Journal of Systems Engineering and Electronics
Vol. 35, No. 4, August 2024, pp.1013 – 1027

Cloud control for IIoT in a cloud-edge environment

YAN Ce[1], XIA Yuanqing[1,*], YANG Hongjiu[2], and ZHAN Yufeng[1]

1. School of Automation, Beijing Institute of Technology, Beijing 100081, China;
2. School of Electrical and Information Engineering, Tianjin University, Tianjin 300072, China

Abstract: The industrial Internet of Things (IIoT) is a new industrial idea that combines the latest information and communication technologies with the industrial economy. In this paper, a cloud control structure is designed for IIoT in cloud-edge environment with three modes of 5G. For 5G based IIoT, the time sensitive network (TSN) service is introduced in transmission network. A 5G logical TSN bridge is designed to transport TSN streams over 5G framework to achieve end-to-end configuration. For a transmission control protocol (TCP) model with nonlinear disturbance, time delay and uncertainties, a robust adaptive fuzzy sliding mode controller (AFSMC) is given with control rule parameters. IIoT workflows are made up of a series of subtasks that are linked by the dependencies between sensor datasets and task flows. IIoT workflow scheduling is a non-deterministic polynomial (NP)-hard problem in cloud-edge environment. An adaptive and non-local-convergent particle swarm optimization (ANCPSO) is designed with nonlinear inertia weight to avoid falling into local optimum, which can reduce the makespan and cost dramatically. Simulation and experiments demonstrate that ANCPSO has better performances than other classical algorithms.

Keywords: 5G and time sensitive network (TSN), industrial Internet of Things (IIoT) workflow, transmission control protocol (TCP) flows control, cloud edge collaboration, multi-objective optimal scheduling.

DOI: 10.23919/JSEE.2024.000074

1. Introduction

In recent years, the industrial Internet of Things (IIoT) is one of the most often discussed industrial concepts. As wireless communication and sensor network technologies advance, more networked and smart devices will be included in IIoT [1,2]. In [3], to plan and analyze IIoT data, a two-priority queuing approach was used. To dump peak loads over higher tiers of the cloud-edge hierarchy, a workload assignment algorithm is created. Then, for multipath transmission control protocol (TCP) over IIoT, extensive quality of service (QoS) modeling is established, and a QoS-aware personalized privacy protection model is provided to work in two layers: one at the cloud under the parameters of three typical 5G application modes, a 5G-based IIoT architecture is provided based on the architecture of 5G wireless communication technology [4, 5]. With limited communication resources, to create the groundwork for achieving the needed estimate accuracy, a transmission-estimation codesign framework was offered [6,7]. In [8], a concept on cloud control systems was defined with cloud computing and extended networked control systems. Some results and applications on cloud control systems were discussed in [9−11]. A security problem was studied and modeled by cloud control systems using a Stackelberg game. A security as a service scheme was given to defend against advanced persistent threats [12]. A networked multiagent predictive control system is analysed with cloud computing by which time delays were compensated actively under a cloud predictive control scheme [13,14]. In view of advantages from the cloud control technology, some applications were developed in practical engineering [15−17]. As a result, in order to effectively integrate IoT technology into industry, the cloud control technique must be evaluated in real-time restrictions, particularly for time-sensitive industrial processes.

The terminal layer, which comprises of smart IIoT devices, is the bottom layer of IIoT. The middle edge layer, which includes computation, storage, and gateways, receives service requests from these terminal devices. In addition, edge devices communicate with the cloud layer to offload latency-tolerant and computation-intensive operations. These large smart devices at the smart factory's edge demand data processing that is low-latency and location-aware. Edge computing was created to meet these stringent require-

Manuscript received March 01, 2022.
*Corresponding author.

ments [18,19]. To send hypertext transfer protocol (HTTP) requests and responses, modern cloud-edge data centers host a variety of HTTP services with TCP connections. End-host control was used to limit the expansion of the switch queue length in highly concurrent TCP connections, whereas probe packets are used to calm the aggressive rise of the congestion window in persistent TCP connections [20]. In [21], a solution was given with software defined networking (SDN) control and the IEEE time sensitive networking (TSN) standards to resolve latency-related issues in IIoT. TSN's effectiveness and efficiency are dependent on a dependable real-time communications scheduling mechanism. In TSN, a formal framework was provided for investigating scheduling options in [22]. There exist two challenging problems: synchronization and scheduling in TSN "IEEE 802.11" wireless networks [23,24], a novel method was proposed by using commercial low-cost components possible to make high-precision wireless synchronization.

Active task management systems were described as a feedback control system and deployed in the gateways of communication networks to support TCP flows. Besides, stochastic differential equations (SDE) were used to explain the interactions between a collection of TCP flows and active queue management (AQM) [25]. In [26], from the perspective of control engineering, a combined TCP and AQM model was examined. A previously built nonlinear TCP model analyzed via linearization. For a delay differential equations model of TCPs, alternative control techniques for the congestion-avoidance mode were given. Then a robust controller was designed with control of systems with time delays [27].

With the develpoment of IIoT and edge computing, workflow scheduling was designed in distributed cloud-edge environment, which is a non-deterministic polynomial (NP)-hard problem. A new data replica placement strategy for data-IIoT activities was offered in the context of collaborative edge and cloud computing [28]. In [29], the goal of an edge cloud-based multi-robot system was to overcome the limitations of a remote cloud-based system when it comes to transmitting time-sensitive data. The scheduling of robotic workflows was modeled as a constrained multi-objective optimization issue that can be solved using a multi-objective evolutionary technique. For IoT-enabled cloud-edge computing, a computational offloading mechanism was proposed. Furthermore, the execution time and energy consumption of mobile devices were evaluated using a system model [30]. In [31], using a directed search procedure, and combining inertia weight, a unique directional particle swarm optimization was presented, which may drastically reduce the makespan and cost while achieving a compromise outcome. The hybrid IIoT workflows scheduling difficulties in a hybrid cloud-edge context are the focus of this research.

The main contributions of this paper are listed as follows:

(i) A cloud control structure is designed for IIoT in cloud-edge environment with 5G and TSN, and a workflow model is given for big data task and time-sensitive tasks in IIoT.

(ii) A 5G logical TSN bridge is designed to transport TSN streams over 5G framework in our cloud control structure to satisfy time-sensitive conditions.

(iii) An adaptive fuzzy sliding mode is given to deal with nonlinear disturbance, time delay and uncertainties in a TCP model.

(iv) For hybrid workflow scheduling problem, an adaptive and non-local-convergent particle swarm optimization is designed to reduce the makespan and cost of IIoT workflows.

2. System model

2.1 Cloud control for IIoT

The IIoT is a new infrastructure, application mode, and industrial ecology that brings together the current generation of information and communication technologies with the industrial economy. It creates a new manufacturing and service system that spans the entire industrial chain as well as the entire value chain by connecting people, machines, things, and systems, so as to provide services for industry and even industry digitization, networking intelligent development provides a way to realize it. There are some industrial applications such as big data analysis driven smart manufacturing and digital twin.

In this subsection, a cloud control architecture is designed for IIoT in cloud-edge environment with three communication modes: massive machine type communication (mMTC), ultra-reliable and low latency communications (URLLC) and enhanced mobile broadband (eMBB) of 5G in Fig. 1. Real-time data collection, networked collaborative manufacturing and digital twin are implemented in our structure under the circumstance of 5G wireless.

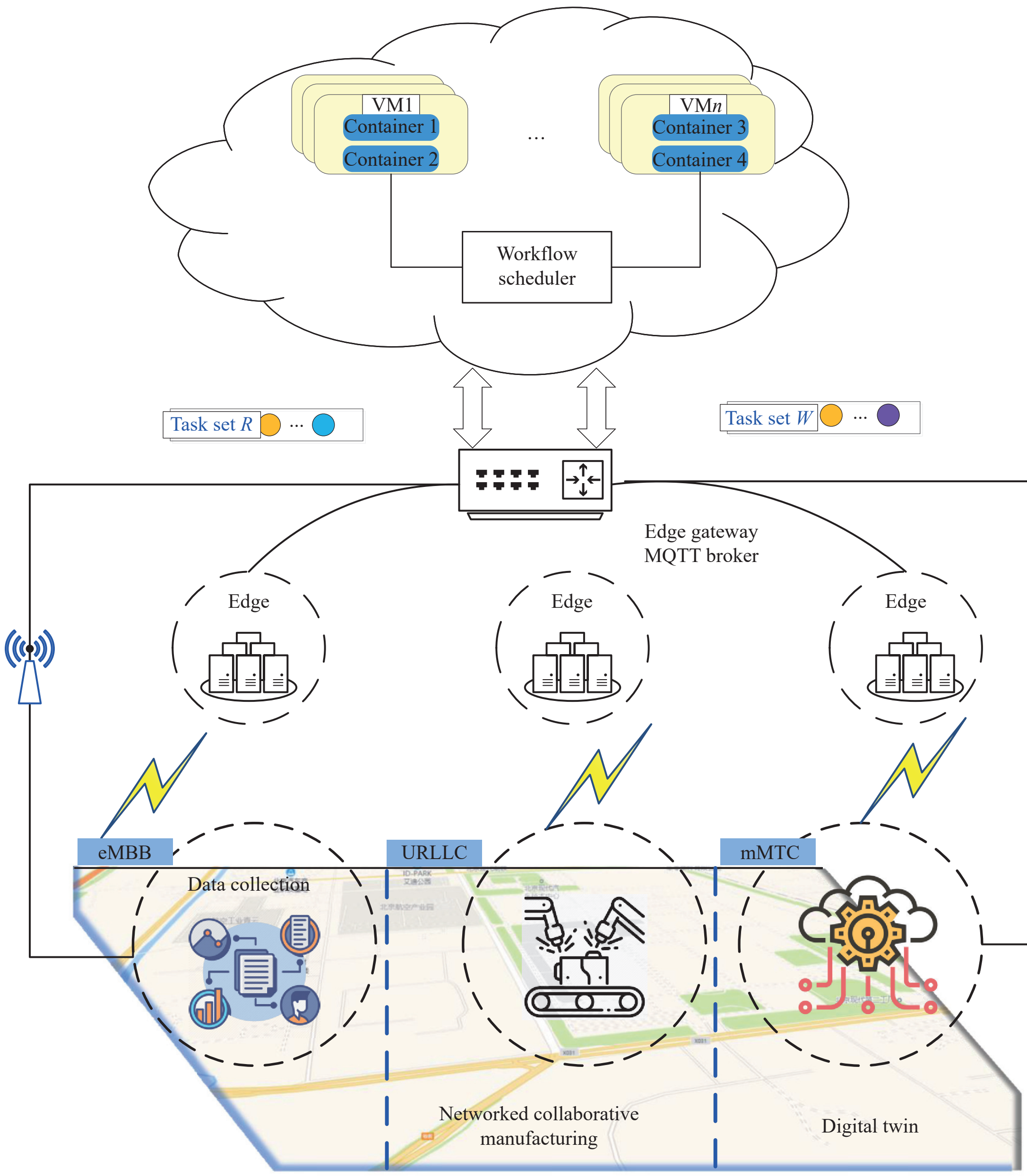


Fig. 1 Cloud control structure for IIoT

The combination of edge and cloud computing paradigms enables the deployment of IIoT operations in real time and at a cheap cost to cooperative user groups. The cloud-edge infrastructure between industrial devices and the cloud is described in this study as a three-tier computational framework that allows industrial applications for smart factories to be implemented with decreased communication latency. Edge devices use on-board sensors to capture ambient data for service execution. The message queuing telemetry transport (MQTT) gateway uses the TCP protocol to communicate with the edge and cloud. Furthermore, when performing any industrial function, it forms a combined resource pool with both local and remote resources, as shown in Fig. 1.

