensing devices are used for information production and acquisition in specific fields; functional devices are used for information aggregation, application partitioning, partial task execution, initiation of application offloading, and interaction with other RPs. In addition, as an important participant in human-computer interaction, functional devices can be divided into general devices and domain-specific devices. The main properties of the terminal are shown as follows:

(i) Location: the position of the device given in the form of coordinates

(ii) Service time: the period of service available for the terminal each day

(iii) Service list: the list of functions provided by the device; for example, the service list of the air conditioner is [refrigeration, heating]

(iv) Type: the type of end device, used to indicate whether the device is sensing, general, or domain-special device

We emphasize the temporal-spatial features of terminals in CPHS, which indicate the availability of terminals and the dynamic changes of resource demand and supply, as the constraints of cross-layer resource scheduling.

3.4. Humans. We aim to abstract the human general ability and the ability to perform the specified task in the CPHS loop through HMIModel, including the perception of the status of the task and the interaction with the task during execution. We further illustrate the role of humans in the CPHS loop as an information provider, not just a consumer, as shown in Figure 2. The first stage of the production loop is the data acquisition after the task is triggered, including the collection of data information and task information. The second stage is data processing, which is a long service, responsible for task execution and result update. The third stage of the loop is an abstract collection of interactive events for process advancement. The fourth stage is human activity, which facilitates the production loop by performing event-driven actions. The impact of the activity on the execution of the task is then reflected in the system, forming a new production loop. The main properties of humans are defined as follows, to ensure the rationality and legitimacy of the participation of humans as a producer in the CPHS loop:

(i) Identification: it is the certificate of human participation in the CPHS loop, which is used to verify the identity of participants to avoid illegal human intervention

(ii) Function list: it is used to describe how a person interacts with other functions; each function has two properties, where FID is used to identify function for other RPs to call and APIs are used to describe how other services interact with humans

(iii) Activities: they involve the collection of information about a person's abilities and past experience to perform a task

(iv) Metadata: it defines the data format for humans to interact with other entities (functions or RPs) in CPHS

(v) Service status: it is used to indicate the feasibility of a person to participate in an assigned task, where location represents the runtime location of a person and service time represents the available time period of a person participating in CPHS

The normalization of cross-layer interaction and collaboration among heterogeneous RPs in CPHS can meet the dynamic demand of CPHS applications for resources in execution, while the abstraction of RPs properties of each layer by HMIModel enables CPHS applications to quantify and customize their demand, to realize application-independent resource scheduling.

4. Hybrid Resource Orchestration Tool

To achieve rapid prototyping of CPHS applications and customize resource requirements based on HMIModel, we

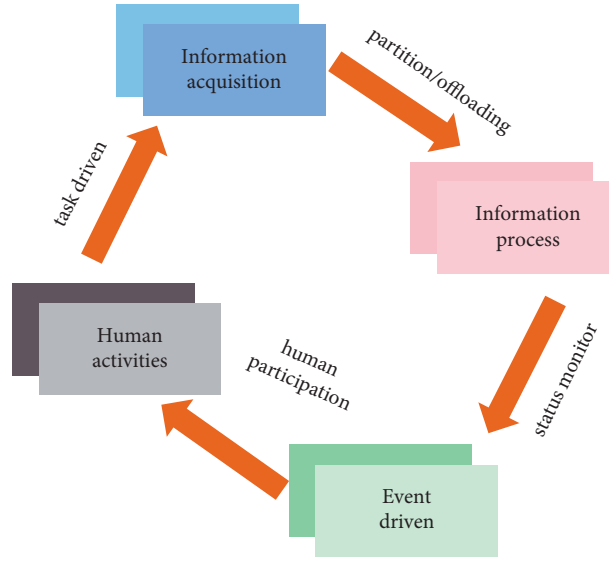

Figure 2: Human activities in CPHS loop.

design and implement HMIEditor, a tool for resource orchestration of CPHS applications. HMIEditor is a web service based on mxGraph 2 [30], is available as open source in [22], is compatible with current major browsers, and provides a variety of components that can be flexibly edited and configured with their properties to characterize different RPs and types of resources. By customizing and combining these functional components, CPHS application prototypes can be easily created. The structure of HMIEditor is shown in Figure 3, which is mainly composed of two modules, namely, prototype building and prototype management.

The prototype building module is responsible for constructing the resource demand model of CPHS application: HMIModel loading is responsible for loading the predefined hybrid resource model; parsing the built-in properties, interaction rules, event lists, etc.; and building them into the corresponding components of HMIEditor. RPs management is responsible for the selection of different combinations of RPs. Properties abstraction is responsible for quantifying resource units for each selected RP and configures interaction and human-triggered events between RPs.

The Topology building is responsible for correlating the configured components to form a complete resource requirement topology. The prototype management module is responsible for managing the generated prototype, where validity checking is responsible for the validation of the prototype to prevent the generation of illegal resource requests, model exporting is responsible for the formatted storage of the prototype, and model importing is used to import the existing prototype for modification or further customization.

