Towards Computing Resource Reservation Scheduling in Industrial Internet of Things

Fan Liang*, Wei Yu*, Xing Liu*, David Griffith†, and Nada Golmie†
*Towson University, USA
Emails: {fliang1,xliu10}@students.towson.edu, wyu@towson.edu
†National Institute of Standards and Technology (NIST), USA
Emails:{david.griffith, nada.golmie}@nist.gov

***Abstract*—The Industrial Internet of Things (IIoT) is a critically important implementation of the Internet of Things (IoT), connecting IoT devices ubiquitously in an industrial environment. Based on the interconnection of IoT devices, IIoT applications can collect and analyze sensing data, which help operators to control and manage manufacturing systems, leading to significant performance improvements and enabling automation. IIoT systems are characterized by a variety of IIoT applications, which generate different computing tasks depending on their functionalities. Some tasks are time-sensitive, while others are not, and more importantly, some tasks are non-preemptive in IIoT scenarios. Thus, processing the different IIoT applications efficiently in an IIoT environment is key to achieving automation. Since computing resources are limited in IIoT, how to rapidly process time-sensitive tasks is a critical issue. Although some existing scheduling schemes can deal with the latency requirements of time-sensitive tasks, they lack consideration for non-preemptive tasks. To address this issue, in this paper we consider a typical smart warehouse system as an example and propose a generic task scheduling scheme that reserves computing resources to wait for upcoming time-sensitive tasks in such an IIoT environment. In doing so, our proposed scheme is capable of minimizing the overall waiting time for time-sensitive tasks. To evaluate the proposed scheme, we have implemented a simulation platform for a smart warehouse and conducted extensive experiments. Our experimental results demonstrate the efficacy of our scheme, which can allocate computing resources so that the processing time for the time-sensitive tasks can be reduced. Additionally, we discuss some potential research directions toward improving performance in IIoT environments with respect to resource management, machine learning, and security and privacy.**

***Keywords*—Industrial IoT, Scheduling, Resource Allocation, Edge Computing.**

I. Introduction

The Industrial Internet of Things (IIoT) is known as one of the key enablers of the fourth industrial revolution, which is a typical implementation of the Internet of Things (IoT) in a body of manufacturing systems [1]. Manufacturing systems can collect useful data from a massive number of IoT devices. Leveraging advanced data analysis and machine learning techniques, IIoT can assist in monitoring and controlling industrial systems automatically [2], [3]. To realize automation, a number of applications have been developed in the IIoT system to achieve specific functionalities. Those applications generate different types of tasks, such as data analysis tasks, system control tasks, and route planning tasks, among others.

Furthermore, due to the characteristics of the IIoT system, some computing tasks are non-preemptive. Generally speaking, from the perspective of response time, we can categorize tasks into two groups: time-sensitive (TS) and non-time-sensitive (non-TS). Due to the information exchange between centralized servers and IIoT sensors, massive amounts of data are transmitted via the IoT network, which raises significant network overhead. In this situation, the centralized network structure cannot provide desirable services for TS applications. Thus, how to optimize network and computing resources while handling the different types of applications with different performance requirements becomes a critical problem in IIoT.

The total response time for tasks is the combination of the transmission time and computing time. To reduce transmission times, one feasible method is to optimize the network structure. Specifically, edge computing is a promising paradigm that offers the benefits of offloading computing tasks from cloud servers to edge computing nodes [3], [4], [5]. Unlike cloud computing, in which tasks are offloaded to remote cloud data centers, edge computing assigns computation tasks to multiple edge nodes, which are close to end users. Thus, edge computing is capable of reducing the amount of data transmission and network traffic between cloud servers and IoT sensors. By doing this, edge computing can be deployed in IIoT environments such that transmission time can be reduced.

In addition to reducing transmission time, reducing computing time is important in IIoT environments. Computing time can be reduced by continuously optimizing task scheduling. As we mentioned before, in IIoT environments, TS tasks require real-time or near real-time response. Moreover, the tasks are non-preemptive in some cases, raising challenges for task scheduling. Existing task scheduling algorithms in IIoT are generally based on the quality of service (QoS) requirements for individual tasks. For example, scheduling methods can assign a weight based on the priority of tasks so that higher priority tasks have a better chance of using resources [6], [7], [8]. Nonetheless, these existing algorithms cannot avoid waiting time in IIoT with non-preemptive tasks. In this case, even when a TS task arrives, it still needs to wait until executed tasks are completed so that computing resources are released. The waiting time of this increases the response time for the TS tasks, which is unacceptable or infeasible for those tasks. Thus, it is necessary to design a proper task scheduling algorithm that guarantees response times to satisfy TS and non-preemptive

tasks.