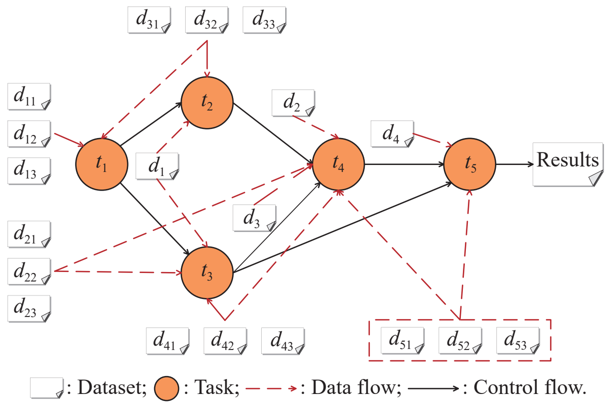
TCP protocol is used to send data in the network between the cloud and the edge with the goal of reducing queue utilization and delay. It is basically a problem of feedback control. For the challenge of congestion control in TCP complicated systems with unknown nonlinear disturbances, an active queue management strategy based on adaptive fuzzy sliding mode control is investigated.

Industrial services workflows are made up of data-hungry, time-sensitive, and compute-intensive processes. As a result, decreasing the makespan and monetary cost of scheduling IIoT workflows is a concern for profit-driven infrastructure as a service (IaaS) cloud-edge providers. In this paper, an efficient workflow scheduling technique for cloud-edge is provided, which takes into account both the makespan and the monetary cost.

2.2 IIoT workflow model

IIoT workflows are made up of a series of subtasks that are linked by the dependencies between sensor datasets and task flows. Fig. 2 shows an IIoT workflow with five jobs, five datasets, and five created intermediate datasets, which are executed in the sequence specified by the control flow.

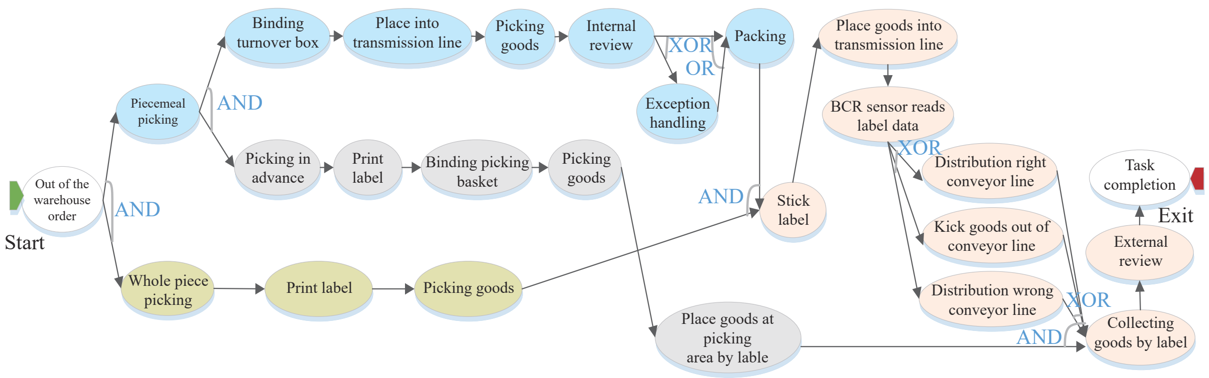
A collection of cooperative users from multiple groups submit a workflow project to be completed in collaborative edge and cloud environments with infinite capacity resources, which uses massive quantities of data collected by IIoT sensors as input and produces output datasets. Finally, the collaborative user group receives the intermediate output and final datasets. If the data is kept locally and has a low latency access cost, an edge cluster to Internet of Things (IoT) devices can respond in real time when a user makes a request.



d_{31} d_{32} d_{33}
d_{11} d_{12} d_{13}
t_2
d_2
d_4
t_1
d_1
t_4
t_5
Results
d_3
d_{21} d_{22} d_{23}
t_3
d_{41} d_{42} d_{43}
d_{51} d_{52} d_{53}
: Dataset; : Task; : Data flow; : Control flow.

Fig. 2 An example of IIoT bigdata workflows

Fig. 3 shows how the delay-ensitive workflow is utilized to develop an out-of-warehouse method for IoT. The two essential parallel executions that make up the whole outbound procedure are whole piece picking.



Start
Out of the warehouse order
AND
Piecemeal picking
AND
Binding turnover box
Place into transmission line
Picking goods
Internal review
XOR
OR
Exception handling
Packing
Picking in advance
Print label
Binding picking basket
Picking goods
Whole piece picking
Print label
Picking goods
AND
Stick label
Place goods into transmission line
BCR sensor reads label data
XOR
Distribution right conveyor line
Kick goods out of conveyor line
Distribution wrong conveyor line
XOR
AND
Place goods at picking area by lable
Collecting goods by label
External review
Task completion
Exit

Fig. 3 Out-of-warehouse control flow of logistics

The picking task in advance as well as the regular picking task are included in the piecemeal picking. At the stick label task, the piecemeal and whole piece picking are combined, and the goods are then placed into the transmission line. The label data is read by a sensor, and the control system distributes the correct conveyor slide line. The sticklabel task then joins the picking in advance process and the other synchronous parallel processes at the gathering goods. The entire out-of-warehouse process is completed after the external review work is completed.

A directed acyclic graph (DAG) is a common way to define a workflow. Formally, a workflow can be formalized as $G=(\varGamma,\mathrm{Data},E)$ with $\varGamma=\{t_1,t_2,\cdots,t_m\}$ denoting the set of m tasks in the workflow and $E=\{(t_i,t_j)|t_i,t_j\in\varGamma\}$ denoting the set of task dependencies. The process G has a deadline constraint D, which states that all tasks in the workflow must be completed before the deadline.

To describe the qualities of a task $t_i(1\leqslant i\leqslant m)$, we utilize a quadruple, i.e., $t_i:\{\mathrm{len}(t_i),\mathrm{trans}(t_i,t_j),\mathrm{pred}(t_i),\mathrm{succ}(t_i)\}$, where $\mathrm{len}(t_i)$ is the execution length of task t_i, the size of data transmitted from the immediate predecessor node t_i to the immediate successor node t_j is $\mathrm{trans}(t_i,t_j)$. $\mathrm{pred}(t_i)$ and $\mathrm{succ}(t_i)$ are the set of immediate predecessor and successor nodes of task t_i and are expressed as $\mathrm{pred}(t_i)=\{t_j\,|\,(t_j,t_i)\in E\}$ and $\mathrm{succ}(t_i)=\{t_j\,|\,(t_j,t_i)\in E\}$, respectively.

3. Heterogeneous network in cloud-edge environment

3.1 Data flow benchmarking cases

As shown in Fig. 4, three benchmarking cases illustrate the measurement of the full stack round trip time (RTT) in three models: device-edge, device-cloud, and device-edge-cloud.

It measures the time between the transmission of an application layer message and the arrival of the corresponding response message to the application layer, and reflects the total latency throughout an edge-cloud architecture’s entire stack. The total stack RTT measure is used to benchmark a variety of factors that can affect system performance, including message sending interval and the number of concurrent devices. These variables are modified to determine the performance limit of various models.

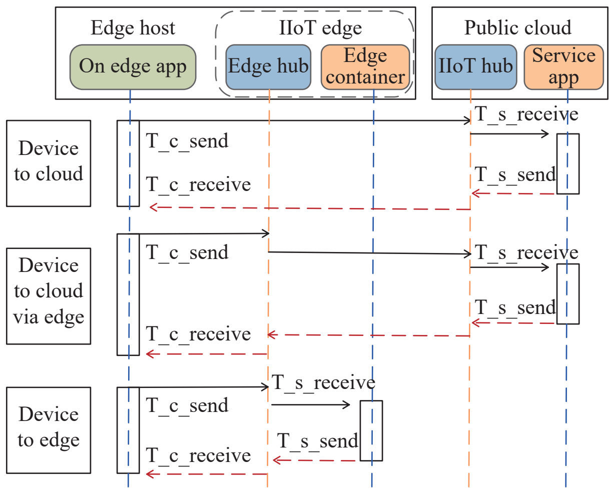


Fig. 4 Data flow in cloud-edge

3.2 TSN service in 5G IIoT

The IEEE 802.1 working group is actively developing TSN, which is a set of Ethernet standards. It enables real-time applications to communicate in a predictable manner across Ethernet. TSN guarantees latency bounds by utilizing synchronization and planned traffic. For industrial applications, TSN is the most promising vendor-agnostic time-deterministic communication solution. The most viable approach for achieving the performance requirements of industrial applications has been found as an adapted integration of TSN into 5G. As shown in Fig. 5, the fundamental concept is to create a black box model in which the 5G system looks to be a TSN bridge to the outer TSN network while transmitting TSN frames through its own framework.