The features of HMIEditor are as follows:

(i) The built-in tree hierarchy makes it easy for users to build a topology of application resource requests. HMIEditor realizes requirement customization by dragging and combining different functional

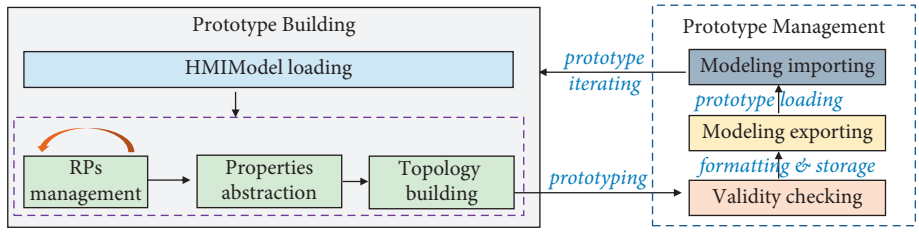

Figure 3: The structure of HMIEditor.

components freely and provides a variety of linking tools to express different dependencies and interactions between components. All components have their own built-in resource layer to which they belong, without manual annotation.

(ii) A key-value-based data organization format is designed, which enables the application prototype to be stored and recovered quickly. HMIEditor uses JSON 2.2 [31] to format the prototype and uses ProtoBuffer [32] for serialization storage and deserialization parsing when the prototype is exported and imported, respectively.

(iii) It supports the scalability of the hybrid resource orchestrating based on HMIModel loading and verifies the legitimacy of the generated prototype to ensure the schedulability of resource requests.

5. Design and Implementation of CPHS Application Scheduling Platform

Compared with the traditional computing resource management platform, the CPHS scheduling platform not only can access the various heterogeneous resources distributed throughout the full-stack, but also needs to support the customization of scheduling strategies for different revenue targets in different application areas. At present, CPHS scheduling is focused on specific application scenarios and does not have extensibility, and iteration cost is expensive. We designed CPHS application scheduling platform HMIPlatform, using the latest software agent and lightweight container technology. Through the flexible access control of various resources and modeling of temporal and spatial constraint, we realize resource scheduling customization and runtime status monitoring.

HMIPlatform is designed based on the container orchestration system Kubernetes [33] and uses HMIModel to specify a set of resource description specifications. Meanwhile, a message-driven mechanism is designed to realize the interaction and collaboration between RPs. The conceptual architecture is shown in Figure 4. HMIPlatform consists of the following modules: message queue, rule controller, event handler, RP agent, and resource controller. All modules are packaged into different images and run in Kubernetes as pods, communicating with each other through remote procedure call (RPC) protocol after service registration and discovery. Next, we explain the role of each component in the platform in terms of resource management and implementation of interaction rules.

5.1. Formal Representation and Management of Resources. The first is the instantiation of CPHS resources in the platform, which is the responsibility of the resource controller module. The module is also responsible for monitoring and updating the status of resources. All physical resources need to be abstracted into data structures to be stored for system update and maintenance. We use custom resource definition (CRD) of Kubernetes to represent the resources and store them in etcd [34], a distributed key-value database. The resource needs to be registered and uniquely identified in the system. The properties of CRD in the platform are shown in Table 1. The properties access mode and tracking URL, as well as the probe object, are used to discover the resource in the platform. Therefore, the resource controller image contains a probe with its address and launch commands and start-up parameters to discover and update the resources in the system. The properties of the probe are shown in Table 2, and the resource status probed is shown in Table 3. Note that the patcher property is specially designed to improve the scalability of the probe.

In effect, the resource controller transforms the CRD into a native resource in Kubernetes, and the deployment in Kubernetes is responsible for producing the actual pod to run the probe containers and then exposing the resource through the corresponding services to make it accessible to other objects.

5.2. Message-Driven Resource Interaction Design. HMIPlatform employs message-driven interactions between components to enable inter-resource collaboration.

First, the RP agent module is designed to achieve unified access to all kinds of resources. Specifically, there are two considerations: (i) It provides a unified resource access model that ensures the versatility of the resource access interface to support access to a variety of heterogeneous CPHS resources, especially smart end devices. (ii) In addition to providing a description of the resource state, the appropriate operational interface is required, which is actually given in the form of a RESTful API in RP agent. Therefore, the RP agent is actually implemented in two steps. The first is the design of the resource access layer. The resource access layer provides access control to protect the access of sensitive resources. The agent service can describe the status of the resources it cares about to the resource access layer, and the access layer will actively push status change notifications under the subscription of the resource agent service. Resource agents are mainly divided into two categories: terminals and humans, and other general