To address the aforementioned issues, in this paper we propose a new *Computing Resource Reservation Scheduling (CRRS)* scheme with the intention that the turnaround time for TS tasks in IIoT environments can be reduced. The proposed scheduling scheme reserves computing resources according to the probability distribution of task occurrence. The reserved computing resources wait for upcoming TS tasks so that the incoming TS tasks can be executed without any waiting time. To demonstrate our idea, we first design an IIoT scenario that deploys an edge computing infrastructure, and several TS and non-TS tasks are generated. Based on the scenario, we design a system model and utilize a length-adjustable sliding window to identify the prediction step length of upcoming TS tasks. By doing this, our proposed scheme can reserve computing resources for the upcoming TS tasks, so that the TS tasks can be executed immediately. Further, we carry out a comprehensive evaluation, which demonstrates that our proposed resource allocation scheme can achieve better performance than representative baseline task scheduling schemes. Further, we discuss extensions of the proposed scheme in Section VI. It is worth noting that the methodology in our proposed CRRS scheme is a generic one and can be applied to other IIoT systems to assist in resource management. In this paper, we use a typical IIoT system (i.e., smart warehouse) as an example to show the efficacy of our proposed CRRS scheme.

To summarize, we make the following contributions in our study:

First, we propose a new Computing Resource Reservation Scheduling (CRRS) scheme that focuses on reducing the turnaround time for TS tasks in the IIoT environment. To be specific, the proposed scheme is based on the probability distribution of task occurrence, which is used to reserve computing capacity for upcoming TS tasks. By doing this, the upcoming TS tasks can execute immediately.

Second, we consider a typical IIoT smart warehouse scenario and design a system model. Based on the model, we evaluate the proposed scheme. In addition, we design a Python-based experimental environment to simulate the IIoT scenario and scheduling scheme. Furthermore, utilizing the evaluation environment, we define the evaluation metrics and evaluate the performance of our proposed scheduling scheme comprehensively, in comparison with some baseline scheduling schemes. Additionally, we discuss several research directions as extensions.

The remainder of this paper is organized as follows: In Section II, we briefly introduce the key techniques and concepts of IIoT and edge computing. In Section III, we introduce our scheme in detail. In Section IV, we present the evaluation environment settings and experimental design. In Section V, we define the evaluation metrics and describe the evaluation results. In Section VI, we discuss some remaining issues related to our study. In Section VII, we conduct a brief literature review of related studies on IIoT and task scheduling. Finally, we summarize the paper in Section VIII.

II. Preliminaries

In this section, we introduce some preliminaries of IIoT and edge computing.

IIoT Systems and Applications: As discussed above, the IIoT system provides network connection for IoT devices, such as monitors, sensors, and controllers. The IIoT applications run on this infrastructure to achieve specific functions and support the automation of the IIoT system. From a cyber-physical system perspective, it is composed of a physical subsystem, network subsystem, and application subsystem, which cooperate with each other so that the manufacturing process can be controlled automatically [1].

In detail, the physical subsystem consists of massively deployed IoT devices, and the network subsystem provides a reliable and efficient connection to support the communication of the physical subsystem. In addition, the application subsystem provides services and functionalities for the IIoT system to achieve automation. Since the IIoT system is dynamic and deployed in a wide area, the IIoT system can be considered as a distributed system [9], [10], [11]. In the IIoT system, all the information is transmitted by the network subsystem. Furthermore, in order to support automation and intelligence for IIoT applications, a large amount of data will be collected and analyzed. The massive data is transmitted via the network, creating substantial overhead to the network subsystem. Thus, network performance is one of the key factors that affect the performance of IIoT applications.