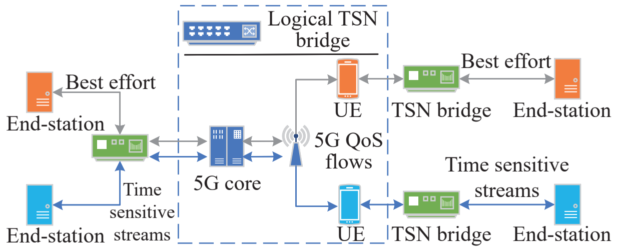


Fig. 5 5G logical TSN bridge transports TSN streams to achieve end-to-end configuration

The 5G system can carry and deliver TSN traffic because Ethernet traffic is supported. 5G employs its internal QoS architecture, which is built on QoS flows, to classify traffic classes as in TSN. If all network instances share a common sense of time, a scheduled TSN stream can be created by reserving the required time slots along its transmission path. The 3rd generation partnership project (3GPP) is actively debating how to send timing data via 5G in order to enable seamless integration with the IEEE 802.1AS Synchronization Standard. The synchronization control device (SCD) controls the sequence of actions and analyzes input parameters during synchronization.

As shown in Fig. 6, an architecture of the network transmission is designed for IIoT application from [32]. The agent in the TSN switch is identical to the cloud and edge agents. The manager sends the agent the orchestrator-generated physical topology and data flows, and the agent then calls the scheduler to construct the TSN switch's particular network configuration in centralized user configuration (CUC).

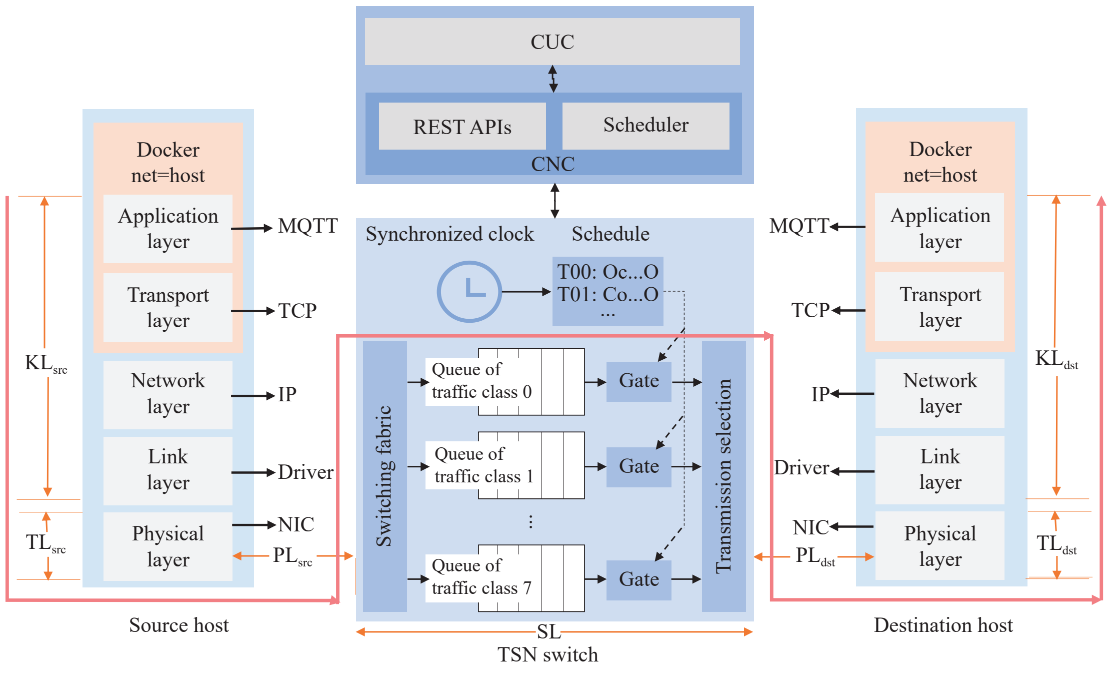


Fig. 6 Breakdown architecture of network transmission

Control data travels from the source host to the switch, then to the destination host. The total latency L is obtained as

$$L = \mathrm{KL}_{\mathrm{src}} + \mathrm{TL}_{\mathrm{src}} + \mathrm{PL}_{\mathrm{src}} + \mathrm{SL} + \mathrm{PL}_{\mathrm{dst}} + \mathrm{TL}_{\mathrm{dst}} + \mathrm{KL}_{\mathrm{dst}}$$

where KL is the protocol processing latency, TL is the transmission latency, PL is the propagation latency and SL is the switch latency.

An open source MQTT broker has been developed for IIoT. It supports large-scale connections, offline transmission, low bandwidth, dependable message delivery, and low energy usage. Industrial data is transferred under the control of TCP in the transport layer. This protocol is often perceived as a mature and rigid solution.

3.3 Adaptive fuzzy control for TCP connection

In [26], a nonlinear dynamic model of a TCP connection through a crowded AQM router is developed based on fluid-flow theory and stochastic differential equation analysis. The simpler version, which does not take into account the TCP timeout mechanism, is as follows:

$$\begin{cases} \dot{W}(t) = \dfrac{1}{\dfrac{q(t)}{C} + T_p} - \dfrac{W(t)W(t-R(t))}{2\left(\dfrac{q(t)}{C} + T_p\right)} p(t-R(t)) \\ \dot{q}(t) = \dfrac{N(t)}{\dfrac{q(t)}{C} + T_p} W(t) - C(t) \end{cases} \tag{1}$$

where $W(t)$ is the average TCP window size, $q(t)$ is the average queue length, C is the capacity of link, T_p is the propagation delay, $N(t)$ is the load factor on number of TCP sessions, $p(t)$ is the probability of packet mark/drop, the delay of transfer is defined as $R(t) = q(t)/C + T_p$, which includes queuing and transmission.

Fig. 7 depicts the characteristics of congestion control based on window and the dynamic queue in the aforementioned differential equation in (1). The coupled, nonlinear differential equation in (1) describes this model, which illustrates TCP window-control and queue dynamics in Fig. 7. To acquire insight for the purposes of feedback control, these dynamics are approximated by small-signal linearization around an operating point. It is worth noting that the link between Fig. 7 and the differential equation in (1) is well described in [26].

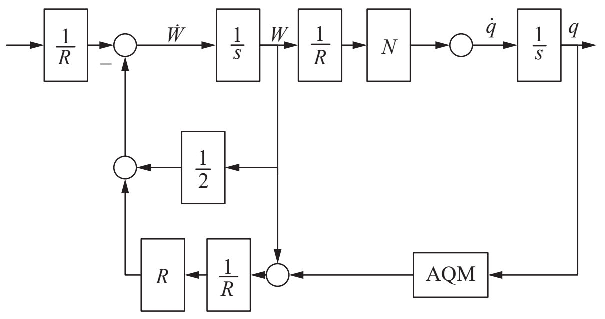


Fig. 7 TCP congestion-avoidance mode with AQM

TCP window control dynamics are described by the first differential equation in (1). In response to packet marking $p(t)$, $1/R(t)$ on the right-hand side roughly models the window's additive rise, whereas $W(t)$ on the left-hand side roughly models the window's multiplicative drop. The bottleneck queue length is essentially an accumulated difference between packet arrival rate $(N(t)W(t))/R(t)$ and link capacity C, according to the second equation in (1).

Define an equilibrium point W_0, q_0, p_0 with $\dot{W}(t) = 0$ and $\dot{q}(t) = 0$, then one has that $R(t) = R_0$, $N(t) = N$ and $C(t) = C$. Moreover, there exist $W_0^2 = 2$, $W_0 = (R_0 C)/N$ and $R_0 = q_0/C + T_p$. To linearize (1), let $\delta W(t) = W(t) - W_0$, $\delta q(t) = q(t) - q_0$ and $\delta p(t) = p(t) - p_0$. A linearized model is shown as

$$\begin{cases} \delta\dot{W}(t) = -\dfrac{N}{R_0^2 C}(\delta W(t) + \delta W(t-R_0)) - \\ \qquad \dfrac{1}{R_0^2 C}(\delta q(t) - \delta q(t-R_0)) - \dfrac{R_0^2 C}{2N^2}\delta p(t) \\ \delta\dot{q}(t) = \dfrac{N}{R_0}\delta W(t) - \dfrac{1}{R_0}\delta q(t) \end{cases} \tag{2}$$

Considering $\boldsymbol{x}(t) = \begin{bmatrix} \delta W(t), & \delta q(t) \end{bmatrix}^{\mathrm{T}}$, $u(t) = \delta p(t)$ and $\tau = R_0$, the system in (2) is written as

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{A}\boldsymbol{x}(t) + \boldsymbol{A}_d \boldsymbol{x}(t-\tau) + \boldsymbol{B}u(t) \tag{3}$$

where

$$\boldsymbol{A} = \begin{bmatrix} -\dfrac{N}{R_0^2 C} & -\dfrac{1}{R_0^2 C} \\ \dfrac{N}{R_0} & -\dfrac{1}{R_0} \end{bmatrix}, \ \boldsymbol{A}_d = \begin{bmatrix} -\dfrac{N}{R_0^2 C} & \dfrac{1}{R_0^2 C} \\ 0 & 0 \end{bmatrix},$$

and

$$\boldsymbol{B} = \begin{bmatrix} -\dfrac{R_0^2 C}{2N^2} \\ 0 \end{bmatrix} = \begin{bmatrix} \bar{B} \\ 0 \end{bmatrix}.$$

Note that the timeout and slow start mechanism are ignored in (3), so a TCP model with uncertain, time-delay and nonlinearity is considered as

$$\dot{\boldsymbol{x}}(t) = (\boldsymbol{A} + \Delta\boldsymbol{A}(t))\boldsymbol{x}(t) + (\boldsymbol{A}_d + \Delta\boldsymbol{A}_d(t))\boldsymbol{x}(t-\tau) + \\ \boldsymbol{B}(u(t) + \boldsymbol{G}(\boldsymbol{x}(t), \boldsymbol{x}(t-\tau), \boldsymbol{\sigma}(t))) \tag{4}$$

where $\boldsymbol{G}(\boldsymbol{x}(t), \boldsymbol{x}(t-\tau), \boldsymbol{\sigma}(t))$ is a nonlinear function, $\sigma(t)$ is a time-varying disturbance parameter. Then there exists that

$$\boldsymbol{G}(\boldsymbol{x}(t), \boldsymbol{x}(t-\tau), \boldsymbol{\sigma}(t)) = \\ \boldsymbol{G}_1(x(t)) + \boldsymbol{G}_2(x(t-\tau)) + \Delta\boldsymbol{G}(\boldsymbol{x}(t), \boldsymbol{x}(t-\tau), \boldsymbol{\sigma}(t))$$

where

$$|\Delta \boldsymbol{G}(\boldsymbol{x}(t),\boldsymbol{x}(t-\tau),\boldsymbol{\sigma}(t))| \leqslant \alpha_0+\alpha_1\|\boldsymbol{x}(t)\|+\alpha_2\|\boldsymbol{x}(t-\tau)\|.$$

