Front. Mech. Eng. 2019, 14(3): 320–331
https://doi.org/10.1007/s11465-019-0542-1

RESEARCH ARTICLE

Shiyong YIN, Jinsong BAO, Jie LI, Jie ZHANG

Real-time task processing method based on edge computing for spinning CPS

Abstract Spinning production is a typical continuous manufacturing process characterized by high speed and uncertain dynamics. Each manufacturing unit in spinning production produces various real-time tasks, which may affect production efficiency and yarn quality if not processed in time. This paper presents an edge computing-based method that is different from traditional centralized cloud computation because its decentralization characteristics meet the high-speed and high-response requirements of yarn production. Edge computing nodes, real-time tasks, and edge computing resources are defined. A system model is established, and a real-time task processing method is proposed for the edge computing scenario. Experimental results indicate that the proposed real-time task processing method based on edge computing can effectively solve the delay problem of real-time task processing in spinning cyber-physical systems, save bandwidth, and enhance the security of task transmission.

Keywords edge computing, real-time task, scheduling, CPS, spinning

1 Introduction

Spinning is the process of transferring fiber from a disorganized state to an orderly arranged state in the longitudinal direction. It is a typical continuous mass production multi-process with high speed and dynamic characteristics. The core of spinning smart factories is the cyber-physical system (CPS). The physical space of spinning CPS is composed of screws, machines, materials, methods, and rings involved in yarn production in spinning production plants. The information space mirrors the physical space. All types of data are processed in the physical space and continuously and iteratively optimized into a base that provides knowledge on spinning design, process, manufacturing, and testing. The information space is considerably important in improving the efficiency of yarn production, tracking yarn quality, reducing production costs, and other aspects of yarn production. Physical and information spaces interact through the industrial Internet of Things (IoT) constructed by various intelligent sensors, radio frequency identification, Wi-Fi, tags, and so on. Figure 1 shows a simplified structure of a spinning CPS.

Spinning strictly requires a real-time property. A task must be handled promptly; otherwise, the spin process may be interrupted and production will become stagnant, which cause huge losses to the enterprise. The tasks in spinning CPS can be summarized in several cases, which are listed in Table 1.

The existing information space of spinning CPS is typically deployed in remote cloud centers due to its strong computational, analytical, and processing capabilities to perform data processing and storage in spinning CPS in a resource-focused manner. However, with the continuous expansion of the production scale and the transformation and upgrading of enterprises, the amount of production equipment required in spinning considerably increases together with various intelligent sensors, detectors, embedded systems, or smart objects. This increment requires remote spinning CPS cloud centers to meet two critical requirements.

Real-time requirements: 1) First, a large amount of data is produced in the spinning CPS physical space within a short time and transmitted to the cloud center as tasks waiting for processing. Given that the amount of real-time data from the entire manufacturing system is extremely large, such data can easily disrupt the industrial network. Reliability cannot be guaranteed in this situation. 2) Second, the cloud computing center is far from the manufacturing site, and the bandwidth for communication

Received February 19, 2019; accepted May 4, 2019

Shiyong YIN, Jinsong BAO (✉), Jie LI, Jie ZHANG
College of Mechanical Engineering, Donghua University, Shanghai 201620, China
E-mail: bao@dhu.edu.cn

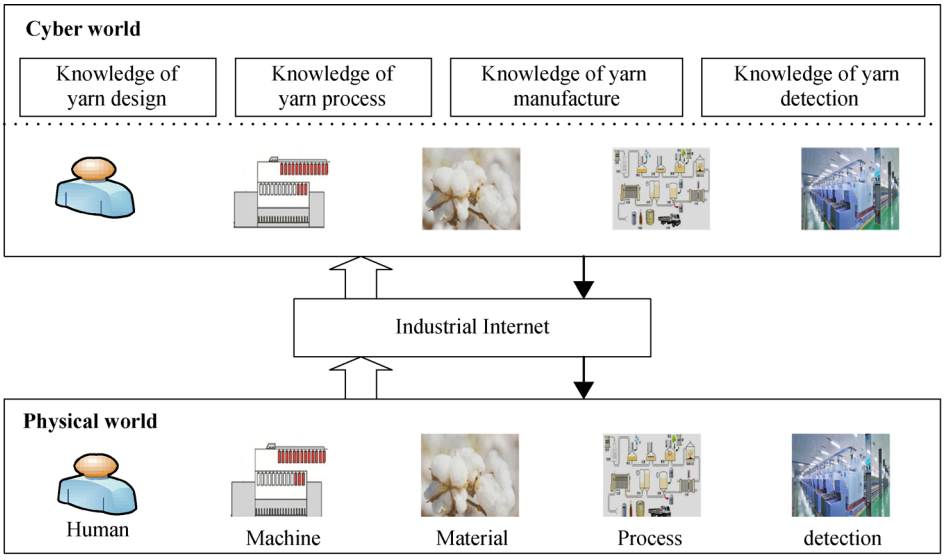

Fig. 1 Diagram of spinning CPS

Table 1 Common types of tasks and examples in spinning CPS

No.	Task category	Example
1	Tasks related to personnel	Manually issue control signals, start and stop the machine, etc.
2	Tasks related to equipment	Equipment failure, equipment preventive maintenance tips, etc.
3	Tasks related to the item or work in progress	Added raw materials, products with anomalies, etc.
4	Tasks related to the process	Parameter adjustment, process switching, etc.
5	Tasks related to the environment	Adjustment of the control temperature, humidity, etc.

is limited. These factors delay real-time task processing and cannot meet real-time requirements.

Security requirements: Risks involving data and order may arise during the transmission of real-time tasks between production equipment and the cloud center. This risk may cause interference or leakage and is thus a security issue.

A method based on edge computing is proposed in this study to solve real-time tasks and meet the security requirements of task processing in spinning CPS. Edge computing node (ECN) and real-time task models are established. The task scale threshold is set. Large-scale tasks are separated for remote cloud center processing to compensate for the lack of real-time processing and ensure the security of centralized cloud computing.

Published studies are reviewed in Section 2 of this paper. The development of the real-time task processing-related model and the algorithm design are illustrated in Section 3. A case study is presented in Section 4, and the conclusion is provided in Section 5.