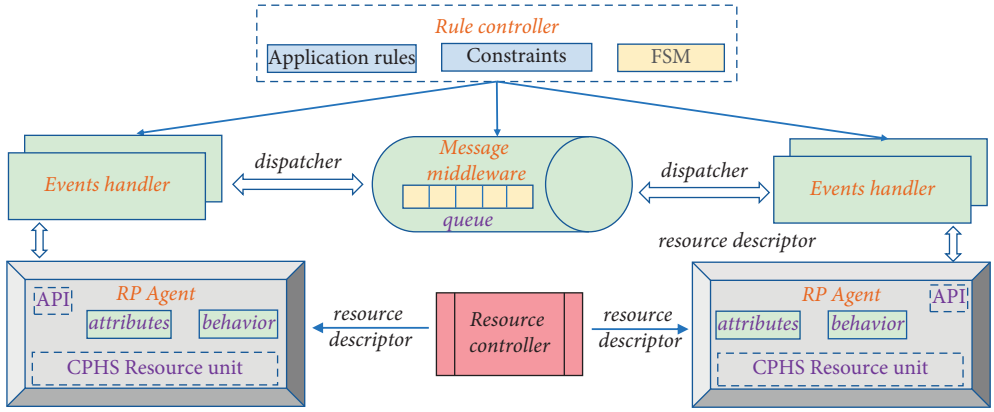

Figure 4: The architecture of HMIPlatform.

Table 1: The properties of terminal in CPHS.

Properties	Type	Meaning
ResourceKind	Enum	Resource types, including humans, machines, services
Icon	String	The location of the identifier used for front display
Description	Map [string]	The resource description
AccessMode	Enum	Resource access mode, namely, shared or exclusive
ProbeSpec	Probe spec	Properties of the probe used to discover the resource
Track	String	The address and port for accessing the resource

Table 2: The properties of probe.

Properties	Type	Meaning
Enable	Bool	Determines whether to enable the probe
Image	String	The probe image
Args	[] String	Start-up parameters of the probe
Patcher	[] String	Field modifier for the probe

Table 3: The properties of resource status.

Properties	Type	Meaning
Phase	Enum	Resource status, including Pending, Running, and Failed
ProbePhase	Enum	Probe status, including NotReady, Pending, Synchronous, and Failed
Bound	Bool	Identifies whether the resource has been bound
CreateTime	Time	Creation time
StratTime	Time	The time the resource participates in the execution of an CPHS application

resources can be directly accessed using the existing interfaces of Kubernetes. For terminal devices, their functionality is relatively fixed, so when the resource agent is deployed, there is no need to make significant modifications to their properties and functional interfaces. Here, let us give an example of an agency service designed with air conditioning. The developer first models the air conditioner, adds its properties and space-time information to the original model, and then provides the appropriate modified interface according to the resource access layer specification. The developer then needs to encapsulate the resource interface so that users can act through the RP proxy, as shown in Figure 5. However, there are some differences in the implementation of the human resource agent. The human resource agent service describes the capabilities of the person in a specific scenario, and the agent service is constantly updated as a person moves. For example, when a person faces an air purifier, its service agent has an API to control the purifier, and when a person moves to a coffee machine, the agent service needs to add the API for using the coffee machine.

Second, the rule controller module is designed to implement the abstraction of the execution logic of CPHS applications, mainly including the following three parts: (i) abstracting the execution status of the CPHS application into customized messages and instructions that can be recognized by the system, such as task partition, offloading, and uploading; (ii) constructing the expression of resource scheduling constraints, such as temporal-spatial properties and QoS constraints; (iii) defining the superclass of

```
@PostMapping ("/turnoff")
@Accessible ("true")
public String turnoffAirConditioner (){
        RestTemplate restTemplate = new RestTemplate ();
        String url = "http://xxx/api/services/fan/turn_off";
        HttpRP <String > request = new HttpRP <>(
        new JSONObject ().put ("entity_id", "fan.xiaomi").toString (),
        new HttpHeaders ().setContentType(MediaType.APPLICATION_JSON));
        return restTemplate.postForObject (url, request , String.class);
}
```

Figure 5: API design for turning off the air conditioner in RP age.

application execution logic using the finite state machine (FSM) to facilitate the inheritance and extension of the subsequent event handlers. To facilitate rapid resource access and interaction among RPs, we define the following three state transition modes for FSM in HMIPlatform: (i) an initial state, a final state, and a trigger event; (ii) an initial state, multiple final states, a trigger event; and (iii) an initial state, a final state, and multiple trigger events. The principle is shown in Figure 6.

The event handler module is then designed to take responsibility for the action taken when a message is received. Handlers inherit from FSM and are represented as quads, namely, pre-state, post-state, event type, and callback functions (hook or transaction). Here, we give an implementation example of event handler for the generic terminal device resources, which demonstrates the implementation of the transition from the resource newly created state to the binding state, as shown in Figure 7, where the addTransition function implements the transformation of a single-arc based quad, that is, an initial state, a final state, and a trigger event.

Finally, the message middleware module of the messaging delivery service is designed. We consider the spatiotemporal information as an important factor to interfere with the scheduling policy, and formulate message routing policies to reflect its impact during resource scheduling. The module is based on traditional message middleware and is designed to include the following features:

(i) The module is written using Java, and network communication is based on Netty [35]. Netty is an asynchronous, event-driven network application framework and tool that can be used to quickly develop maintainable, high-performance server and client network application frameworks, greatly simplifying network programming.