Furthermore, the system in (4) is written as

$$\begin{gathered}\dot{\boldsymbol{x}}_1(t)=(\boldsymbol{A}_{11}+\Delta\boldsymbol{A}_{11})\boldsymbol{x}_1(t)+(\boldsymbol{A}_{d11}+\Delta\boldsymbol{A}_{d11})\boldsymbol{x}_1(t-\tau)+\\(\boldsymbol{A}_{12}+\Delta\boldsymbol{A}_{12})\boldsymbol{x}_2(t)+(\boldsymbol{A}_{d12}+\Delta\boldsymbol{A}_{d12})\boldsymbol{x}_2(t-\tau)+\\\bar{\boldsymbol{B}}(u(t)+\boldsymbol{G}(\boldsymbol{x}(t),\boldsymbol{x}(t-\tau),\boldsymbol{\sigma}(t))),\\\dot{\boldsymbol{x}}_2(t)=(\boldsymbol{A}_{21}+\Delta\boldsymbol{A}_{21})\boldsymbol{x}_1(t)+(\boldsymbol{A}_{d21}+\Delta\boldsymbol{A}_{d21})\boldsymbol{x}_2(t-\tau)+\\(\boldsymbol{A}_{22}+\Delta\boldsymbol{A}_{22})\boldsymbol{x}_2(t)+(\boldsymbol{A}_{d22}+\Delta\boldsymbol{A}_{d22})\boldsymbol{x}_2(t-\tau).\end{gathered}$$

Without loss of generality, the following sliding surface is designed as

$$\begin{gathered}\boldsymbol{S}=\hat{\boldsymbol{K}}\boldsymbol{x}(t)=\begin{bmatrix}-\boldsymbol{K} & \boldsymbol{I}\end{bmatrix}\begin{bmatrix}\boldsymbol{x}_1(t)\\\boldsymbol{x}_2(t)\end{bmatrix}=\\-\boldsymbol{K}\boldsymbol{x}_1(t)+\boldsymbol{x}_2(t)=\boldsymbol{0}.\end{gathered}\tag{5}$$

In this paper, for every $t>0$, a switch input $u(t)$ is meant to keep the system states on the sliding surface $\boldsymbol{S}(t)$. The reaching condition for the unknown nonlinear disturbance is satisfied by an adaptive fuzzy control law. The output of a fuzzy system with fuzzy IF-THEN rules is

$$y(x)=\frac{\sum_{j=1}^{m}y^j\left[\prod_{i=1}^{n}\mu_{A_i^j}(x_i)\right]}{\sum_{j=1}^{m}\left[\prod_{i=1}^{n}\mu_{A_i^j}(x_i)\right]}.$$

Introduce the fuzzy basis function vector $\boldsymbol{\xi}(x)$, it is obtained that $y(x)=\boldsymbol{\theta}^{\mathrm{T}}\boldsymbol{\xi}(x)$, where

$$\begin{gathered}\boldsymbol{\theta}=\begin{bmatrix}\theta_1, & \theta_2, & \cdots, & \theta_m\end{bmatrix}^{\mathrm{T}},\\\boldsymbol{\xi}(x)=\begin{bmatrix}\xi_1(x), & \xi_2(x), & \cdots, & \xi_j(x)\end{bmatrix}^{\mathrm{T}},\end{gathered}$$

and

$$\xi_j(x)=\frac{\prod_{i=1}^{n}\mu_{A_i^j}(x_i)}{\sum_{j=1}^{m}\left[\prod_{i=1}^{n}\mu_{A_i^j}(x_i)\right]}.$$

Then $\hat{\boldsymbol{G}}_1(\boldsymbol{x}(t),\boldsymbol{\theta}_1)$ and $\hat{\boldsymbol{G}}_2(\boldsymbol{x}(t-\tau),\boldsymbol{\theta}_2)$ are used to approximate the uncertain terms, $\boldsymbol{\theta}_1$ and $\boldsymbol{\theta}_2$ are tunable parameters. By fuzzy rule, one has that

$$\hat{\boldsymbol{G}}_1(\boldsymbol{x}(t),\boldsymbol{\theta}_1)=\boldsymbol{\theta}_1^{\mathrm{T}}\boldsymbol{\xi}_1(\boldsymbol{x}(t))$$

and

$$\hat{\boldsymbol{G}}_2(\boldsymbol{x}(t-\tau),\boldsymbol{\theta}_2)=\boldsymbol{\theta}_2^{\mathrm{T}}\boldsymbol{\xi}_2(\boldsymbol{x}(t-\tau)).$$

There exist optimal approximation parameters $\boldsymbol{\theta}_1^*$ and $\boldsymbol{\theta}_2^*$ such that $\boldsymbol{\xi}_1(\boldsymbol{x}(t))^{\mathrm{T}}\boldsymbol{\theta}_1^*$ and $\boldsymbol{\xi}_2(\boldsymbol{x}(t-\tau))^{\mathrm{T}}\boldsymbol{\theta}_2^*$ approximate to $\bar{\boldsymbol{G}}_1(\boldsymbol{x}(t)$ and $\bar{\boldsymbol{G}}_2(\boldsymbol{x}(t-\tau)$ at any desired degree. Moreover, the optimal parameters are given as follows:

$$\begin{gathered}\boldsymbol{\theta}_1^*=\arg\min_{\boldsymbol{\theta}_1\in\boldsymbol{G}1}\left[\sup_{x(t)\in\boldsymbol{\Omega}_x}|\hat{\boldsymbol{G}}_1(\boldsymbol{x}(t),\boldsymbol{\theta}_1)-\bar{\boldsymbol{G}}_1(\boldsymbol{x}(t)|\right],\\\boldsymbol{\theta}_2^*=\arg\min_{\boldsymbol{\theta}_2\in\boldsymbol{G}2}\left[\sup_{x(t-\tau)\in\boldsymbol{\Omega}_x}|\hat{\boldsymbol{G}}_2(\boldsymbol{x}(t-\tau),\boldsymbol{\theta}_2)-\bar{\boldsymbol{G}}_2(\boldsymbol{x}(t-\tau)|\right],\end{gathered}$$

where $\hat{\boldsymbol{G}}_1$, $\hat{\boldsymbol{G}}_2$, and $\boldsymbol{\Omega}_x$ are sets of bounds on $\boldsymbol{\theta}_1$, $\boldsymbol{\theta}_2$, and $\boldsymbol{x}(t)$.

Theorem 1 For the system in (4), an adaptive fuzzy sliding mode control law is designed as

$$\boldsymbol{u}(t)=\boldsymbol{u}_1(t)+\boldsymbol{u}_2(t)+\boldsymbol{u}_3(t)\tag{6}$$

where

$$\begin{gathered}\boldsymbol{u}_1(t)=-(\hat{\boldsymbol{K}}\boldsymbol{B})^{-1}\left(\boldsymbol{\theta}_1^{\mathrm{T}}\boldsymbol{\xi}_1(\boldsymbol{x}(t))+\boldsymbol{\theta}_2^{\mathrm{T}}\boldsymbol{\xi}_2(\boldsymbol{x}(t-\tau))\right),\\\boldsymbol{u}_2(t)=-(\hat{\boldsymbol{K}}\boldsymbol{B})^{-1}(\eta_0+\eta_1\|\boldsymbol{x}(t)\|+\eta_2\|\boldsymbol{x}(t-\tau)\|)\operatorname{sgn}(\boldsymbol{S}),\\\boldsymbol{u}_3(t)=-(\hat{\boldsymbol{K}}\boldsymbol{B})^{-1}\beta\boldsymbol{S},\end{gathered}$$

where $\boldsymbol{\theta}_1$, $\boldsymbol{\theta}_2$, η_0, η_1 and η_2 satisfy the following adaptive laws: $\dot{\boldsymbol{\theta}}_1=r_1\boldsymbol{S}(t)\boldsymbol{\xi}_1(\boldsymbol{x}(t))$, $\dot{\boldsymbol{\theta}}_2=r_2\boldsymbol{S}(t)\boldsymbol{\xi}_2(\boldsymbol{x}(t-\tau))$, $\dot{\eta}_0=r_3|\boldsymbol{S}(t)|$, $\dot{\eta}_1=r_4|\boldsymbol{S}(t)|\|\boldsymbol{x}(t)\|$ and $\dot{\eta}_2=r_5|\boldsymbol{S}(t)|\|\boldsymbol{x}(t-\tau)\|$. With two fuzzy logic functions, $\boldsymbol{u}_1(t)$ approximates unknown nonlinear functions, the adaptive controller $\boldsymbol{u}_2(t)$ compensates time-varying uncertainties, and the sliding mode $\boldsymbol{S}(t)$ converges to zero with $\boldsymbol{u}_3(t)$ asymptotically. Finally, the sliding surface can be reached in a limited amount of time.