The application subsystem could generate massive amounts of computing tasks [12]. Depending on the purposes of the tasks, they may be TS or non-TS tasks [13]. For example, precision control and environmental perception tasks are typical TS tasks. The TS tasks require fast handling and rapid computing, in order to adapt to the rapid status changes of industrial systems. Furthermore, tasks could be non-preemptive in some cases. Although optimizing network performance can reduce the transmission time for TS tasks, it still cannot meet the requirements of TS tasks in IIoT, especially for those that are non-preemptive. Thus, it is necessary to design a scheduling algorithm to tackle this problem.

Edge Computing: Optimizing network performance is one viable way to reduce the response times for TS tasks. Edge computing is an active distributed computing architecture that is widely integrated in IIoT systems. It offloads computation tasks from cloud to edge nodes to provide computation at shorter distances to end users [14], [15]. Thus, by leveraging edge computing, users can send tasks to nearby edge computing nodes that reduce network traffic and avoid network congestion [16]. Similar to edge computing, IIoT is a distributed system and edge computing offers latency reduction benefits for TS tasks in IIoT. Nonetheless, edge nodes have limited computation capacity relative to cloud computing, and generally requires longer computing time. Furthermore, task exchanging and synchronization between edge nodes could affect computing tasks, as they are distributed to heterogeneous edge nodes that must cooperate. Thus, it calls for design an efficient task scheduling scheme that can reduce the turnaround time for tasks.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2020.3044057, IEEE Internet of Things Journal

TABLE I
NOTATIONS

Symbols	Descriptions
W	A list to represent the warehouse
l, w	The length and width of warehouse
τ	The coverage efficiency of wireless network
A	A list to represent access points (APs)
X, Y	The location coordinate of APs
r	The communication radius of APs
$\mathbb{C}$	The computing capacity
M	A list to represent unmanned vehicles
x, y	The location coordinate of unmanned vehicles
$\vec{\nu}$	The speed vector of unmanned vehicles
$\mathbb{T}$	A list to represent tasks
θ	The type of tasks
δ	The smallest computing capacity
n	The number of computing slots
R	The communication coverage range of AP
T_t, T_e, T_w	Turnaround, execution, and waiting times
k	The number of packets in each computing slot
λ	The average number of packets in a computing slot
μ	The average of computing requirements
σ^2	The variance of computing requirements
D	The distance between AP and unmanned vehicle
N	The number of non-TS tasks

III. OUR SCHEME

In this section, we introduce our scheme in detail. Particularly, we first present the design rationale and outline the problem space. Then, we introduce the investigated IIoT scenario. Based on the scenario, we detail the system model and our scheme. Finally, we introduce our designed CRRS algorithm. Table I lists key notations in this paper.

A. Design Rationale

Fig. 1 illustrates the problem space of IIoT, which consists of three dimensions (i.e., QoS Requirements, Physical Resources, and System Structure). In this paper, we define the tasks in IIoT as TS task and non-TS task. In order to reduce the response time for TS tasks, distributed computing (i.e., edge computing and fog computing) will be viably utilized in IIoT environments, since distributed computing can reduce the transmission time of network packets. Nonetheless, simply applying edge computing to optimize the network is not sufficient. As we mentioned in Section II, how to optimize computing performance for edge computing resources is critical. In the following, we focus on reducing the response time for TS tasks in an IIoT system. Specifically, we propose a new reservation-based task scheduling scheme, which reserves computing resources based on the probability distribution of TS tasks occurring. By doing so, there are always available computing resources to execute upcoming TS tasks. The solid blue sectors in the figure indicate our area of focus in this paper.

We now introduce our design rationale that focuses on reducing the response time for TS tasks. Recall that the main