2 Related work

Many scholars have conducted detailed research on real-time CPS data processing and resource allocation. A real-time data center was introduced in a previous study to store, retrieve, and process large amounts of real-time data efficiently [1]. Real-time and parallel data processing are supported to balance the loads of the CPS-aware data center. However, this approach is inapplicable to CPS manufacturing loads. Considering the importance of the packet retransmission protocol decision, Ref. [2] proposed a new congestion control mechanism for CPS data collection to minimize the estimation and reconstruction errors of physical phenomena. Reference [3] developed a scheduling strategy with minimum energy efficiency on the basis of wireless network topology while considering the sleep scheduling of wireless nodes and processor execution models simultaneously. However, the strategy focuses on the single-hop network mode and is unsuitable for CPS application. To minimize the energy consumption of the sensor network, Ref. [4] studied linear sensor setting problems in CPS and solved power configuration issues through mixed-integer linear programming. An integrity-protecting cluster-based private data aggregation protocol was proposed in Ref. [5] to resolve data aggregation and protect the privacy and integrity of data during CPS data aggregation. Reference [6] proposed a periodic fault-tolerant CPS task model to achieve CPS stability, and a

new scheduling mechanism was developed to prove the practicability of this model. The model can enhance the stability and effectiveness of CPS and reduce operating costs. A CPS control scheduling algorithm was designed in Ref. [7] for the CPS task scheduling problem constrained by the feedback control rule. The algorithm achieves balance between stable scheduling and schedulable power consumption. A new task model called rhythm task was presented in Ref. [8] to deal with the problem of combining the traditional periodic task model and ordinary schedule for CPS task processing. A concept of buffer time was introduced in Ref. [9] to enhance traditional preemptive task scheduling, improve the performance of real-time task scheduling and utilization of system resources in the physical network in the systems, and reduce the time of task switching. A dynamic multi-priority scheduling strategy for CPS based on large-scale sensor networks was proposed in Ref. [10] to meet the diversity requirements of CPS tasks, which traditional scheduling algorithms cannot achieve. A rapid and extensible static sequential scheduling method was proposed in Ref. [11] for applications with rigid waiting time requirements and fixed binding on multiprocessor platforms. This method makes scheduling decisions on the basis of new metrics to find feasible schedules that can meet the time requirements as quickly as possible. Given that preemptive scheduling easily leads to frequent task switching and affects real-time CPS task issues, a real-time scheduling algorithm based on the reservation model was proposed in Ref. [12]. Task switching times were reduced, and the real-time performance of CPS was improved by setting the threshold of relaxation time and the protection model. A polynomial-time optimal data retrieval algorithm was proposed in Ref. [13] for the multi-interval availability-constrained fresh data retrieval problem of CPS. A novel control decision structure based on the cloud-supported CPS concept for process manufacturing was proposed in Ref. [14].

The complexity of a data center that consumes energy results in many difficulties during resource management. A macro resource management level was proposed in Ref. [15] to allocate energy in a coordinated manner. A new data center control strategy was proposed in Ref. [16] as a replacement for traditional methods of controlling network and physical resources independently. The proposed strategy can effectively manage different processing interfaces of a data center and assign tasks to ensure maximum service quality and minimum energy consumption. A method based on resource scheduling was developed in Ref. [17] by using the queue model to implement constant revision of the scheduling policy while considering objectivity and data locality simultaneously. Reference [18] introduced a resource-based scheduling method based on classification to achieve balance between the utilization and cost benefits of equipment. The method implements priority scheduling of suitable resources while avoiding task interference. Meanwhile, an online and scalable resource scheduling method that uses collaborative filtering technology to solve the resource allocation issue in online application was proposed in Ref. [19]. The method can schedule resources rapidly and accurately and deals with heterogeneity and interference in multiple shared resources. An accurate power model in data centers was proposed in Ref. [20] for time-constrained servers in cloud computing. Reference [21] developed an agent-based server system approach that improves the resource sharing between heterogeneous wireless sensor networks (WSNs) in IoT/CPS providers.

Given that technologies related to cloud computing cannot efficiently handle massive amounts of data from devices located at the edge of networks, computing models continue to be developed; these include micro-data centers [22], cloud-sea computing for humans and subsystems facing the physical world [23], cloud integration sensor networking, peer-to-peer networking, network virtualization, configuration management technology for fog computing [24–26], and cloudlet. Cloudlet focuses on business logic to connect cloud mobile devices and cloud servers in mobile cloud computing [27] and is considered to be the key to reduce the mobile core utilization and latency of mobile terminal users at the edge of mobile computing [28–31]. These technologies enable computing on the edge of networks and represent the development of executing edge computing of downstream data-represented cloud services and upstream data-represented IoT services [32].

Research on CPS in manufacturing has been extensively conducted, but only a few studies on the processing of real-time tasks in CPS are available. Edge computing is a new type of computing mode that provides near-end services by means of network, computing, storage, and other technologies on the production side. Resources, timeliness, and other factors are often considered in industrial production, and edge computing can benefit real-time task processing.

3 Modeling and algorithm design

3.1 Edge computing-based intelligent spinning manufacturing CPS

Figure 2 shows the framework of the centralized computing mode in spinning CPS. Under this framework, the real-time tasks produced by physical entity of spinning CPS are transmitted to the cloud center through industrial Internet. The tasks are subsequently processed to become knowledge, and intelligent decisions are made to control the physical entities.

The disadvantages of the centralized CPS system are prominent. The system can satisfy real-time task processing requirements in a high-speed production environment,

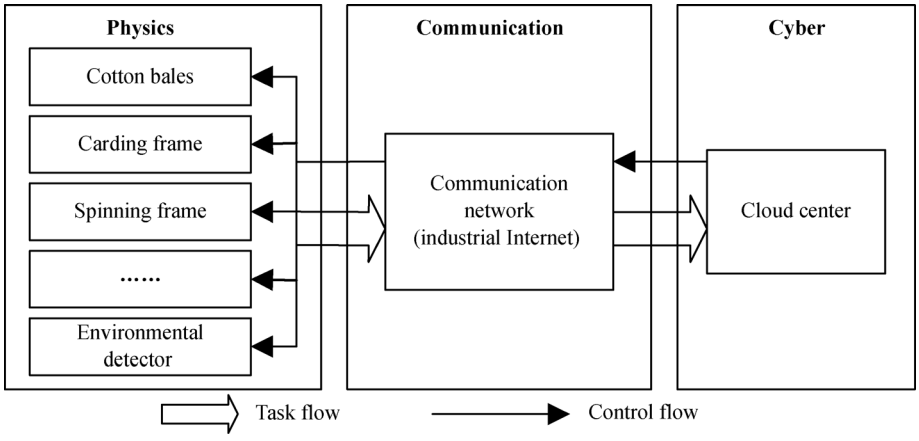

Fig. 2 Architecture of spinning CPS in the centralized computing mode

but security is not guaranteed. In this study, the functions of the cloud center are transferred near the physical entity. For the spinning manufacturing feature, two modes performing edge computing are designed, and their core is a distributed ECN, as illustrated in Fig. 3. Digital twin focuses on the establishment of a completely virtualized digital object [33,34], which is a CPS unit with complete features. Edge computing is a subset of the digital twin and focuses on finite task computation and communication.