Proof Define a Lyapunov function for system in (4) as follows:

$$\boldsymbol{V}=\frac{1}{2}\left[\boldsymbol{S}^2+\frac{1}{r_1}\bar{\boldsymbol{\theta}}_1^{\mathrm{T}}\bar{\boldsymbol{\theta}}_1+\frac{1}{r_2}\bar{\boldsymbol{\theta}}_2^{\mathrm{T}}\bar{\boldsymbol{\theta}}_2+\frac{1}{r_3}\bar{\eta}_0^2+\frac{1}{r_4}\bar{\eta}_1^2+\frac{1}{r_5}\bar{\eta}_2^2\right]$$

where $\bar{\boldsymbol{\theta}}_1=\boldsymbol{\theta}_1^*-\boldsymbol{\theta}_1$, $\bar{\boldsymbol{\theta}}_2=\boldsymbol{\theta}_2^*-\boldsymbol{\theta}_2$, $\bar{\eta}_0=\eta_0^*-\eta_0$, $\bar{\eta}_1=\eta_1^*-\eta_1$, $\bar{\eta}_2=\eta_2^*-\eta_2$. The derivative along the state trajectory of the Lyapunov function is

$$\boldsymbol{V}=\boldsymbol{S}\dot{\boldsymbol{S}}+\frac{1}{r_1}\bar{\boldsymbol{\theta}}_1^{\mathrm{T}}\dot{\bar{\boldsymbol{\theta}}}_1+\frac{1}{r_2}\bar{\boldsymbol{\theta}}_2^{\mathrm{T}}\dot{\bar{\boldsymbol{\theta}}}_2+\frac{1}{r_3}\bar{\eta}_0\dot{\bar{\eta}}_0+\frac{1}{r_4}\bar{\eta}_1\dot{\bar{\eta}}_1+\frac{1}{r_5}\bar{\eta}_2\dot{\bar{\eta}}_2.$$

Then it is known that

$$\begin{gathered}\boldsymbol{S}\dot{\boldsymbol{S}}=\hat{\boldsymbol{K}}\boldsymbol{x}(t)\hat{\boldsymbol{K}}\dot{\boldsymbol{x}}(t)=\\\hat{\boldsymbol{K}}\boldsymbol{x}(t)\hat{\boldsymbol{K}}[(\boldsymbol{A}+\Delta\boldsymbol{A}(t))\boldsymbol{x}(t)+(\boldsymbol{A}_d+\Delta\boldsymbol{A}_d(t))\boldsymbol{x}(t-\tau)+\\\boldsymbol{B}(u(t)+\boldsymbol{G}(x(t),\boldsymbol{x}(t-\tau),\boldsymbol{\sigma}(t)))].\end{gathered}$$

Moreover, let

$$\begin{gathered}\bar{\boldsymbol{G}}_1(\boldsymbol{x}(t))=\hat{\boldsymbol{K}}\boldsymbol{A}\boldsymbol{x}(t)+\boldsymbol{G}_1(\boldsymbol{x}(t)),\\\bar{\boldsymbol{G}}_2(\boldsymbol{x}(t-\tau))=\hat{\boldsymbol{K}}\boldsymbol{A}_d\boldsymbol{x}(t-\tau)+\boldsymbol{G}_2(\boldsymbol{x}(t-\tau)).\end{gathered}$$

Then an inequality is shown as

$$\begin{gathered}\boldsymbol{S}\dot{\boldsymbol{S}} \leqslant |\hat{\boldsymbol{K}}\boldsymbol{x}(t)| \times \\ [(\eta_0^* - \eta_0) + (\eta_1^* - \eta_1)\|\boldsymbol{x}(t)\| + (\eta_2^* - \eta_2)\|\boldsymbol{x}(t-\tau)\|] + \\ \left[(\boldsymbol{\theta}_1^* - \boldsymbol{\theta}_1)^{\mathrm{T}}\boldsymbol{\xi}_1(x(t)) + (\boldsymbol{\theta}_2^* - \boldsymbol{\theta}_2)^{\mathrm{T}}\boldsymbol{\xi}_2(\boldsymbol{x}(t-\tau))\right]\hat{\boldsymbol{K}}\boldsymbol{x}(t) - \beta\boldsymbol{S}^2.\end{gathered}$$

Therefore, one has that $\dot{V} \leqslant -\beta\boldsymbol{S}^2$. And the stability condition is satisfied. □

4. Hybrid cloud workflows scheduling in cloud-edge environment

The cost and execution time of cloud-edge servers in a cloud-edge environment are considered in this research as part of a hybrid cloud workflows scheduling model.

4.1 Resource model

A cloud control framework for IIoT with cloud-edge computing is illustrated in Fig. 1. The requirements of the cost and execution time for the industrial devices is satisfied with a cloud-edge computing paradigm in IIoT. In the cloud control framework, M devices are covered by an edge cloud, which is connected with the public cloud in remote area. The IIoT application is defined as a DAG workflow.

In the cloud-edge environment, computing tasks will be executed by the terminal devices, the edge or the cloud servers by computation offloading for IIoT workflows. X_m denotes hybrid scheduling deployment strategies of the workflow G_m. The scheduling strategy of IIoT task t_i is defined as element $x_{m,i}$, which is shown as

$$x_{m,i} = \begin{cases} 1, & t_i \text{ is ecxecuted in the edge} \\ 2, & t_i \text{ is ecxecuted in the cloud} \end{cases}.$$

4.2 Makespan model

For an IIoT workflow, the characteristics of the real-time response and the minized total completion time should be satisfied. Thus, the maximum execution time of an IIoT workflow is makespan. In our paper, docker containers deployed in the cloud-edge environment, which are divided into cloud and edge cloud containers.

The execution time of G_m includes three parts, i.e., the computing time T^e, the offloading latency T^L, and the transmission time T^t.

For the task t_i, the offloading latency $T^L(t_i)$ is given as

$$T^L(t_i) = \begin{cases} T_{\mathrm{LAN}}, & x_{m,i} = 1 \\ T_{\mathrm{WAN}}, & x_{m,i} = 2 \end{cases}$$

where T_{LAN} and T_{WAN} are the latency of local area network (LAN) and wide area network (WAN), respectively.

Considering the execution process of a task, the computing time is calculated through the workload of the industrial task and the executive ability of the docker container. Then the computing time $T^e(t_i)$ is calculated as follows:

$$T^e(t_i) = \begin{cases} \dfrac{t_i}{f_{\mathrm{edge}}}, & x_{m,i} = 1 \\ \dfrac{t_i}{f_{\mathrm{cloud}}}, & x_{m,i} = 2 \end{cases}$$

where f_{edge} and f_{cloud} denote the computing power of the edge and the cloud, respectively.

The transmission time between tasks t_i and t_j, $T^t(t_i, t_j)$ is given as

$$T^t(t_i, t_j) = \begin{cases} \dfrac{d(t_i, t_j)}{B_{\mathrm{edge}}}, & x_{m,i} = 1;\ x_{m,i} = 1 \\ \dfrac{d(t_i, t_j)}{B_{\mathrm{ec}}}, & x_{m,i} = 1;\ x_{m,i} = 2 \\ \dfrac{d(t_i, t_j)}{B_{\mathrm{ec}}}, & x_{m,i} = 2;\ x_{m,i} = 1 \\ \dfrac{d(t_i, t_j)}{B_{\mathrm{cloud}}}, & x_{m,i} = 2;\ x_{m,i} = 2 \end{cases}$$

where $d(t_i, t_j)$ is the transmission data between tasks t_i and t_j, and B_{ec} is the network bandwidth between cloud and edge.

Cloud computing at the remote end and edge computing at the close end are both part of the hybrid cloud datacenters environment $\mathrm{DC} = \{\mathrm{DC}_{\mathrm{cld}}, \mathrm{DC}_{\mathrm{edg}}\}$, which consists of numerous datacenters. Cloud computing is a type of computing that uses $\mathrm{DC}_{\mathrm{cld}} = \mathrm{dc}_1, \mathrm{dc}_2, \cdots, \mathrm{dc}_n$ which is made up of n datacenters, as well as edge computing. The datacenter $\mathrm{DC}_{\mathrm{edg}} = \mathrm{dc}_1, \mathrm{dc}_2, \cdots, \mathrm{dc}_m$ is made up of m datacenters. The bandwidth between several datacenters is measured in megabits per second as

$$\textbf{Bandwidth} = \begin{bmatrix} b_{11} & b_{12} & \cdots & b_{1|\mathrm{DC}|} \\ b_{21} & b_{22} & \cdots & b_{2|\mathrm{DC}|} \\ \vdots & \vdots & \ddots & \vdots \\ b_{|\mathrm{DC}|1} & b_{|\mathrm{DC}|2} & \cdots & b_{|\mathrm{DC}||\mathrm{DC}|} \end{bmatrix},$$

$$b_{i,j} = <\mathrm{band}_{ij}, \mathrm{type}_i, \mathrm{type}_j>,$$

where b_{ij} is the bandwidth between datacenters dc_i and dc_j. band_{ij} is the measured value of bandwidth b_{ij}. Let $T(G_m)$ be the execution time of the mth industrial workflow, which is obtained as

$$T(G_m) = T^L(G_m) + T^t(G_m) + T^e(G_m).$$

4.3 Cost model

Users should be aware that the public cloud's docker container resources are subject to fees. A public docker container is mostly used for local purposes. The monetary cost of performing workflow G_m is calculated by multiplying the execution time by the docker container unit pricing. Both processing and communication costs must be considered in the cloud control architecture for IIoT, as well as the balance of employing cloud or edge docker containers.

Without loss of generality, the corresponding cost for task t_i using cloud-edge server is given as

$$C(t_i) = \begin{cases} T(t_i)\cdot c_{\mathrm{edge}}, & x_{m,i} = 1 \\ T(t_i)\cdot c_{\mathrm{cloud}}, & x_{m,i} = 2 \end{cases}$$

where c_{edge} and c_{cloud} are the processing unit price of edge servers and cloud servers, respectively.

The corresponding communication cost from task t_i to task t_j in cloud-edge environment is defined as

$$C(t_i,t_j) = T^t(t_i,t_j)\cdot c_{\mathrm{Bandwidth}}$$

where $c_{\mathrm{Bandwidth}}$ is the communication unit price of network bandwidth.

The total cost of workflow G_m is denoted as

$$C(G_m) = \sum_{i=1}^{m} C(t_i) + \sum_{i=1}^{m}\sum_{j=1}^{m} C(t_i,t_j)$$

which is the total cost of computation plus the cost of transmission.

4.4 Multi-objective optimal scheduling for IIoT workflows

For IIoT workflows, the optimal makespan and cost are necessary for users. Thus, the optimal scheduling problem to minimize cost and makespan is expressed as a multi-objective optimization problem, in which the weight of two conflicting optimization objectives is adjusted. A multi-objective optimization problem is defined as

$$\min : \boldsymbol{F}(x) = (f_1(x), f_2(x), \cdots, f_l(x))^{\mathrm{T}}$$
$$\text{S.t.} \quad x \in X,$$

where X denotes the solution space, and $f_l(x)$ is the lth optimization goal. Pareto optimal theory is introduced to adjust the conflict between makespan and cost objectives in different solutions. Let x_1 and x_2 be two solutions to the multi-objective optimization problem (i.e., $x_1, x_2 \in X$), x_1 would dominate x_2 iff

$$\forall i : f_i(x_1) \leqslant f_i(x_2) \wedge \exists j : f_j(x_1) < f_j(x_2).$$

As shown in Fig. 8, for a multi-objective minimization problem, points A, C, D and E are Pareto-optimal solutions. However, point B is not a Pareto-optimal solution, which is dominated by other solutions.