(ii) We use defining topics to distinguish between different types of resource agent services. We perform topic bindings for different messages to make it easier to deliver messages to the appropriate event handlers. For example, the topic of coffee machines of different brands is coffee machines; thus, the same type of resources can be aggregated and different types of resources can be distinguished.

(iii) We serialize the message with Kryo [37] to reduce the transmission delay.

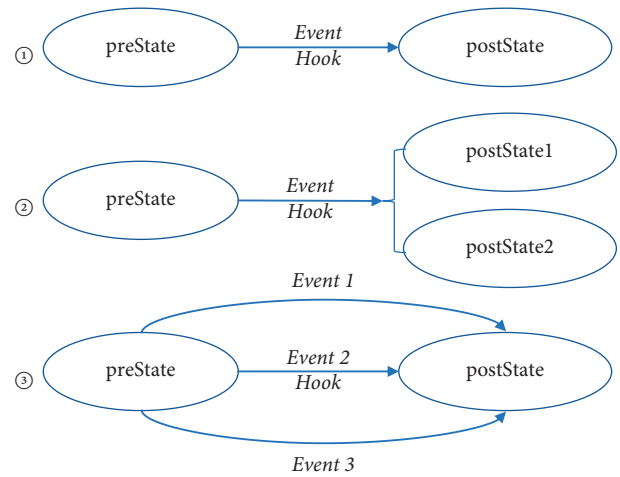

Figure 6: The state transition modes of FSM in HMIPlatform.

(iv) Message routing policies based on spatiotemporal modeling are supported. We use the IndoorGML [36] specification to model static space, maintain distance information between different resource entities in space, generate routing tables based on distance information, and periodically send the latest routing tables to message middleware.

From the perspective of the traditional resource management platform, HMIPlatform implements resource management and status update according to the resource description in HMIModel. From the perspective of CPHS application, HMIPlatform connects down to all kinds of CPHS resources in the real world and provides a relatively unified resource access interface to the resource agent service and message middleware at the upper level, thus realizing the scheduling of hybrid resources.

5.3. Design and Validation of CPHS Resource Scheduling. HMIPlatform enables formal representation and interaction of CPHS resources with the help of CBD and message middleware. In this section, we verify our support for custom resource scheduling policies through case studies of resource scheduling. The essence of resource scheduling is to find a match between available resources and tasks waiting to be allocated. Compared with the traditional computing paradigm, the multisource collaboration in CPHS makes its scheduling process more complex. Based on the customized resource requirements using the hybrid resources model HMIModel, QoS requirements and human activities in CPHS are abstracted as concerns and integrated into the system architecture for scheduling design that is shown in Figure 8. Physical RPs view outputs the resource supply capacity of the current system and the spatial-temporal factors of heterogeneous resources that affect scheduling performance to system view, while the concerns' view is used to abstract and integrate QoS constraints and the change of human abilities into resource scheduling. The scheduling mechanism performs multilevel resource allocation of heterogeneous resources based on the first two views.

```
private static final StateMachineFactory<CPHSTerminalResourceImpl,
                        CPHS TerminalResourceState,
                        CPHS TerminalResourceEventType,
                        CPHS TerminalResourceEvent> stateMachineFactory

                        = new StateMachineFactory<CPHSTerminalResourceImpl,
                       CPHS TerminalResourceState,
                       CPHS TerminalResourceEventType,
                      CPHS TerminalResourceEvent> (CPHSTerminalResourceState.NEW).

.addTransition ( CPHSTerminalResourceState.NEW,
                        CPHS TerminalResourceState.BIND,
                        CPHS TerminalResourceEventType. NODE_BIND,
                        new CPHS TerminalResourceNodeBindTransition () ).installTopology ();
```

Figure 7: An implementation of an event handler in HMIPlatform.

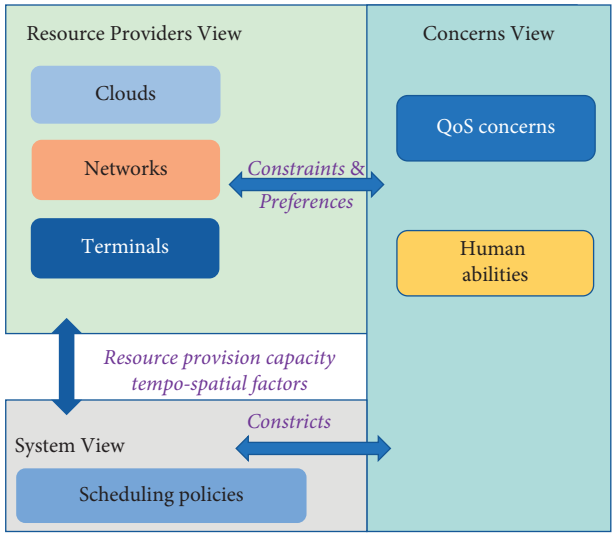

Figure 8: Scheduling view of CPHS.