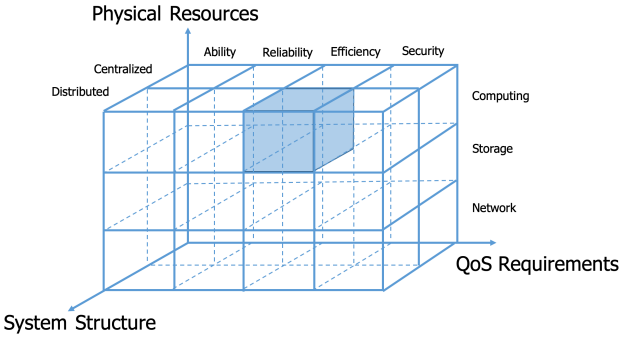

Fig. 1. Problem Space of IIoT

purposes of IIoT are achieving industrial system automation and increasing system efficiency. The IIoT system collects relevant data, and IIoT applications analyze the collected data and control the system to realize system automation. Thus, IIoT applications generate different types of computing tasks, such as data analysis, system control, and others. We define the computing tasks as TS and non-TS tasks. For instance, a typical TS task in IIoT is the control signal, which is the core heartbeat of the IIoT system, such that it requires very short computing time to complete control tasks. In addition, since some industrial systems are non-interruptable, and the tasks are non-preemptive as well, which intensifies the complexity of scheduling. To this end, we focus on designing a new scheduling scheme. The core idea of our scheme is to reserve computing resources based on the probability distribution of TS task occurrence. The reserved computing resources are dedicated for upcoming TS tasks and used to execute TS tasks immediately when they arrive. Obviously, our designed scheme is capable of reducing the overall turnaround time by reducing wait times for TS tasks in IIoT.

B. Motivated Scenario

We now introduce the investigated IIoT scenario in detail. In a typical smart warehouse, such as Amazon warehouses [17], unmanned vehicles move the packages around the warehouse according to various requirements. The unmanned vehicles are operated by the IIoT system. In detail, the IIoT system computes the routes and sends the routes to the unmanned vehicles, which follow the routes and carry the packages to their destinations. Meanwhile, the unmanned vehicles send location information and surrounding event information back to the IIoT system while traveling. Based on the information, the IIoT system can control individual unmanned vehicles to avoid collisions and other accidents.

In this scenario, we define the following two types of tasks: (i) *Route generation*, which is a non-TS task. Based on the location of the unmanned vehicles and the packages, the routes are generated by the computing resources. During the route generation, the unmanned vehicles stop and wait until receiving the route information. Since the waiting status does not cause any failures of the system, the route generation tasks are non-TS tasks. (ii) *Collision avoidance*, which is a TS task.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2020.3044057, IEEE Internet of Things Journal

In the warehouse, unmanned vehicles perceive surrounding environments by collecting and analyzing relevant environment data. Nonetheless, data analysis incurs large computing loads. Due to the computing capacity limitation, the unmanned vehicles send the related environment data to the distributed computing resources to analyze the data. Because the states of the surrounding environments are changing rapidly, computing resources are required to calculate the next action in a short time period. Thus, collision avoidance tasks require real-time response and are TS tasks.

Fig. 2 illustrates the system structure of the investigated scenario. All the unmanned vehicles are located at the bottom, and the edge computing clusters are located at the top. The APs connect with unmanned vehicles via wireless networks and connect with edge computing clusters via wired networks. Likewise, the APs connect with each other via wired networks. Based on this scenario, the unmanned vehicles send two types of tasks, which are non-TS (i.e., route calculation) and TS (i.e., collision avoidance). All the tasks are sent by the unmanned vehicles via wireless networks to the nearest APs, which distribute the tasks to edge computing clusters.

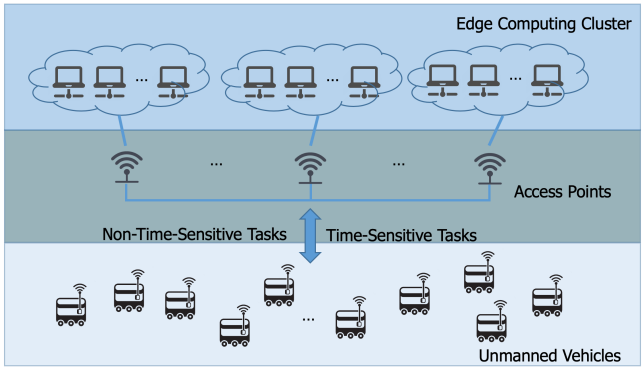

Fig. 2. System Structure

C. System Model

Following the scenario described in Section III-B, we now design the system model. Denote a large warehouse as $W\{l, w, \tau\}$, where l is the length and w is the width of the warehouse. Also, τ denotes the coverage efficiency of the APs in the warehouse.