1-1 ECN: Each physical entity has its own ECN. The carding frame corresponds to the computing node on the edge of this frame, the spinning frame corresponds to the computing node on the edge of this frame, and so on. The communication and negotiation mechanisms between ECNs is presented in Fig. 3(a).

n-1 ECN: Multiple physical entities have a common ECN and require few computing nodes installed at the side. However, this condition increases the complexity of calculation and scheduling, as shown in Fig. 3(b).

The mode adopted depends on the specific application scenario. For a high-speed ring-spinning workshop, the 1-1 ECN mode can be selected in the carding process and the n-1 ECN mode in the drawing–roving–spinning process to achieve a combination of these modes. For a single ECN, tasks can be time-sequence real-time tasks from a single entity or time-sequence real-time tasks from multiple entities.

3.2 Constitution of spinning ECN

A single spinning ECN includes the basic elements of the manufacturing unit, such as tasks and process, resource, and quality requirements, as depicted in Fig. 4.

Task agent: It detects the scale of tasks to determine whether to drive directly to the cloud center or to the buffer queue.

Task buffer: It accepts related tasks that satisfy the requirements.

Task scheduling: It schedules tasks in task buffers.

Resource monitoring: It monitors local edge node, external edge node, and cloud center resources through the interaction interface to provide a fundamental basis for resource allocation.

Resource allocation: It allocates resources for scheduled tasks.

Communication and interface: They serve as the channel for local ECNs to interact with the external nodes and to monitor the external node resource situation being monitored by other external nodes.

Production parameter reconfiguration: It is implemented to dynamically reconfigure the production process parameters according to the resources allocated and the actual needs of related task processing.

For simplicity, the following assumptions are made:

Assumption 1: Every single physical entity in CPS only commits a real-time task at a moment; the real-time task is independent and inseparable and can only be performed on one ECN;

Assumption 2: ECNs must be able to handle any single real-time task from the buffer queue;

Assumption 3: The cloud center can deal with all tasks, that is, although a real-time task is delayed due to the inability to be processed by the ECN, the task can be processed when transmitted to the cloud center.

The computing capability of ECNs is limited compared with that of cloud centers. For example, when the scale of a task is too large for an ECN and beyond its memory capacity or when the operation time of the node is longer than the longest delay time that the task can accept because of the performance of the central processing unit (CPU), the task needs to be transmitted to the cloud center. The task is then sent to a task agent in advance before the task is accepted into buffers by the ECN, where it is judged via a previously set threshold to decide whether to step into buffers or be transmitted to the cloud center for processing.

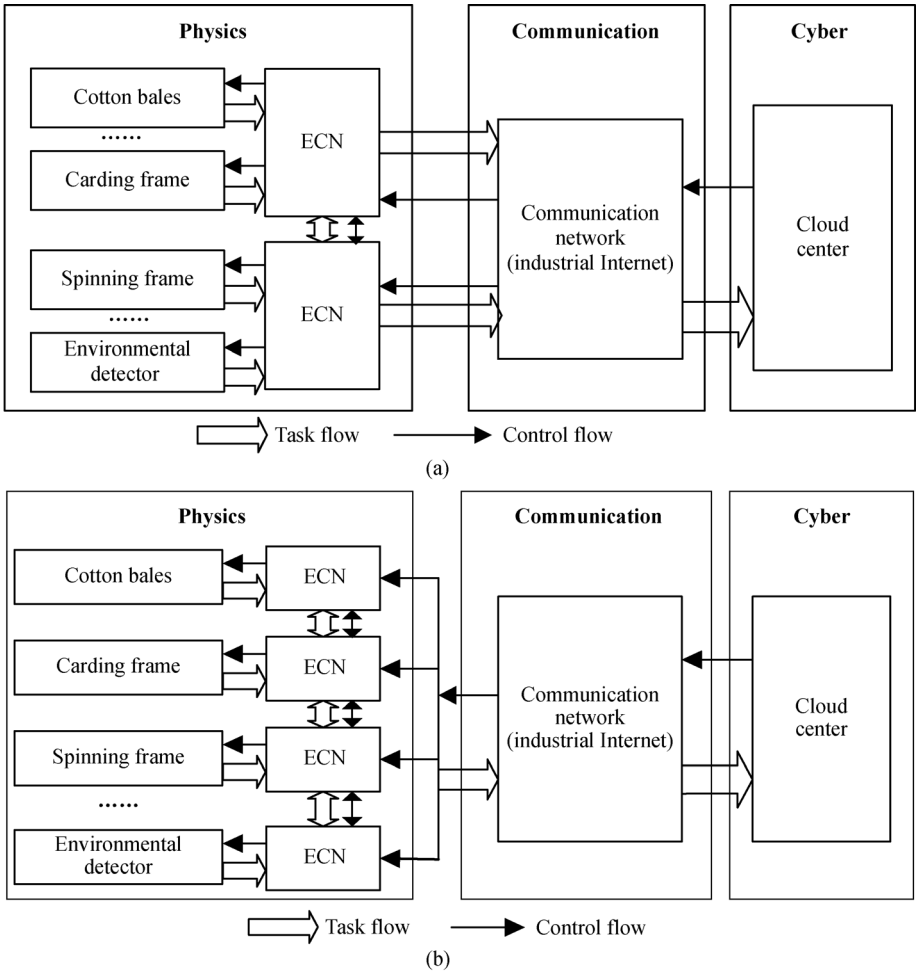

Fig. 3 Architecture of spinning CPS edge computing mode. (a) A unique ECN per entity; (b) ECN shared by multiple entities

3.3 Modeling

3.3.1 ECN model

The studied ECN is an intelligent agent connected with the physical entity in the CPS, and it can store, compute, and communicate.

Definition 1: ECN is defined as

$$e = (m,\ s,\ f^{\max},\ b,\ p^{\max},\ trTime,\ task\_agent,\ task\_buffer,\ CRpool,\ interaction\_window).$$

Definition 1 describes 10 tuples of ECN, in which m and *task_buffer* refer to the memory (GB) and real-time task buffer (MB), respectively, and are related to storage; s, $f^{\max}$, *task_agent*, and *CRpool* refer to the CPU's computing speed (MIPS, million instructions per second), maximum frequency (MHz), maximum power (W), real-time task agent, and computing resource pool, respectively, and are concerned with computing; b and *trTime* refer to the bandwidth (Mb/s) and time (s) to transmit data from a node to the cloud center, respectively, which pertain to transmission on the Internet; and *interaction_window* refers to the interface for the ECN to interact with external nodes.