5.3.1. Definition of CPHS Application. Based on the scheduling view, we define CPHS application as a triple $A = <\alpha, \beta, \gamma>$, where

(i) α is a set of tasks for application partitions, with each task being composed of an identifier and software requirement S and hardware requirement H.

(ii) β is the set of constraints in CPHS scheduling detailed above.

(iii) γ is a set of interactions between tasks in a topology composed of multitier resource requests.

5.3.2. Injection of Cooperation Policies into HMIPlatform. Various factors are considered in CPHS resource scheduling. CPHS should consider environmental factors and human social attributes in addition to typical traffic, system load, and SLA constraints. For example, when someone in the office wants to print documents quickly, distance and printer availability will be the dominant factor in the scheduling strategy, and if this person is more concerned about the cost of printing, the price factor will have the greatest weight in the scheduling strategy. Specifically, constraints at each level can be abstracted into a set of function q, which is a mapping of the processing logic from the message to the bound event handler in HMIPlatform, as shown in the following equation:

$$\theta(\text{msg}) = \{b \mid b \in B, \text{events handlers of this msg}\}. \quad (1)$$

In a particular application scenario, the cooperation between RPs depends on the multilayer policies, which can be implemented as multiple scheduling functions, and the final scheduling q can be formed by combining the policies.

Taking the "nearest" policy as an example, the developer first needs to upload a configuration file to the spatial mapping engine in HMIPlatform using Indoor GML in [38] to spatially model the entire environment; then, the space mapping engine can obtain the distance information between each subspace in the entire space. When the resource service is registered to the platform through the RP agent, the RP location information will also be injected into the spatial mapping engine, so that the platform can get the distance information between RPs. Then, the distance information is stored in a hash table and sent to the scheduling policy module of each event handler. This module makes scheduling decisions at runtime based on policy functions passed in by the developer, such as "nearest first" and "lowest cost first," to achieve dynamic collaboration between heterogeneous RPs.

5.3.3. Validation of Scheduling Algorithm under Multiple Constraints. We inject two policies of "nearest" and "maximum revenue" into the handlers of HMIPlatform and simulate the scheduling process for different tasks at t_1 and t_2 on this basis. We conduct quantitative analysis of the benefits of task scheduling at time t_1 and t_2, as shown in Figure 9. Assume that there are no dependencies between tasks; there are 5 pending tasks and 7 available resource units at time t_1, and 4 pending tasks and 5 available resource units at time t_2. Each element (that is, each 2-tuple) in the matrix represents the benefit of using the resource unit to process the task and the resource allocation overhead caused by the distance, respectively. It is expensive to assign R_4 and R_5 to tasks at time t_1 due to geographic location. The same situation also applies to R_5 at time t_2.

Due to the constraint of "nearest first" when resources are scheduled, the scheduling algorithm first considers the location information of the available resources and then carries out the matching algorithm based on the benefit

t1	$Task_1$	$Task_2$	$Task_3$	$Task_4$	$Task_5$
R_1	(12, 1)	(7, 1)	(9, 1)	(7, 1)	(9, 1)
R_2	(8, 1)	(9, 1)	(6, 1)	(6, 1)	(6, 1)
R_3	(7, 1)	(17, 1)	(12, 1)	(14, 1)	(9, 1)
R_4	(12, 10)	(9, 10)	(12, 10)	(15, 10)	(6, 10)
R_5	(7, 10)	(4, 10)	(14, 10)	(12, 10)	(6, 10)
R_6	(15, 1)	(14, 1)	(6, 1)	(6, 1)	(10, 1)
R_7	(4, 1)	(10, 1)	(10, 1)	(10, 1)	(9, 1)

t2	$Task_6$	$Task_7$	$Task_8$	$Task_9$
R_1	(2, 1)	(11, 1)	(3, 1)	(8, 1)
R_2	(5, 1)	(6, 1)	(1, 1)	(9, 1)
R_3	(7, 1)	(4, 1)	(10, 1)	(2, 1)
R_4	(8, 1)	(10, 1)	(7, 1)	(5, 1)
R_5	(7, 10)	(4, 10)	(14, 10)	(12, 10)

Figure 9: Quantified benefits of resource-task matching.

maximization. Since the problem is NP-hard, we use the greedy Hungarian algorithm [39] to match resources and tasks in pairs. The result is as follows:

(i) t_1: (R_1, Task2), (R_2, Task4), (R_3, Task5), (R_6, Task3), (R_7, Task1).

(ii) t_2: (R_1, Task6), (R_2, Task8), (R_3, Task7), (R_4, Task9).