Definition 1. The coverage efficiency is defined by the ratio of the union of the communication coverage area for the APs and the total area of communication circle for the APs. Thus, the coverage efficiency τ can be represented by

$$\tau = \frac{R_1 \bigcup R_2 \cdots \bigcup R_i}{\sum_{1,2,3,\ldots,i} R_i}. \quad (1)$$

Here, R denotes the area of the communication coverage range. Based on Equation (1), τ is a number less than 1. A larger τ means fewer communication overlaps between neighboring APs.

The APs in the warehouse provide wireless communication between the unmanned vehicles and computing resources. In order to identify the locations of the APs, we consider coverage and interference. First, because the unmanned vehicles constantly communicate with computing resources, full wireless network coverage is required throughout the entire warehouse. Second, the wireless communication range of an AP is a circle. As we know, leveraging several circles to cover rectangle causes overlaps between neighboring circles. Thus, to fully cover the warehouse, there are communication overlaps between neighboring APs. The communication overlaps cause interference. To reduce interference, the smallest communication overlap between neighboring APs should be used.

Denote unmanned vehicles as $M\{x, y, \vec{\nu}\}$, where x and y are the location coordinates of unmanned vehicles and $\vec{\nu}$ is a vector that represents the speed of unmanned vehicles. In addition, denote APs as $A\{X, Y, r, \mathbb{C}\}$, where X and Y are the location coordinates of the AP, r is the communication radius, and $\mathbb{C}$ is the computing capacity of the edge computing cluster that connects with AP: A.

D. Computing Resource Reservation Scheduling (CRRS) Algorithm

We now focus on the reduction of turnaround time for TS tasks. To do so, we propose a new scheduling algorithm to reduce turnaround time. As we know, turnaround time can be computed via $T_t = T_e + T_w$, where T_e represents the execution time and T_w represents the waiting time. Since the computing capacity is constant, reducing the waiting time T_w for tasks is the only viable way to reduce T_t. Our proposed CRRS scheme focuses on minimizing the waiting time for TS tasks. In detail, the CRRS scheme reserves computing resources based on the probability distribution of TS task occurrence. The reserved computing resources are utilized for executing the upcoming TS tasks. Since the CRRS scheme reserves computing resources, no starvation occurs in the system. In other words, the TS tasks can be immediately executed without any waiting time. We identify the computing resources as computing slots, which all have the same computing capacity. Here, we define the computing slot as follows:

Definition 2. The computing slot is the smallest indivisible computing time period of execution (smallest CPU cycle) in computing resources. Denote δ as the computing slot. We then define a task as $\mathbb{T}\{\theta, n \cdot \delta\}$. Here, θ denotes the types of the tasks, $\theta = 1$ denotes the TS tasks and $\theta = 0$ denotes the non-TS tasks. In addition, n denotes the number of computing slots that the task occupies. Thus, $n \cdot \delta$ denotes the length of tasks.

In our scenario, we assume that the surrounding events of each unmanned vehicle are independent and no correlations between different events exist. In addition, the surrounding events are the triggers of TS tasks. Thus, TS tasks are independent, and we assume the generation of TS tasks follows the *Poisson* distribution. The Poisson distribution mass function is as follows:

$$f(k, \lambda) = Pr(X = k) = \frac{\lambda^k e^{-\lambda}}{k!}. \quad (2)$$

Here, k represents the number of TS tasks generated in each computing slot and λ represents the average number of TS tasks generated in a computing slot. In addition, we assume the computing slot requirements for TS tasks are the same and each TS task occupies '1' computing slot, which is δ.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2020.3044057, IEEE Internet of Things Journal

Based on the classification of TS and non-TS tasks, non-TS tasks are the complement set of TS tasks, because unmanned vehicles do not send surrounding information to the computing resources during the route generation process. Thus, the generation of non-TS tasks can be represented by $Pr(X) = 1 - Pr(TS)$. Furthermore, we assume the computing slot requirements of each non-TS task (task length) are random and follow the normal distribution.

$$f(x, \mu, \sigma) = \frac{1}{\sigma\sqrt{2\pi}} \mathbf{exp}\left(\frac{-(x-\mu)^2}{2(\sigma)^2}\right). \tag{3}$$