Definition 2: A set of ECNs is given as

$$E = \{e_1,\ e_2,\ ...,\ e_n\},$$

in which n refers to the number of nodes. This definition describes the set of all ECNs of the entire spinning CPS physical space.

3.3.2 Resources of edge computing model

Definition 3: It pertains to the resource pool of ECN e_i. $CRpool_i = \{r_{i1}, r_{i2}, ..., r_{im}\}$, $i \in [1, n]$, in which m refers to the number of resource blocks of e_i.

Definition 3 describes the resource pool of any node e_i. This pool is the sum of all resources the node has, and r_{ik} ($k \in [0, m]$) refers to each resource block.

Definition 4: Resource block $r_{ik}=(m_{ik}, s_{ik}, f_{ik}, b_{ik}, p_{ik}, c_{ik})$, $i \in [1, n]$.

Definition 4 describes five tuples for a resource block of any ECN, in which m_{ik}, s_{ik}, f_{ik}, b_{ik}, and p_{ik} refer to the

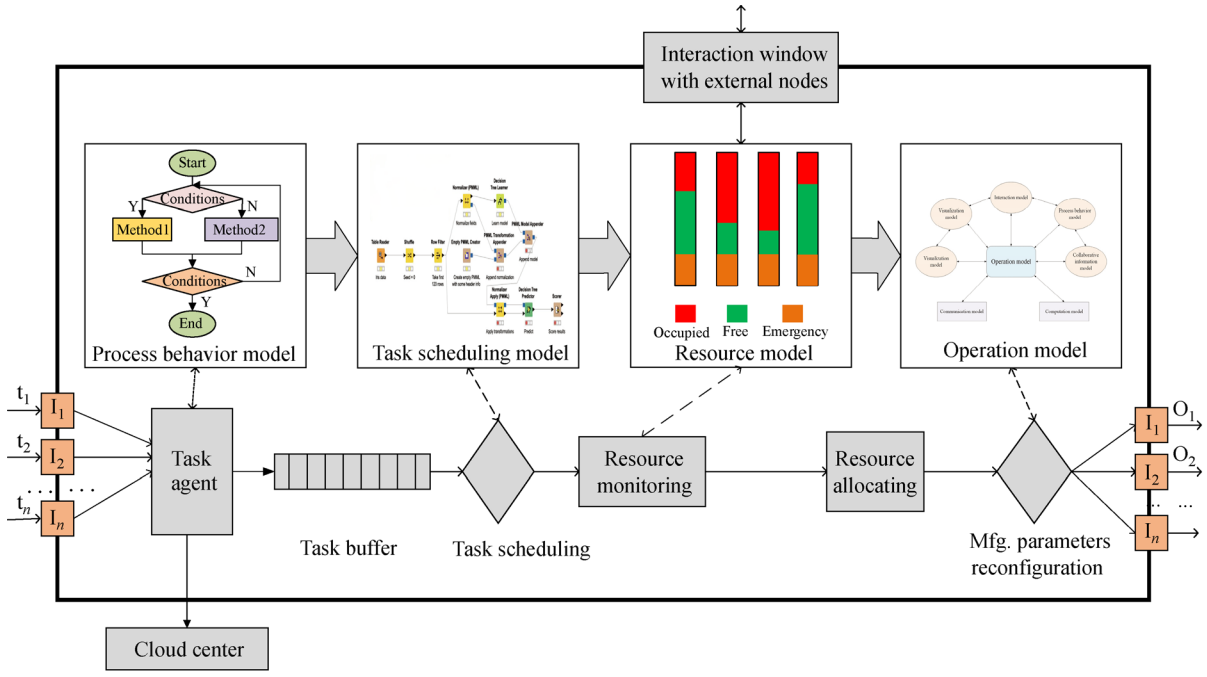

Fig. 4 Components and process logic of ECN. Mfg.: Manufacturing; O: Output; I: Input; t: Task

memory, CPU computing speed, CPU frequency, bandwidth, and power that e_i distributes to each resource block; c_{ik} refers to the state of the CPU, namely, occupied, ready, or idle.

3.3.3 Task model

In CPS, independent tasks are submitted independently by users.

Definition 5:

Task $t = (length, comeTime, startTime, endTime, runTime, turnAroundTime)$.

Definition 5 describes six tuples of a task, as follows: *length* (s) refers to the time length of a task; *comeTime* refers to the time for the task to arrive; *startTime* and *endTime* refer to the start and end times of a task, respectively; *runTime* refers to the operation time of a task; and *turnAroundTime* refers to the turnaround time of a task. The framework for describing a task is presented in Fig. 5.

Definition 6: Set of tasks $T = \{t_1, t_2, ..., t_p\}$.

Definition 6 describes the set of tasks that a node receives, and p refers to the number of tasks received at one moment.

Assuming that tasks numbered p–c are transmitted to the cloud center for processing because their scale is beyond

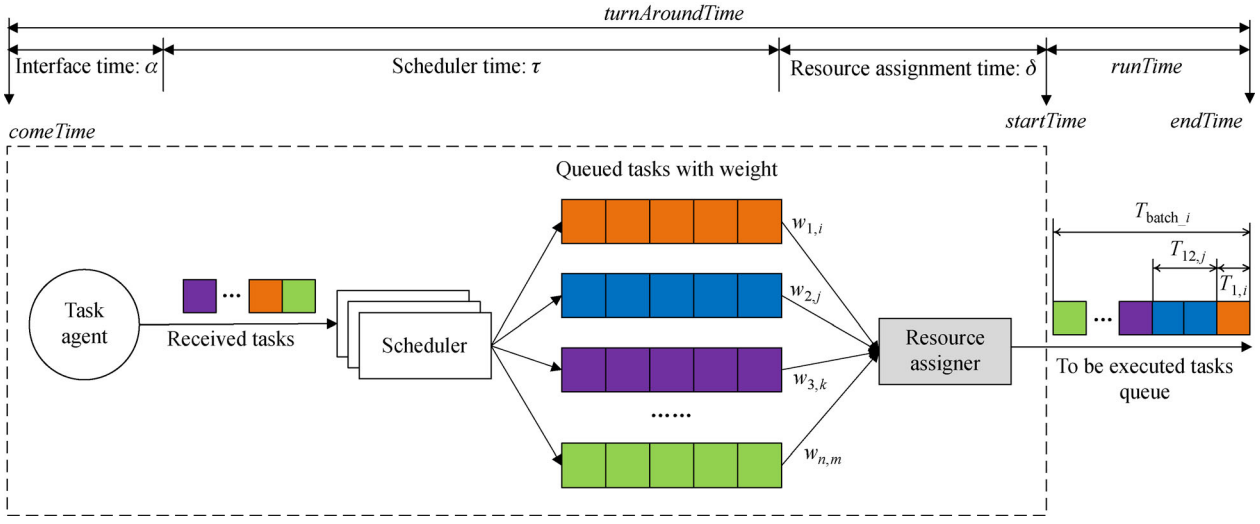

Fig. 5 Framework for describing a task

the computing capability of a node, other tasks numbered c are processed at ECNs.