In addition, we employ the traditional greedy algorithm [40] for resource scheduling. The algorithm only considers the maximization of revenue. The following matching groups are obtained at time t_1 and t_2, respectively. The result is as follows:

(i) t_1: (R_2, Task3), (R_4, Task5), (R_5, Task2), (R_6, Task4), (R_7, Task1)

(ii) t_2: (R_1, Task6), (R_2, Task8), (R_3, Task9), (R_5, Task7)

The results show that HMIPlatform can easily construct the spatiotemporal information and QoS constraints of the real environment in CPHS, flexibly customize the resource scheduling algorithm, and realize the application-independent resource scheduling and management.

6. Related Work

6.1. Resource Modeling in CPHS. For CPHS with multiple and heterogeneous resources, designing a set of semantic description and quantification rules of resources is a prerequisite for resource management and scheduling. Many studies have been done to model resources in the CPHS loop for conceptual common understanding, but they focus on application-specific domains. A novel service model for service discovery is proposed in [41] to improve the forwarding efficiency of the network layer. Li et al. [42] proposed a task scheduling model to reduce computational resource overhead for IoT-cloud. To implement a robot-based cyber-physical system, a set of customized semantics is proposed and integrated into the system design in [43]. However, even in specific application areas, the current work is still very poor in modeling and instantiation of heterogeneous resources and their interactions. In this paper, we propose HMIModel, a hybrid resource model for CPHS, to achieve a general consensus of CPHS concepts. On this basis, we design an orchestration tool to generate resource requirement prototype to quickly customize resource requirements.

6.2. Resource Scheduling in CPHS. Although a lot of efforts have been made to improve the productivity of CPHS applications, achieving full heterogeneous resource scheduling in CPHS is still a huge challenge for two reasons. One is the lack of quantitative modeling methods for heterogeneous resources, including physical environments (e.g., space-time elements and human participation); the other is the lack of a general scheduling platform, resulting in tight coupling between the resource scheduling and the application itself, which is difficult to expand and leads to high iteration costs. Research in [44] concentrates on the field of automatic driving, which solves the fault tolerance of the system and improves the efficiency of the processor through a tiered-based heuristic task assignment method. The work in [45] protects data integrity against, for example, interference attacks, by executing operational instructions on the traditional physical infrastructure. Electric vehicle charging scheduling is solved using a combination of Lyapunov optimization and Markov chain Monte Carlo sampling techniques in [46]. Research in [47] proposed a method of task time allocation and rewards for charging advertisement of plug-in electric vehicles (PEVs). Research in [48, 49] improves the efficiency of task offloading for limited RPs in MEC and IoT, respectively. A QoS-aware VM scheduling method is proposed in cloud-based CPHS in [50]. In this paper, we design and implement an application-independent CPHS resource scheduling platform that uses the recent software agent and lightweight container technology. The platform provides an instantiated environment for all kinds of resources on the basis of our hybrid resource model, while supporting plug-and-play for different scheduling strategies.

7. Conclusion

From the perspective of resource provision, our paper focuses on the roles of different resource providers in CPHS resource scheduling and the method of cross-domain cooperation among them and proposes a complete domain-independent CPHS application scheduling framework. We first propose a hybrid resource abstract model for heterogeneous RPs in clouds, networks, terminals, and humans to achieve the conceptual consensus on CPHS. Meanwhile, we design an orchestration tool based on this model to realize the rapid customization of the resource requirement prototype for a CPHS application. More importantly, we design and implement a common CPHS scheduling platform, which realizes the discovery and maintenance of heterogeneous resources and the customization of multiobjective optimization resource scheduling strategies under the multisource collaboration. We have injected a personalized

scheduling strategy into this platform, and the resource scheduling results are in line with expectations. The platform can greatly improve the iterative efficiency of the CPHS resource scheduling algorithm. In the future, we will improve our scheduling framework by adapting to a variety of CPHS applications and simultaneously updating our existing open-source projects.

Data Availability

The source code data used to support the findings of this study are included within the article.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This work was supported by the National Key Research and Development Program of China (2018YFB1004805).

References

[1] D. Sahinel, C. Akpolat, O. C. Gorür, F. Sivrikaya, and S. Albayrak, "Human modeling and interaction in cyber-physical systems: a reference framework," *Journal of Manufacturing Systems*, vol. 59, pp. 367–385, 2021.

[2] J. Zeng, L. T. Yang, M. Lin, H. Ning, and J. Ma, "A survey: cyber-physical-social systems and their system-level design methodology," *Future Generation Computer Systems*, vol. 105, pp. 1028–1042, 2020.

[3] S. Latif, Z. Idrees, J. Ahmad, L. Zheng, and Z. Zou, "A blockchain-based architecture for secure and trustworthy operations in the industrial Internet of Things," *Journal of Industrial Information Integration*, vol. 21, Article ID 100190, 2021.

[4] J. Quintas, P. Menezes, and J. Dias, "Information model and architecture specification for context awareness interaction decision support in cyber-physical human–machine systems," *IEEE Transactions on Human-Machine Systems*, vol. 47, pp. 323–331, 2016.