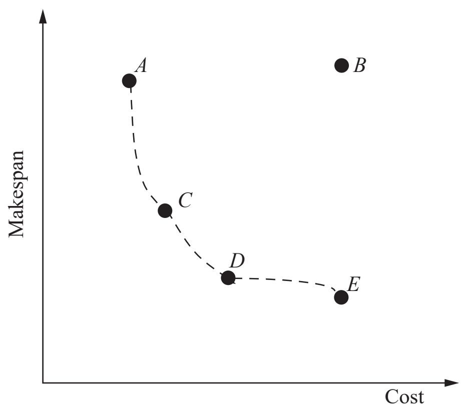


Fig. 8 A simple example of Pareto optimal front

Considering the makespan and cost of an IIoT workflow G_m, which are computed in cloud-edge environment, a responding objective function of G_m is designed as follows:

$$F = \alpha\cdot T(G_m) + (1-\alpha)\cdot C(G_m) \tag{7}$$

where α is a weight parameter, $\alpha \in [0, 1]$. Based on (7), two main factors are took into consideration, for IIoT workflow scheduling, it is well known that a balance between time and cost is maintained. It aims to achieve the lowest possible value in our objective function in (7) with two factors. Thus, a minimum F of function in (7) should be obtained as follows:

$$\min\ F = \alpha T(G_m) + (1-\alpha)C(G_m)$$
$$\text{s.t.} \begin{cases} \sum_{i=1}^{m} C(t_i) \leqslant B_m \\ \sum_{i=1}^{m} T(t_i) \leqslant D_m \end{cases}.$$

By setting the objective function with cost and makespan of IIoT workflows, an adaptive adaptive non-local-convergent particle swarm optimization (ANCPSO) algorithm will be designed to solve the scheduling problem between industrial tasks load and cloud-edge resources.

4.5 ANCPSO algorithm

In general, a workflow multi-objective scheduling problem is solved with the original particle swarm optimization (PSO). However, an optimal solution is not obtained by the typical PSO, because of the random search process. Morevoer, a local extremum is more easily convergented in PSO. To address this challenge, an ANCPSO algorithm is designed to overcome some shortcomings in PSO. Taking full advantage of cloud-edge computing, a mapping strategy will be obtained by deploying our ANCPSO for IIoT workflows in a cloud-edge environment.

The original PSO method, as is generally known, is simpler to build with only a few parameters. The following three steps are used to improve the original PSO algorithm. To begin, the parameter adjustment for non-local-convergent targets should be investigated. In the developed ANCPSO, a typical dynamic approach using non-linear inertia weights balances particles' global and local search ability. Second, various auxiliary procedures are required in order to acquire a better search strategy. To determine the best solution direction, the search algorithm updates velocity and position data with some auxiliary operations. Finally, in order to achieve a non-local-convergent goal, we incorporate both selection and mutation procedures into our technique. The selection operations in the developed ANCPSO are used to identify preeminent particles, and the particle's position is modified during the mutation operation. The aforementioned three improvement characteristics are optimized for local and worldwide search capabilities, allowing the population to evolve more swiftly and precisely to the best answer. Furthermore, when population variety is also provided, all individuals escape slipping into local extremum.

(i) Adjusting inertia weight

It is the central idea of ANCPSO that the optimal solution is searched with the information,which is shared by individuals in a group referred to as a population. Each individual relates to the number of IIoT workflow tasks in a searching space.

One population is assumed to have N particles, the searching space is D. The position $\boldsymbol{X}_i = (x_{i1}, x_{i2}, \cdots, x_{iD})$ and velocity $\boldsymbol{V}_i = (v_{i1}, v_{i2}, \cdots, v_{iD})$ of a particle $P_i (i = 1, 2, \cdots, N)$ are two typical parameters for a particle. In the kth iteration of PSO, the following two equations will update the particle's velocity and position:

$$\begin{cases} V_i^k = w^{(k)} \cdot V_i^{k-1} + c_1 \cdot r_1 \cdot (p\text{best}_i - X_i^{k-1}) + \\ \qquad c_2 \cdot r_2 \cdot (g\text{best} - X_i^{k-1}) \\ X_i^k = X_i^{k-1} + V_i^k \end{cases} \tag{8}$$

where c_1 and c_2 are learning factors, r_1 and r_2 are random values from the range [0,1], and $w^{(k)}$ is inertia weight, which influences particle search capability and alters the solution space's search ability. In the original PSO, the inertia weight w is a decreasing linear function, i.e.,

$$w^{(k)} = w_{\text{start}} - (w_{\text{start}} - w_{\text{end}} \cdot k/N) \tag{9}$$

where w_{start} and w_{end} are the initial and final value of inertia weight w, and N is the maximum number of iterations in PSO.

As a result, this research proposes the following unique updating method:

$$w^{(k)} = w_{\text{end}} + (w_{\text{start}} - w_{\text{end}}) \cdot \sin\left(\frac{\pi}{2}\sqrt{\left(1 - \frac{k}{N}\right)^3}\right). \tag{10}$$

(ii) Updating velocity and position

Cosidering (8) and (9), the updated search direction speed is a completely random procedure. In this case, each particle's search space is large, and certain spots that are distant from the best solution do not require particle search. In our ANCPSO technique, the search process adds two positional variables Xr_i^k and Xl_i^k to each individual in the particle swarm, which are analogous to beetle antennae.

The particle's velocity will be updated by (8), and the particle's position will be updated by the following equation in the kth iteration of ANCPSO:

$$X_i^{k+1} = X_i^k + \lambda V_i^k + (1 - \lambda)\chi_i^k \tag{11}$$

where λ is a positive constant, the increment in particle position is determined as follows:

$$X_i^{k+1} = \sigma^k \cdot V_i^k \cdot \text{sign}(f(Xr_i^k) - f(Xl_i^k)) \tag{12}$$

where $f(Xr_i^k)$ and $f(Xl_i^k)$ are fitness function values for positional variables Xr_i^k and Xl_i^k, respectively, and sign is a symbolic function. If $f(Xr_i^k) - f(Xl_i^k) > 0$, the value is 1, and if $f(Xr_i^k) - f(Xl_i^k) = 0$, the value is 0. When $f(Xr_i^k) - f(Xl_i^k) < 0$, however, the value is equivalent to -1. The step size σ^k can be computed by using the following formula:

$$\sigma^{k+1} = \text{ets} \cdot \sigma^k \ . \tag{13}$$

These two positional variables Xr_i^k and Xl_i^k can be updated by

$$\begin{cases} Xr_i^{k+1} = Xr_i^k + V_i^k \cdot d^k/2 \\ Xl_i^{k+1} = Xl_i^k + V_i^k \cdot d^k/2 \end{cases}. \tag{14}$$

The distance d^k between two antennas is updated by

$$d^k = \sigma^k / c_3 \tag{15}$$

where c_3 is a positive constant.

(iii) Employing selection and mutation operations

Some particles will discover a place area of the best solution while the particle swarm iteration reaches a certain value. When the entire particle swarm achieves a local maximum, it is said to have reached a local optimum, a local optimal position for the optimal area is established, the genuine optimal solution can be obtained. It is impossible to get the global ideal position. Thus, selection and mutation procedures are used to maintain the variety of the entire particle swarm, resulting in a better near-optimal solution for IIoT workflows scheduling.

Setting the mutation probability and randomly generating a number, each mutation is required for the concrete mutation procedure. m vectors are chosen at random for each individual's D-dimensional position, and their previous values are replaced with new random numbers. Consider the two five-dimensional particles represented in Fig. 9, which are undergoing mutation.

Individual number	Individual initial value	Mutation position	Mutation result
1	0 1 2 3 4	2	0 6 2 3 4
2	1 2 3 4 5	4	1 2 3 8 5

Fig. 9 A mutation operation example

Algorithm 1 Employing selection and mutation operations.

```
Input: Number of servers $M$, size of particle swarm $K$, particle dimension $D$.
Output: Mutational position of particle $X'$
1 for $i = 1$ to $K$ do
2   Calculate the fitness value of a given position $X_i$;
3 end
4 Sort all of the particles according to their fitness levels;
5 Set mutation rate to $p$;
6 $t$=Rand[0, 1];
7 for $i = 1$ to $K$ do
8   if $t < p$ then
9     $n$=Rand[0,$K$];
10    pso=Rand[0,$D$];
11    $R$=Rand[0,$M$];
12    Choose particles from the particle swarm that have the number $n$;
13    Temp=$X_{\text{pso}}$;
14  end
15  if Temp $\neq X_{\text{pso}}$ then
16    $X_{\text{pso}} = R$;
17  end
18 end
19 return $X'$.
```

Algorithm 1 shows a thorough implementation of ANCPSO. The fitness value for each particle is computed first (lines 1–3) by this procedure. Then, from tiny to large, sort all of the particles according to their fitness values (see line 4). Then, using the mutation rate as a guide, alter several particles at random (see lines 5–18), where Rand is a function for generating random numbers. Finally, the procedure returns the mutational positions of all particles (see line 19).

(iv) ANCPSO on workflow scheduling

In ANCPSO, a particle's solution space is used to represent an industrial job scheduling plan. In a DAG workflow, each task node corresponds to one particle dimension. To put it another way, the number of process jobs equals the dimension of a particle encoding. The task-to-server service mapping is then generated for each dimension of one particle.

The ANCPSO scheduling algorithm is designed for workflow application based on particle encoding and fitness function. Algorithm 2 shows a detailed implementation of ANCPSO on process scheduling in a cloud-edge scenario.