The completion time of a task is

$$\text{time}_j = runTime_j + startTime_j - comeTime_j$$

$$= turnAroundTime_j.$$

The total time for all tasks processed at local nodes is

$$\text{total time} = \sum_{j=1}^{c} turnAroundTime_j.$$

3.4 Real-time task process based on edge computing

3.4.1 Real-time task process

The CPS real-time task process based on edge computing proceeds as follows.

Step 1: The task is accepted.

Step 2: Task agent: The task is processed according to the behavioral processing model. The scale of the task is checked. If the scale is greater than or equal to the set size threshold, then the task is transmitted directly to the cloud center. Otherwise, Step 3 is implemented.

Step 3: The task enters the task buffer and waits for task scheduling.

Step 4: Task scheduling is implemented according to the scheduling algorithm.

Step 5: Resources are allocated to the scheduled tasks according to the resource monitoring situation.

Step 6: The resource allocation results are produced.

3.4.2 Real-time task processing algorithm design

According to the process of real-time task processing, the algorithm design for real-time task processing is as follows:

Input: Real-time set of tasks $T = \{t_1, t_2, ..., t_p\}$; the set of ECNs $E = \{e_1, e_2, ..., e_n\}$; the set of available resource blocks of all ECNs r_{ik}; and the cloud center.

Output: The resource block allocated for each task.

(1) Task buffer = Null, taskScale = taskScaleMax;
(2) while new $t_i \in T$ do
(3) read tlength;
(4) if (tlength >= taskScale), transmit t_i to cloud center;
(5) else t_i step into Task buffer;
(6) end while
(7) while (Task buffer != Null) do
(8) scheme a task t_i;
(9) if (local resources meet the requirement of t_i), allocate resources to t_i
(10) else for other ECNs resources do
(11) if (resources meet the requirement of t_i), allocate resources to t_i; break;
(12) else t_i transmit to cloud center;
(13) end for
(14) end if
(15) end while

4 Experiment and discussion

4.1 Experiment design

This experiment is based on the real production environment of a spinning workshop under the CPS architecture. The workshop has 2 opening frames, 20 carding frames, 8 combing frames, 16 drawing frames, 13 roving frames, 30 spinning frames, and 13 winding frames.

According to the actual layout of the workshop, 1 opening frame, 4 carding frames, 2 combing frames, 4 drawing frames, 4 roving frames, 8 spinning frames, and 2 packing-stacking machines are selected as experimental machines and compared. The compared machines operate normally under the CPS architecture.

In accordance with the diagram illustrated in Fig. 6, the simulation environment is set up, and the same number of machines (the same number of object entities) is selected. ECN1 only connects one opening frame to realize the 1-1 ECN mode. ECN2 connects a part of the spinning frames, carding frames, and palletizer. ECN3 connects a part of the spinning and combing frames, and ECN4 connects the roving and drawing frames to realize the n-1 ECN mode. Four ECNs are interconnected through a bus then connected to the cloud center through a local server. Each ECN has three normal resource blocks (for use by this node task) and two spare resource blocks (available for local and neighbor nodes). The enterprise 2015 version of Microsoft C# language is selected for the simulation environment for the following reasons. (1) The .NET framework is well accepted and can be created for a portable code that runs on iOS, Android, and Windows tablets/phones, desktops, servers, and embedded systems. (2) Everything from the compiler to the core runtime is open source. The program runs in the following environment: Windows 10, Intel (R) Core™ i5-5200U 2.20 GHz CPU, and 8.0 GB of RAM. The application types of C# include Windows console, Windows forms, and ASP.NET web service applications. We select Windows forms application programming to implement our simulation. In the experiment, 15 common tasks, such as broken yarn joint, yarn cutting, and material requirement processing, are used for the simulation.