[5] Y. Zhou, F. R. Yu, J. Chen, and Y. Kuo, "Cyber-physical-social systems: a state-of-the-art survey, challenges and opportunities," *IEEE Communications Surveys & Tutorials*, vol. 22, no. 1, pp. 389–425, 2020.

[6] H. Li, C. Wu, Z. Li, and F. C. Lau, "Profit-maximizing virtual machine trading in a federation of selfish clouds," in *Proceedings of the 2013-IEEE INFOCOM*, pp. 25–29, Turin, Italy, April 2013.

[7] Y. Zhao, R. Calheiros, G. Gange, J. Bailey, and R. Sinnott, "SLA-based profit optimization resource scheduling for big data analytics-as-a-service platforms in cloud computing environments," *IEEE Transactions on Cloud Computing*, vol. 9, no. 3, pp. 1236–1253, 2018.

[8] W. Wei, X. Fan, H. Song, X. Fan, and J. Yang, "Imperfect information dynamic stackelberg game based resource allocation using hidden Markov for cloud computing," *IEEE Transactions on Services Computing*, vol. 11, pp. 78–89, 2016.

[9] B. P. Rimal and M. Maier, "Workflow scheduling in multi-tenant cloud computing environments," *TPDS*, vol. 28, pp. 290–304, 2016.

[10] N. Hassan, S. Gillani, E. Ahmed, I. Yaqoob, and M. Imran, "The role of edge computing in internet of things," *IEEE Communications Magazine*, vol. 56, no. 11, pp. 110–115, 2018.

[11] M. Liu, F. R. Yu, Y. Teng, V. C. Leung, and M. Song, "Distributed resource allocation in blockchain-based video streaming systems with mobile edge computing," *IEEE Transactions on Wireless Communications*, vol. 18, pp. 695–708, 2018.

[12] H. Tan, Z. Han, X. Y. Li, and F. C. Lau, "Online job dispatching and scheduling in edge-clouds," in *Proceedings of the IEEE INFOCOM-IEEE Conference on Computer Communications*, pp. 1–9, Atlanta, GA, USA, May 2017.

[13] G. Zhang, W. Zhang, Y. Cao, D. Li, and L. Wang, "Energy-delay tradeoff for dynamic offloading in mobile-edge computing system with energy harvesting devices," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 10, pp. 4642–4655, 2018.

[14] X. Ma, S. Zhang, W. Li, P. Zhang, C. Lin, and X. Shen, "Cost-efficient workload scheduling in cloud assisted mobile edge computing," in *Proceedings of the 2017 IEEE/ACM 25th International Symposium on Quality of Service (IWQoS)*, pp. 1–10, Vilanovaila Geltrú, Spain, June 2017.

[15] H. A. Alameddine, S. Sharafeddine, S. Sebbah, S. Ayoubi, and C. Assi, "Dynamic task offloading and scheduling for low-latency IoT services in multi-access edge computing," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 3, pp. 668–682, 2019.

[16] Y. Xue and X. Yu, "Beyond smart grid-cyber-physical-social system in energy future [point of view]," *Proceedings of the IEEE*, vol. 105, no. 12, pp. 2290–2292, 2017.

[17] S. Han, "Parallel smart car: CPSS-based networked self-driving car," in *Proceedings of the China Automation Conference (CAC) International Intelligent Manufacturing Innovative Conference (CIMIC)*, 2017.

[18] F.-Y. Wang, "Parallel economics: a new supply-demand philosophy via parallel organizations and parallel management," *IEEE Transactions on Computational Social Systems*, vol. 7, no. 4, pp. 840–848, 2020.

[19] G. Cabour, E. Ledoux, and S. Bassetto, "A work-centered approach for cyber-physical-social system design: applications in aerospace industrial inspection," 2021, https://arxiv.org/abs/2101.05385.

[20] D. Bernstein, "Containers and cloud: from LXC to docker to kubernetes," *IEEE Cloud Computing*, vol. 1, no. 3, pp. 81–84, 2014.

[21] J. Zhu, W. Xu, W. Tianyu, and C. Hu, "Hybrid resource orchestration and scheduling for cyber-physical-human systems," in *Proceedings of the IEEE 22nd International Conference on High Performance Computing and Communications; IEEE 18th International Conference on Smart City; IEEE 6th International Conference on Data Science and Systems (HPCC/SmartCity/DSS)*, pp. 1126–1133, Yanuca Island, Cuvu, Fiji, December 2020.

[22] SCP in Mulan Community. https://toscode.gitee.com/SCP%202018/.

[23] Q. Zhang, L. Cheng, and R. Boutaba, "Cloud computing: state-of-the-art and research challenges," *Journal of internet services and applications*, vol. 1, no. 1, pp. 7–18, 2010.

[24] P. Lama, S. Wang, X. Zhou, and D. Cheng, "Performance isolation of data-intensive scale-out applications in a multi-tenant cloud," in *Proceedings of the IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, pp. 85–94, IEEE, Vancouver, BC, Canada, May 2018.