Algorithm 2 ANCPSO on workflow scheduling

```
Input: Size of particle swarm $K$, number of tasks $N$, maximum numbers of iteration $N_{\max}$.
Output: The near-optimal scheduling solution $S_{\text{best}}$.
1 for $i = 1$ to $K$ do
2   Randomize the schedule initialization $S_i$, the search velocity $V_i$, the position of two antennas $Xr_i^k$, $Xl_i^k$, and a few other parameters;
3 end
4 for $i = 1$ to $K$ do
5   Calculate the worth of a scheduling strategy in terms of fitness $S_i$;
6 end
7 Choose the $K$ scheduling plan with the best near-optimal minimal fitness schedule;
8 for $i = 1$ to $T_{\max}$ do
9   for $i = 1$ to $K$ do
10    Update weight $w^k$;
11    Update the step size $\sigma^k$ and the distance $d^k$;
12    Update the positions of two antennas $Xr_i^k$ and $Xl_i^k$;
13    Calculate the fitness value of $Xr_i^k$ and $Xl_i^k$;
14    Calculate $\chi_i^k$;
15    Update $V_i$;
16    Update the scheduling plan $S_i$;
17    Perform mutation on selected object particles;
18    Servers should be redistributed for tasks in the schedule;
19    Calculate the value of a scheduling plan's fitness $S_i$;
20  end
21  Choose the $K$ scheduling plan with the best near-optimal minimal fitness schedule;
22 end
23 return $S_{\text{best}}$.
```

This algorithm randomly initializes position (i.e., the scheduling plan), the velocity of all particles, and the position of two antennas, among other parameters, in Algorithm 2 (lines 1–7). Then, based on the current approach, update the schedule, including speed, position, selection, mutation, and so on, and reassign the task to the server (lines 10–17). After that, each scheduling plan's fitness value should be calculated, and the global optimal scheduling plan should be updated (lines 18, 19, and 21). Then an almost-ideal schedule is provided (line 23).

5. Efficiency of cloud-edge architecture

In this section, we officially demonstrate the benefit of hierarchical fog computing in terms of boosting cloud resource usage efficiency when handling IIoT peak loads. We first offer a generalized analytical model for a two-tier hierarchical cloud. The research findings based on this two-tier model are then extended to a more complex cloud-edge hierarchy structure.

A two-tier cloud-edge architecture, as shown in Fig. 10, comprises of s servers in tier-1 and one server in tier-2. The computational capacity of the ith tier-1 servers is $f_{\text{edge}}(i)$, and the amount of received IIoT workload is W_i; both $f_{\text{edge}}(i)$ and W_i are measured in central processing unit (CPU) cycles. In tier-2, however, we only have one server with f_{cloud} capacity.

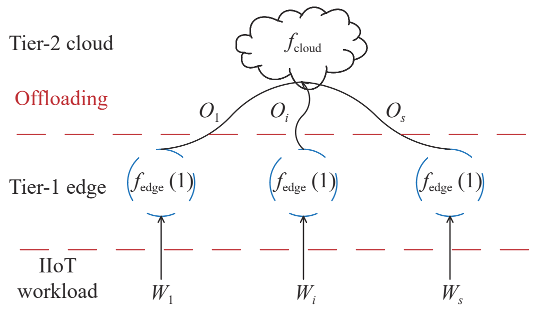


Fig. 10 Two-tier hierarchy of cloud-edge for IIoT workloads

We investigate the benefit of hierarchical cloud-edge on enhancing the efficiency of server resource consumption using the workload models above. Without loss of generality, f_{cloud} will be provisioned to edge servers, when using the same amount of capacity in a flat cloud architecture. Moreover, we also distinguish between flat (one layer) and hierarchical (cloud-edge) systems. To compare and contrast flat and hierarchical cloud-edges, for any computation capacities $f_{\text{edge}}(1)$ and $f_{\text{edge}}(2)$, $f_{\text{edge}}(1)+f_{\text{edge}}(2)=f_{\text{cloud}}$, it is obtained that

$$P(O_1+O_2 \leqslant f_{\text{cloud}}) \geqslant$$
$$P((O_1 \leqslant f_{\text{edge}}(1))P((O_2 \leqslant f_{\text{edge}}(2)).$$

Then for any $f_{\text{edge}}(i) \in [0,\ C]$ and $\sum_{i=1}^{s} f_{\text{edge}}(i) = f_{\text{cloud}}$, there exists

$$P\left(\sum_{i=1}^{s} O_i \leqslant f_{\text{cloud}}\right) \geqslant \prod_{i=1}^{s} P(O_i \leqslant f_{\text{edge}}(i)).$$

When using a fixed amount of processing resources, hierarchical cloud-edge computing has a better possibility of successfully scheduling workloads.

Based on our prior research, we create an analytic cloud-edge hierarchy comparison model that compares multi-tier cloud-edge and flat architectures. Then we partition the offloaded workloads $O_i (i=1,2,\cdots,s)$ into K mutually groups $\{G_1,G_2,\cdots,G_K\}$. Thus, for any $f_{\text{edge}}(i) \in [0,\ C]$ and $\sum_{k=1}^{K} f_{\text{edge}}(k) = f_{\text{cloud}}$, it is given that

$$P\left(\sum_{i=1}^{s} O_i \leqslant f_{\text{cloud}}\right) \geqslant \prod_{k=1}^{K} P\left(\sum_{j\in G_k} O_k \leqslant f_{\text{edge}}(k)\right). \tag{16}$$

When there is only one cloud server in (16), the efficiency of resource consumption in the 2-tier cloud-edge hierarchy will be maximized. The number of edge and cloud servers are the same for $s=K$ in (16); as a result, flat and hierarchical cloud-edge computing are comparable. We can clearly expand the conclusions for hierarchical cloud-edge computing with more than two tiers by recursively repeating the above research.

6. Simulation and performance analysis

The adaptive fuzzy sliding mode control (AFSMC) law $u(t)$ is verified for TCP flows in this section. The usefulness and efficiency of our ANCPSO algorithm for scheduling IIoT operations in a cloud-edge setting will next be tested through multiple groups of tests.

6.1 Simulation on TCP flows control

Let $N=50$, $C=300$ packets/s, $R_0=0.533$s and $q_d=100$ packets from [27]. Then the system matrix parameters are shown as

$$\boldsymbol{A}=\begin{bmatrix} -0.6 & -0.01 \\ 94 & -2 \end{bmatrix},$$

$$\boldsymbol{A}_d=\begin{bmatrix} -0.6 & -0.01 \\ 0 & 0 \end{bmatrix},$$

and

$$\boldsymbol{B}=\begin{bmatrix} 10 \\ 0 \end{bmatrix}.$$

The following uncertain parameters are chosen as

$$\boldsymbol{G}_1(\boldsymbol{x}(t))=\frac{1}{2}\|\boldsymbol{x}(t)\|^2,$$
$$\boldsymbol{G}_2(\boldsymbol{x}(t))=\|\boldsymbol{x}(t-\tau)\|^2,$$
$$\Delta G=0.1\sin t.$$

A sliding surface is given as follows:

$$\boldsymbol{S}(\boldsymbol{x}(t)) = 3\boldsymbol{x}_1(t) + \boldsymbol{x}_2(t).$$

Then adaptive parameters are obtained as $r_1 = 5$, $r_2 = 1$, $r_3 = 8$, $r_4 = 3$, $r_5 = 10$. The membership functions are designed as

$$\mu_1(x_i) = \exp\left[-\left(\frac{x_i + \frac{\pi}{6}}{\frac{\pi}{24}}\right)^2\right],$$

$$\mu_2(x_i) = \exp\left[-\left(\frac{x_i + \frac{\pi}{12}}{\frac{\pi}{24}}\right)^2\right],$$

$$\mu_3(x_i) = \exp\left[-\left(\frac{x_i}{\frac{\pi}{24}}\right)^2\right],$$

$$\mu_4(x_i) = \exp\left[-\left(\frac{x_i - \frac{\pi}{12}}{\frac{\pi}{24}}\right)^2\right],$$

$$\mu_5(x_i) = \exp\left[-\left(\frac{x_i - \frac{\pi}{6}}{\frac{\pi}{24}}\right)^2\right].$$

$u(t)$ is put to the test to see if it is reliable. The value of N ranges from 50 to 130, whereas the value of C is between 300 and 250. Fig. 11 illustrates the simulation findings. Due to incorrect parameters, SMC has a lot of volatility. AFSMC, on the other hand, keeps the current queue length close to the desired queue length. As a result, the AFSMC method performs better under a variety of network conditions and is useful for TCP network congestion control.

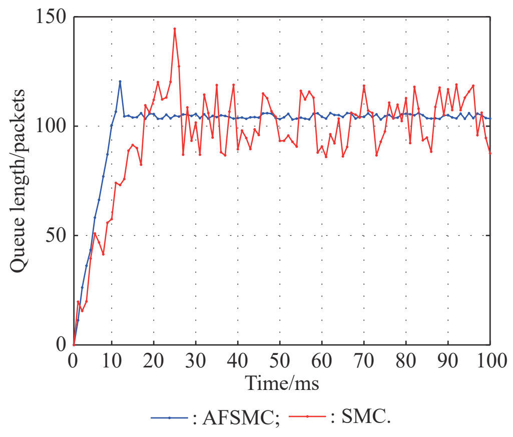


Fig. 11 Queue length responses

6.2 Experiment on ANCPSO for IIoT workflows

This subsection introduces the WorkflowSim simulation framework as well as some information on IIoT workflows. WorkflowSim is extremely close to a fully distributed environment in terms of functionality, and it offers a variety of job scheduling, clustering, and resource provisioning techniques. As a result, this platform may be used to simulate process scheduling in a cloud-edge scenario. The WorkflowSim platform, which includes server resource parameter settings, the ANCPSO algorithm, and other scheduling algorithms, is used to simulate the execution of IIoT processes in a cloud-edge environment in this study.

The size of the ANCPSO population is set to 50, and the particle dimension is set to the number of process tasks. $w_{\text{start}} = 0.9$ and $w_{\text{end}} = 0.4$, respectively, are the initial and final values of inertia weight. Maximum iteration times are set at 300 in ANCPSO. Each round of tests is repeated 10 times to produce an average result, which helps to lessen the impact of experimental uncertainties. Our ANCPSO method, which can find an optimal value for fitness function, runs a series of tests to identify the value of weight coefficient α in (7), as shown in Table 1.

Table 1 Fitness values

α	Fitness value	α	Fitness value
0.1	129.53	0.6	119.53
0.2	128.2	0.7	123.14
0.3	126.98	0.8	108.11
0.4	121.18	0.9	115.12
0.5	119.26		

Using an IIoT workflow G with 30 task nodes, the proposed ANCPSO method is compared against established scheduling algorithms such as the PSO, heterogeneous earliest finish time (HEFT), genetic algorithm (GA), and MIN-MIN algorithms.

On IIoT processes, ANCPSO delivers the best results in terms of cost, time to market, and fitness value, as illustrated in Fig. 12. There are three aspects in total, the original PSO performs worse than the other algorithms. Because PSO particles communicate their location information, they flock together when one of them finds a solution. Among these three metrics, the traditional GA performs somewhat better than the PSO algorithm. The ANCPSO algorithm, on the other hand, combines the benefits of these two methods to discover a superior near-optimal solution. Furthermore, the static scheduling algorithms HEFT and MIN-MIN allocate resources to each task before the scheduling process begins. As a result, ANCPSO achieves low cost, makespan, and fitness values, which may be used to plan IIoT workflows in cloud-edge computing environment.