4.2 Experimental results and discussion

4.2.1 Experiment 1: Effect of different scheduling algorithms on real-time task processing

The 15 common tasks shown in Table 2 are used for

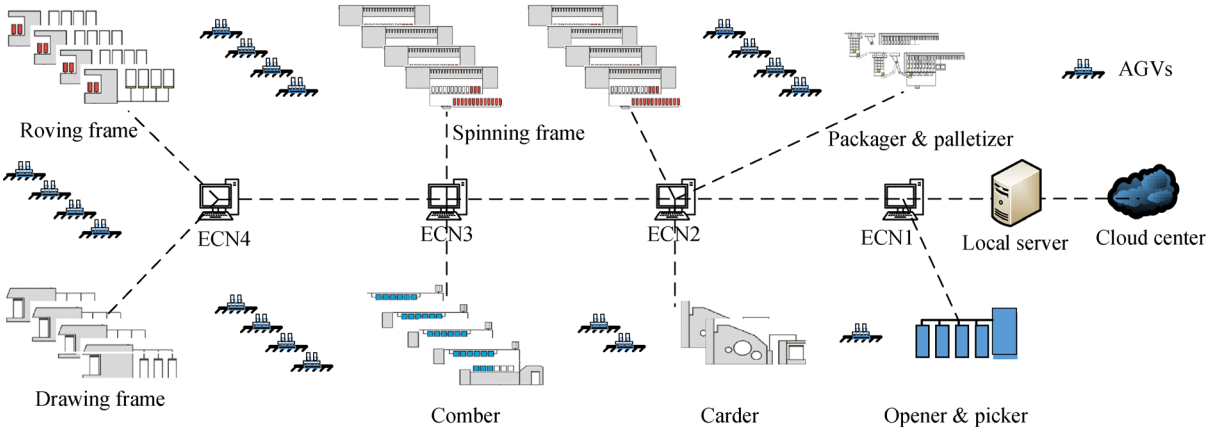

Fig. 6 Experimental diagram

Table 2 Resource allocation and execution of tasks

No.	Task name	Length/s	Interface time/ms	Scheduler time/s	Resource assignment time/ms	Resource allocation	Run time/s	Turnaround time/s
1	t1	240	5.7521	0	28.4385	ECN2.1	240.0526	240.0868
2	t2	901	5.9254	–	–	Cloud center	–	–
3	t3	285	5.7945	0	28.5256	ECN2.2	285.0512	285.0855
4	t4	320	5.7789	0	37.9196	ECN2.3	320.0651	320.1088
5	t5	240	5.6214	0	28.6335	ECN2.spare1	240.0489	240.0831
6	t6	350	5.8012	0	27.6193	ECN2.spare2	350.0693	350.1027
7	t7	210	5.6287	0	162.8069	ECN3.spare1	210.0446	210.2130
8	t8	180	5.6146	0	147.578	ECN4.spare1	180.0479	180.2011
9	t9	270	5.6812	0	150.4799	ECN1.spare1	270.0512	270.2073
10	t10	195	5.6349	0	147.3993	ECN1.spare2	195.0456	195.1987
11	t11	305	5.7871	0	162.5112	ECN4.spare1	305.0660	305.2343
12	t12	95	5.4243	0	153.7191	ECN1.spare2	95.0384	95.1975
13	t13	310	5.6479	82.3154	156.2954	ECN1.spare2	310.0732	392.5505
14	t14	25	5.2273	111.0159	24.9676	ECN2.1	25.0080	136.0469
15	t15	72	5.3394	108.3160	140.6214	ECN1.spare2	72.0025	180.4645

simulation. The task length threshold set in the experiment is 15 min. The resource allocation and execution of the 15 tasks received on ECN2 implemented with the "first in, first out" (FIFO) scheduling method are depicted in Table 2.

The t2 task, which has a length of 901 s and exceeds the set threshold, is transferred to the cloud center for processing. Therefore, the *runTime* and *turnAroundTime* of t2 are not calculated in this experiment.

When a new task arrives at ECN, the size of the task is detected by the task agent, that is, the interface time α of the task is calculated, as shown in the fourth column in Table 2. Thereafter, in accordance with the real-time monitoring results of the resource monitoring and quick allocation window, whether the task should enter the task buffer or not is determined (that is, whether task scheduling time τ needs to be calculated). When t1 and t3–t12 reach ECN2, the resources that can be directly allocated to the tasks are sufficient without calculating scheduler time τ (as shown in the fifth column in Table 2). However, when t13–t15 arrive, no resources are available for the time being, and these tasks are sent to the task buffer to wait for scheduling. The scheduling time τ of the task consists of the time waiting for scheduling in the task buffer and the scheduling time of the scheduling algorithm. The values of resource assignment time δ of the tasks in Table 2 are distributed in two intervals, [24, 38] and [147, 163]. The tasks (t1, t3–t6, t14) where the δ values fall in the first interval are assigned with local resources, and the tasks (t7–t13, t15) where the δ values fall in the second interval

are assigned with external resources. Therefore, the δ value of a task includes the time for assigning resources and for migrating from the local node to external resources. Figure 7 shows the resource assignment time δ for each task.

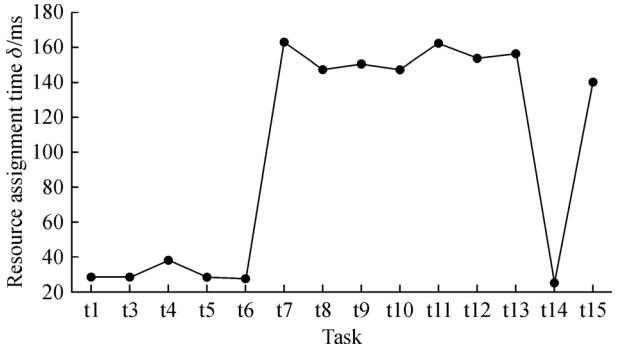

Fig. 7 Resource assignment time of tasks

ECN2 has a priority in using its own resource blocks; therefore, t1–t6 (t2 is transferred to the cloud center) allocate resource blocks 1–3 and spare resource blocks 1 and 2 in their order of arrival. When the t7 task arrives, all resources of ECN2 are occupied. In this case, the resources of the neighboring ECNs are determined, and that of ECN1 is determined first. However, ECN1's two spare resource blocks are also occupied. ECN3 is determined subsequently. At this time, ECN3's spare resource block 2 is in the free state, and ECN2 migrates t7 to this free state for processing. The resources for t8–t15 are allocated similarly. Table 2 shows that ECN3's spare resource block 2 and ECN4's spare resource block 2 have not been assigned to the tasks from ECN2, indicating that the two spare resource blocks are always occupied.

Figure 8 compares the resource configuration results of two different task scheduling algorithms, namely, highest response ratio next (HRRN) and FIFO. When tasks t1–t12 arrive, the resource blocks of ECN2 are sufficient, and waiting is not required. The allocation of resources is achieved in accordance with the arrival order of the tasks. When tasks t7–t15 arrive, the resources of the local edge node and its neighbor edge nodes are allocated, and no resources are available. Therefore, tasks t7–t15 enter the task buffer and wait. Owing to the differences between scheduling algorithms, tasks are scheduled from the buffer queue in a different order, and the allocated resources are also different. Nonetheless, from the perspective of delay rate, no effect exists, and the difference is due to the refreshing frequency of the program calculation.

Figure 9 shows a comparison of the delay rates in the three cases: normal operation of the spinning CPS architecture and edge computing task scheduling with FIFO and HRRN algorithms. The delay rate is calculated via the following formula.

$$\text{Delay rate} = \frac{\sum turnRoundTime - \sum length}{\sum length} \times 100\%,$$

where *turnRoundTime* is the turnaround time of the tasks and *length* is the length of the tasks.

The delay rate of the tasks is 26.528% under the spinning CPS architecture. After adding the ECNs in the simulation experiment, the total turnaround time of tasks is reduced, resulting in a reduced processing delay rate of the tasks. When FIFO and HRRN scheduling algorithms are utilized, the delay rate decreases by 62.63% and 62.75%, respectively, compared with the delay rate under the spinning CPS architecture. The real-time task processing of the spinning CPS with increased ECNs is advanced.