[25] X. Xu, R. Mo, F. Dai, W. Lin, S. Wan, and W. Dou, "Dynamic resource provisioning with fault tolerance for data-intensive meteorological workflows in cloud," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 9, 2019.

[26] X. Hu, K.-K. Wong, K. Yang, and Z. Zheng, "UAV-assisted relaying and edge computing: scheduling and trajectory optimization," *IEEE Transactions on Wireless Communications*, vol. 18, no. 10, pp. 4738–4752, 2019.

[27] H. Li, K. Ota, and M. Dong, "Learning IoT in edge: deep learning for the internet of things with edge computing," *IEEE network*, vol. 32, no. 1, pp. 96–101, 2018.

[28] T. Q. Dinh, J. Tang, Q. D. La, and T. Q. Quek, "Offloading in mobile edge computing: task allocation and computational frequency scaling," *IEEE Transactions on Communications*, vol. 65, pp. 3571–3584, 2017.

[29] M. Chen and Y. Hao, "Task offloading for mobile edge computing in software defined ultra-dense network," *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 3, pp. 587–597, 2018.

[30] Maxgraph2. https://jgraph.github.io/mxgraph.

[31] JSON2.2.0. https://rubygems.org/gems/json/versions/2.2.0.

[32] Protocol Buffers. https://developers.google.cn/protocol-buffers/.

[33] B. Burns, J. Beda, and K. Hightower, *Kubernetes: Up and Running: Dive into the Future of Infrastructure*, O'Reilly Media, Newton, Massachusetts, USA, 2019.

[34] Etcd. https://etcd.io/.

[35] Netty. https://github.com/netty/netty.

[36] IndoorGML. http://www.indoorgml.net/.

[37] Kryo. https://github.com/EsotericSoftware/kryo/.

[38] H.-K. Kang and K.-J. Li, "A standard indoor spatial data model-OGC IndoorGML and implementation approaches," *ISPRS International Journal of Geo-Information*, vol. 6, no. 4, p. 116, 2017.

[39] G. A. Mills-Tettey, A. Stentz, and M. B. Dias, "The dynamic Hungarian algorithm for the assignment problem with changing costs,"Technical Report CMU-RI-TR-07-27 2007, Pittsburgh, PA.

[40] D. G. Amalarethinam and T. L. A. Beena, "Cloud scheduling-a survey," *International Journal of Computer Application*, vol. 97, 2014.

[41] C. Cabrera, G. White, A. Palade, and S. Clarke, "The right service at the right place: a service model for smart cities," in *Proceedings of the 2018 IEEE International Conference on Pervasive Computing and Communications (PerCom)*, pp. 1–10, IEEE, Athens, Greece, March 2018.

[42] W. Li, K. Liao, Q. He, and Y. Xia, "Performance-aware cost-effective resource provisioning for future grid IoT-cloud system," *Journal of Energy Engineering*, vol. 145, no. 5, Article ID 04019016, 2019.

[43] F. Li, J. Wan, P. Zhang, D. Li, D. Zhang, and K. Zhou, "Usage-specific semantic integration for cyber-physical robot systems," *ACM Transactions on Embedded Computing Systems*, vol. 15, no. 3, pp. 1–20, 2016.

[44] A. Bhat, S. Samii, and R. Rajkumar, "Practical task allocation for software fault-tolerance and its implementation in embedded automotive systems," *Real-Time Systems*, vol. 55, no. 4, pp. 889–924, 2019.

[45] K. Gai, L. Qiu, M. Chen, H. Zhao, and M. Qiu, "SA-EAST," *ACM Transactions on Embedded Computing Systems*, vol. 16, no. 2, pp. 1–22, 2017.

[46] Q. Li and R. Negi, "Distributed scheduling in cyber-physical systems: the case of coordinated electric vehicle charging," in *Proceedings of the 2011 IEEE GLOBECOM Workshops (GC Wkshps)*, pp. 1183–1187, IEEE, Houston, TX, USA, December 2011.

[47] M. Li, J. Gao, L. Zhao, and X. S. Shen, "Task time allocation and reward scheme for PEV charging station advertising," in *Proceedings of the IEEE International Conference on Communications*, pp. 1–6, Shanghai, China, May 2019.

[48] S. Luo, Y. Wen, W. Xu, and D. Puthal, "Adaptive task offloading auction for industrial CPS in mobile edge computing," *IEEE Access*, vol. 7, pp. 169055–169065, 2019.

[49] R. Ezhilarasie, M. S. Reddy, and A. Umamakeswari, "A new hybrid adaptive GA-PSO computation offloading algorithm for IoT and CPS context application," *Journal of Intelligent & Fuzzy Systems*, vol. 36, no. 5, pp. 4105–4113, 2019.

[50] L. Qi, Y. Chen, Y. Yuan, S. Fu, X. Zhang, and X. Xu, "A QoS-aware virtual machine scheduling method for energy conservation in cloud-based cyber-physical systems," *World Wide Web*, vol. 23, no. 2, pp. 1275–1297, 2020.