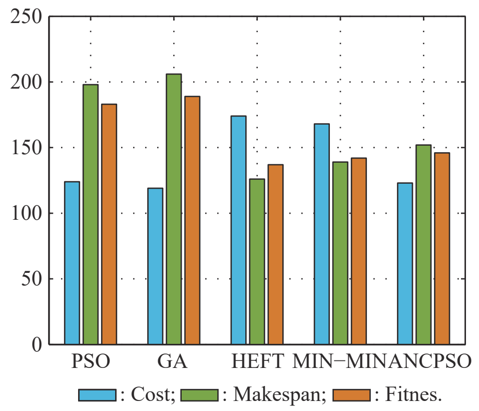


Fig. 12 Comparison of ANCPSO and traditional algorithms

7. Conclusions

In this paper, a cloud control structure has been designed for IIoT in cloud-edge environment with three modes of 5G. For 5G based IIoT, the TSN service is introduced in transmission network. It is a promising approach to extend TSN capabilities over 5G cellular network technologies, which meets the requirements of a highly flexible IIoT in the context of Industry 4.0. A 5G logical TSN bridge is designed to transport TSN streams over 5G framework. For a TCP model with uncertainties, an AFSMC is given with fuzzy rules. A collection of cooperative users from various groups is proposed for usage in collaborative edge and cloud settings with infinite capacity resources. The ANCPSO is built with nonlinear inertia weight to prevent it from slipping into a local optimum, which can drastically lower the makespan and cost.

References

[1] TAO F, CHENG J F, QI Q L. IIHub: an industrial Internet-of-Things hub toward smart manufacturing based on cyber-physical system. IEEE Trans. on Industrial Informatics, 2017, 14(5): 2271–2280.

[2] ZHOU L F, ZHANG L, REN L, et al. Real-time scheduling of cloud manufacturing services based on dynamic data-driven simulation. IEEE Trans. on Industrial Informatics, 2019, 15(9): 5042–5051.

[3] CHEKIRED D A, KHOUKHI L, MOUFTAH H T. IIOT data scheduling based on hierarchical fog computing: a key for enabling smart factory. IEEE Trans. on Industrial Informatics, 2018, 14(10): 4590–4602.

[4] POKHREL S R, QU Y, GAO L. QoS-aware personalized privacy with multipath TCP for IIOT: analysis and design. IEEE Internet of Things Journal, 2020, 7(6): 4849–4861.

[5] SUN J F, YUAN Y, TANG M J, et al. Privacy-preserving bilateral finegrained access control for cloud-enabled IIOT healthcare. IEEE Trans. on Industrial Informatics, 2022, 18(9): 6483–6493.

[6] LYU L, CHEN C L, ZHU S Y, et al. 5G enabled codesign of energyefficient transmission and estimation for IIOT systems. IEEE Trans. on Industrial Informatics, 2018, 14(6): 2690–2704.

[7] CHENG J F, CHEN W H, TAO F, et al. IIOT in 5G environment towards smart manufacturing. Journal of Industrial Information Integration, 2018, 10: 10–19.

[8] XIA Y Q. Cloud control systems. IEEE/CAA Journal of Automatica Sinica, 2015, 2(2): 134–142.

[9] XIA Y Q, QIN Y, ZHAI D H, et al. Further results on cloud control systems. Science China Information Sciences, 2016, 59(7): 1–5.

[10] TAO F, ZUO Y, XU L D, et al. IoT-based intelligent perception and access of manufacturing resource toward cloud manufacturing. IEEE Trans. on Industrial Informatics, 2014, 10(2): 1547–1557.

[11] WANG C G, BI Z M, XU L D. IoT and cloud computing in automation of assembly modeling systems. IEEE Trans. on Industrial Informatics, 2014, 10(2): 1426−1434.

[12] YUAN H H, XIA Y Q, ZHANG J H, et al. Stackelberg-game-based defense analysis against advanced persistent threats on cloud control system. IEEE Trans. on Industrial Informatics, 2019, 16(3): 1571–1580.

[13] LIU G P. Predictive control of networked multiagent systems via cloud computing. IEEE Trans. on Cybernetics, 2017, 47(8): 1852–1859.

[14] TAN H R, MIAO Z Q, WANG Y N, et al. Data-driven distributed coordinated control for cloud-based model-free multiagent systems with communication constraints. IEEE Trans. on Circuits and Systems I: Regular Papers, 2020, 67(9): 3187–3198.

[15] RAYATI M, RANJBAR A M. Resilient transactive control for systems with high wind penetration based on cloud computing. IEEE Trans. on Industrial Informatics, 2017, 14(3): 1286–1296.

[16] TAO F, LAILI Y J, XU L D, et al. FC-PACO-RM: a parallel method for service composition optimal-selection in cloud manufacturing system. IEEE Trans. on Industrial Informatics, 2012, 9(4): 2023−2033.

[17] LIU Q, MA Y J, ALHUSSEIN M, et al. Green data center with IoT sensing and cloud-assisted smart temperature control system. Computer Networks, 2016, 101: 104–112.

[18] TRAN T X, HAJISAMI A, PANDEY P, et al. Collaborative mobile edge computing in 5G networks: new paradigms, scenarios, and challenges. IEEE Communications Magazine, 2017, 55(4): 54–61.

[19] TALEB T, SAMDANIS K, MADA B, et al. On multi-access edge computing: a survey of the emerging 5G network edge cloud architecture and orchestration. IEEE Communications Surveys and Tutorials, 2017, 19(3): 1657–1681.

[20] ZHANG T, WANG J X, HUANG J W, et al. Tuning the aggressive TCP behavior for highly concurrent HTTP connections in intra-datacenter. IEEE/ACM Trans. on Networking, 2017, 25(6): 3808–3822.

[21] BALASUBRAMANIAN V, ALOQAILY M, REISSLEIN M. An SDN architecture for time sensitive IIOT. Computer Networks, 2021, 186: 107739.

[22] LYU J, ZHAO Y X, WU X, et al. Formal analysis of TSN scheduler for real-time communications. IEEE Trans. on Reliability, 2020, 70(3): 1286–1294.

[23] ROMANOV A M, GRINGOLI F, SIKORA A. A precise synchronization method for future wireless TSN networks. IEEE Trans. on Industrial Informatics, 2020, 17(5): 3682–3692.

[24] ZHAO L X, POP P, GONG Z J, et al. Improving latency analysis for flexible window-based GCL scheduling in TSN networks by integration of consecutive nodes offsets. IEEE

Internet of Things Journal, 2020, 8(7): 5574–5584.
[25] MISRA V, GONG W B, TOWSLEY D. Fluid-based analysis of a network of AQM routers supporting TCP flows with an application to RED. Proc. of the Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication, 2020: 151–160.
[26] HOLLOT C V, MISRA V, TOWSLEY D, et al. Analysis and design of controllers for AQM routers supporting TCP flows. IEEE Trans. on Automatic Control, 2002, 47(6): 945–959.
[27] QUET P F, OZBAY H. On the design of AQM supporting TCP flows using robust control theory. IEEE Trans. on Automatic Control, 2004, 49(6): 1031–1036.
[28] SHAO Y L, LI C L, TANG H L. A data replica placement strategy for IoT workflows in collaborative edge and cloud environments. Computer Networks, 2019, 148: 46–59.
[29] AFRIN M, JIN J, RAHMAN A, et al. Multi-objective resource allocation for edge cloud based robotic workflow in smart factory. Future Generation Computer Systems, 2019, 97: 119–130.
[30] XU X L, LIU Q X, LUO Y, et al. A computation offloading method over big data for IoT-enabled cloud-edge computing. Future Generation Computer Systems, 2019, 95: 522–533.
[31] XIE Y, ZHU Y W, WANG Y G, et al. A novel directional and nonlocal-convergent particle swarm optimization based workflow scheduling in cloud-edge environment. Future Generation Computer Systems, 2019, 97: 361–378.
[32] WANG Y M, YANG S S, REN X B, et al. IndustEdge: a timesensitive networking enabled edge-cloud collaborative intelligent platform for smart industry. IEEE Trans. on Industrial Informatics, 2021, 18(4): 2386–2398.

Biographies

YAN Ce was born in 1991. He received his B.S. degree in mathematics and applied mathematics from Hebei University of Science and Technology, Shijiazhuang, China, in 2014, and M.E. degree in control theory and control engineering from Yanshan University, Qinhuangdao, China, in 2017. He is currently working toward his Ph.D. degree in control science and engineering in Beijing Institute of Technology, Beijing, China. His research interests include cloud control systems and application, cloud workflow scheduling, industrial IoT and intelligent manufacturing systems.
E-mail: yancemc@163.com

XIA Yuanqing was born in 1971. He received his M.S. degree in fundamental mathematics from Anhui University, China, in 1998 and Ph.D. degree in control theory and control engineering from Beihang University, Beijing, China, in 2001. He is now the dean of School of Automation, Beijing Institute of Technology. His research interests are cloud control systems, networked control systems, robust control and signal processing, active disturbance rejection control, and unmanned system control.
E-mail: xia_yuanqing@bit.edu.cn

YANG Hongjiu was born in 1981. He received his B.S. degree in mathematics and applied mathematics and M.S. degree in applied mathematics from Hebei University of Science and Technology, Shijiazhuang, China, in 2005 and 2008, respectively. He received his Ph.D. degree in control science and engineering from Beijing Institute of Technology, Beijing, China. He was an associate professor with the Department of Automation, Institute of Electrical Engineering, Yanshan University, Qinhuangdao, China, in 2013 and 2018. He is currently a professor with the Department of Automation, School of Electrical and Information Engineering, Tianjin University, China. His main research interests include robust control/filter theory, delta operator systems, networked control systems, and active disturbance rejection control.
E-mail: yanghongjiu@tju.edu.cn

ZHAN Yufeng was born in 1990. He received his Ph.D. degree in control science and engineering from Beijing Institute of Technology in 2018. He is an assistant professor with the School of Automation, Beijing Institute of Technology, Beijing, China. His research interests include networking systems, game theory, and machine learning.
E-mail: yu-feng.zhan@bit.edu.cn