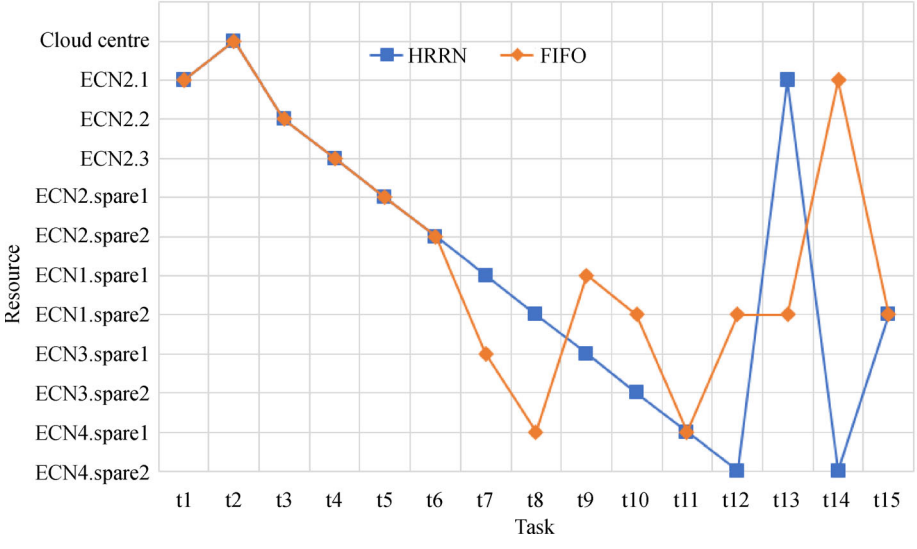

Fig. 8 Resource allocation results

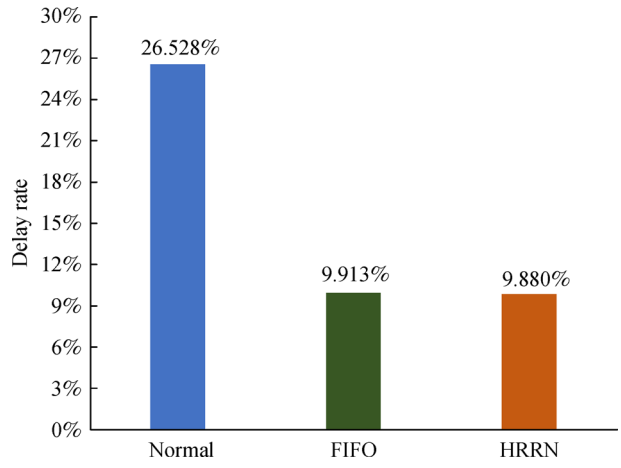

Fig. 9 Comparison of delay rates

4.2.2 Experiment 2: Impacts of task scale on real-time task processing

In this experiment, real-time tasks are submitted artificially by different physical entities of the spinning CPS for a period of time, and real-time tasks of the same scale (same length and number of tasks) are submitted in the simulation experiments. The numbers of tasks are 15, 50, 100, 200, and 500.

Figure 10 shows a comparison of the effects of different numbers of real-time tasks on the delay rate. The experimental results indicate that the delay rate of the edge computing mode of the spinning CPS is much lower than that of the computational mode. In the centralized computing mode, all real-time tasks are transmitted to the cloud center for processing. In the edge computing mode, the real-time tasks are processed at the edge nodes near the physical entities, which reduces the transmission time of the tasks. When the real-time tasks are increased to a certain number, the delay rate of the edge computing mode becomes stable, but that of the centralized computing mode increases. This result occurs because the resource of the ECN is limited, and the tasks are processed using the established scheduling algorithm. When the number of tasks in the centralized computing mode increases, the bandwidth of the transmission becomes limited, which exacerbates the congestion extent.

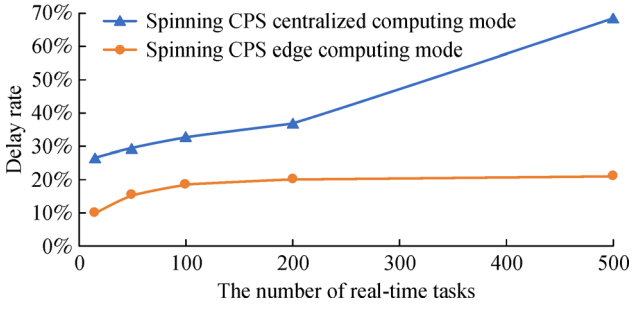

Fig. 10 Effect of number of tasks on the delay rate

4.2.3 Experimental summary

The three experiments show that the delay rate in the real-time task processing of spinning CPS can be reduced by using a reasonable configuration of the resources of ECNs and applying appropriate scheduling algorithms. The majority of the tasks are processed at the ECNs, which reduces various security risks that are incurred during the task transfer to the remote cloud center. In addition, the concurrent tasks transmitted to the remote cloud center are reduced so that sufficient bandwidth exists for transmitting the tasks that cannot be processed by the local ECNs to the cloud center; this also helps improve the effectiveness of real-time task processing.

5 Conclusions

This study established a model of tasks, ECNs, and edge computing resources and proposed a method for real-time task processing on the basis of edge computing to address the problem of processing delays, bandwidth shortage, and security risks in the process of real-time tasks being transferred to a remote cloud center. By setting the task scale threshold, the method allocates large-scale tasks to the cloud center and small-scale tasks to the local ECNs for processing while effectively solving the problem of delay in the processing of real-time tasks. This method can ensure sufficient bandwidth and avoid security risks during task transmission.

In industrial production, edge computing is expected to play an increasingly important role in particular cases, such as low/intermittent connectivity, instant analysis, and access to temporal data for real-time analysis. Edge computing can support predictive maintenance and resource or energy efficiency management. Therefore, edge computing can be applied to other industrial practices, especially in continuous manufacturing, such as weaving in textile manufacturing, chemical fiber production, and pharmaceutical and food industries.

We will conduct further research from two aspects: practical case study and development of the task scheduling algorithm. First, on the basis of the results of the experimental simulation, the effect of real-time ECNs on task processing is presented. After balancing the demands and costs of real-time task processing, physical entities in the spinning CPS can utilize ECN devices with reasonable configuration resources and then use the production data to verify the real-time effect of edge computing in handling CPS tasks. Second, the real-time performance of task processing is related to the task scheduling algorithm, resource quantity, and processing capability of resources.

Improved algorithms are required for task scheduling. In addition, the number of ECNs should be set based on the actual physical layout of the spinning workshop, and the processing capability of the resources should be rationally allocated.

Acknowledgements This work was supported in part by the Fundamental Research Funds for the Central Universities and the Graduate Student Innovation Fund of Donghua University (Grant No. CUSF-DH-D-2019096), the National Key Research and Development Plan of China (Grant No. 2017YFB1304000), and the National Natural Science Foundation of China (Grant No. 51475301).

References

1. Kang K D, Basaran C. Adaptive data replication for load sharing in a sensor data center. In: Proceedings of the 29th IEEE International Conference on Distributed Computing Systems Workshops. Montreal: IEEE, 2009, 20–25
2. Ahmadi H, Abdelzaher T, Gupta I. Congestion control for spatio-temporal data in cyber-physical systems. In: Proceedings of the 1st ACM/IEEE International Conference on Cyber-physical Systems. Stockholm: ACM, 2010, 89–98
3. Xue C J, Xing G L, Yuan H, et al. Joint sleep scheduling and mode assignment in wireless cyber-physical systems. In: Proceedings of the 29th IEEE International Conference on Distributed Computing Systems Workshops. Montreal: IEEE, 2009, 1–6
4. Guo Y F, Kong F X, Zhu D K, et al. Sensor placement for lifetime maximization in monitoring oil pipelines. In: Proceedings of the 1st ACM/IEEE International Conference on Cyber-physical Systems. Stockholm: ACM, 2010, 61–68
5. He W B, Liu X, Nguyen H, et al. A cluster-based protocol to enforce integrity and preserve privacy in data aggregation. In: Proceedings of IEEE International Conference on Distributed Computing Systems Workshops. Montreal: IEEE, 2009, 109–118
6. Lee J, Shin K G. Development and use of a new task model for cyber-physical systems: A real-time scheduling perspective. Journal of Systems and Software, 2017, 126(4): 45–56
7. Zhang F, Szwaykowska K, Wolf W, et al. Task scheduling for control oriented requirements for cyber-physical systems. In: Proceedings of IEEE Real-Time Systems Symposium. Barcelona: IEEE, 2008, 47–56
8. Kim J, Lakshmanan K, Rajkumar R. Rhythmic tasks: A new task model with continually varying periods for cyber-physical systems. In: Proceedings of International Conference on Cyber-Physical Systems. Beijing: IEEE, 2012, 55–64
9. Zhang J, Yang X D, Fan H B. An improved real-time task preemptive scheduling in cyber-physical systems. In: Proceedings of Chinese Control and Decision Conference. Chongqing, 2017, 5843–5848
10. Liu C Y, Zhang L C. Dynamic multi-priority scheduling for cyber-physical systems. Computer Science, 2015, 42(1): 28–32 (in Chinese)
11. Adyanthaya S, Geilen M, Basten T, et al. Fast multiprocessor scheduling with fixed task binding of large scale industrial cyber physical systems. In: Proceedings of Euromicro Conference on Digital System Design. Los Alamitos: IEEE, 2013, 979–988
12. Zhou B H, Yao D P. Research on reserved real-time scheduling approach for cyber and physical system. In: Proceedings of International Conference on Intelligent Networks and Intelligent Systems. Shenyang: IEEE, 2013, 62–65
13. Fu C, Wu P, Li M, et al. Real-time data retrieval with multiple availability intervals in CPS under freshness constraints. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2018, 37(11): 2743–2754
14. Ning Z, Hou W, Hu X, et al. A cloud-supported CPS approach to control decision of process manufacturing: 3D ONoC. In: Proceedings of the 13th IEEE Conference on Automation Science and Engineering. Xi'an: IEEE, 2017, 458–463
15. Liu J, Zhao F, Liu X, et al. Challenges towards elastic power management in internet data centers. In: Proceedings of the 29th IEEE International Conference on Distributed Computing Systems Workshops. Montreal: IEEE, 2009, 65–72
16. Paroliniy L, Toliaz N, Sinopoliy B, et al. A cyber-physical systems approach to energy management in data centers. In: Proceedings of the 1st ACM/IEEE International Conference on Cyber-physical Systems. Stockholm: ACM, 2010, 168–177
17. Ousterhout K, Wendell P, Zaharia M, et al. Sparrow: Distributed, low latency scheduling. In: Proceedings of the 24th ACM Symposium on Operating Systems Principle. Farmington: ACM, 2013, 69–84
18. Delimitrou C, Kozyrakis C. Quasar: Resource-efficient and QoS-aware cluster management. ACM SIGPLAN Notices, 2014, 49(4): 127–144
19. Delimitrou C, Kozyrakis C. Paragon: QoS-aware scheduling for heterogeneous datacenters. ACM SIGPLAN Notices, 2013, 48(4): 77–88
20. Higuera-Toledano M T, Risco-Martin J L, Arroba P, et al. Green adaptation of real-time web services for industrial CPS within a cloud environment. IEEE Transactions on Industrial Informatics, 2017, 13(3): 1249–1256
21. Yildirim G, Tatar, Y. Simplified agent-based resource sharing approach for WSN-WSN Interaction in IoT/CPS projects. IEEE ACCESS, 2018, 6: 78077–78091
22. Armbrust M, Fox A, Griffith R, et al. A view of cloud computing. Communications of the ACM, 2010, 53(4): 50–58
23. Xu Z W. Cloud-sea computing systems: Towards thousand-fold improvement in performance per watt for the coming zettabyte era. Journal of Computer Science and Technology, 2014, 29(2): 177–181

24. Vaquero L M, Rodero-Merino L. Finding your way in the fog: Towards a comprehensive definition of fog computing. Computer Communication Review, 2014, 44(5): 27–32
25. Singh S P, Nayyar A, Kumar R, et al. Fog computing: From architecture to edge computing and big data processing. Journal of Supercomputing, 2019, 75(4): 2070–2105
26. Pereira J, Ricardo L, Luís M, et al. Assessing the reliability of fog computing for smart mobility applications in VANETs. Future Generation Computer Systems, 2019, 94: 317–332
27. Gai K K, Qiu M K, Zhao H, et al. Dynamic energy-aware cloudlet-based mobile cloud computing model for green computing. Journal of Network and Computer Applications, 2016, 59(c): 46–54
28. Beck M T, Werner M, Feld S, et al. Mobile edge computing: A taxonomy. In: Proceedings of the 6th International Conference on Advances in Future Internet. Lisbon, 2014, 48–54
29. Zhang F, Ge J, Wong C, et al. Online learning offloading framework for heterogeneous mobile edge computing system. Journal of Parallel and Distributed Computing, 2019, 128: 167–183
30. Wang S G, Zhao Y L, Xu J L, et al. Edge server placement in mobile edge computing. Journal of Parallel and Distributed Computing, 2019, 127: 160–168
31. Do T V, Do N H, Nguyen H T, et al. Comparison of scheduling algorithms for multiple mobile computing edge clouds. Simulation Modelling Practice and Theory, 2019, 93: 104–118
32. Shi W S, Cao J, Zhang Q, et al. Edge computing: Vision and challenges. IEEE Internet of Things Journal, 2016, 3(5): 637–646
33. Uhlemann T H J, Lehmann C, Steinhilper R. The digital twin: Realizing the cyber-physical production system for Industry 4.0. Procedia CIRP, 2017, 61: 335–340
34. Yun S, Park J H, Kim W T. Data-centric middleware based digital twin platform for dependable cyber-physical systems. In: Proceedings of the 9th International Conference on Ubiquitous and Future Networks. Milan: IEEE, 2017, 922–926