We assume the computing slot requirements of non-TS tasks follows $X \sim N(\mu, \sigma^2)$. We utilize Equation (4) to represent the average value of computing slot requirement, where n_i denotes the different number of computing slots that task i occupies and N denotes the number of non-TS tasks.

$$\mu = \bar{n_i} \cdot \delta = \frac{\sum_{i=1,2,3\cdots k} n_i \cdot \delta}{N}. \tag{4}$$

Likewise, we utilize Equation (5) to represent the variance of the computing slot requirement.

$$\sigma^2 = \frac{\sum_{i=1,2,3\cdots k} (n_i \cdot \delta - \bar{n_i} \cdot \delta)^2}{N}. \tag{5}$$

Based on the communication range of APs, unmanned vehicles upload TS tasks to different APs during travel. To identify the best AP for uploading TS tasks, we compute the distance between unmanned vehicles and APs, in order to select the nearest AP to upload the task. Obviously, the distance $D_{i,j} = \sqrt{(X_j - x_i)^2 + (Y_j - y_i)^2}$. From the AP communication range perspective, all unmanned vehicles with distance $D_{i,j} < r$ can connect with AP N_j. By obtaining the number of unmanned vehicles n_j connected to AP N_j, we can then calculate the number of TS tasks received by AP N_j,

$$\sum_{n_i} \sum_k k \cdot Pr(n_{i_k}) (Pr(n_{i_k}) > Pr(m)). \tag{6}$$

Here, n_{i_k} represents the probability of a certain number of the upcoming TS tasks for all unmanned vehicles in the communication range of an AP. $Pr(n_{i_k}) > Pr(m)$ means that if the probability of a certain number of upcoming TS tasks is greater than a threshold, the CRRS scheme reserves the computing capacity for those upcoming TS tasks.

Algorithm 1: Computing Resource Selection

Data: A_j: AP, n_j: the number of unmanned vehicles connected to AP, M_i: unmanned vehicles
Output: t

initialization
$n_j = 0$
while $\sqrt{(X_j - x_i)^2 + (Y_j - y_i)^2} < r$ **do**
  $n_j = n_j + 1$
  Check the index i and put i into a list
  Record M_i according to i, put M_i into a 3 column n_j row list
  Obtain the speed $\vec{\nu_i}$ of each M_i
  if $t \cdot \vec{\nu_i} > r$ **then**
    Obtain t, indicating at time t, M_i will arrive in next AP's coverage
    Send message to neighbor AP A_{j+1}: M_i will arrive in next $t_{\theta=1}$ time
  else
    continue

Based on the system model, we now present the algorithms in detail. Algorithm 1 presents the method, by which each AP discovers the surrounding unmanned vehicles in communication range. The APs discover the surrounding unmanned vehicles simply by calculating the distance between the target unmanned vehicles and themselves. Then, the APs obtain the details of the unmanned vehicles, including the task types and unmanned vehicle speeds. Also, based on the speeds of the unmanned vehicles, the APs will notify neighboring APs as to whether the unmanned vehicles will travel into the neighbor's communication range during the next computing slot. If the unmanned vehicle will travel to a neighboring AP's communication range, the neighboring AP will reserve computing capacity. Based on analysis of the algorithm, the time complexity of the algorithm is $O(m) \approx O(N)$, where m is the number of unmanned vehicles in the communication range of an AP and N is the scale of the problem.

Algorithm 2: Computing Resource Reserved Scheduling

Data: Aj: AP, n_j: the number of unmanned vehicles connected to AP, $\mathbb{C}_j$: the computing capacity of the computing resource, M_i: unmanned vehicle, $\mathbb{T}$: the tasks

initialization
$n_{TS} = 0$
$n_{Non-TS} = 0$
while $A_j; (j = 1, 2, 3 \cdots, k)$ **do**
  Update the remaining computing capacity $\mathbb{C}_j$
  while $M_i; (i = 1, 2, 3 \cdots, m)$ **do**
    if $\theta_i = 0$ **then**
      $n_{Non-TS} = n_{Non-TS} + 1$
      Push the task M_i to the waiting list
      Predict the length of M_i:$f(x, \mu, \sigma) = \frac{1}{\sigma\sqrt{2\pi}} \mathbf{exp}\left(\frac{-(x-\mu)^2}{2(\sigma)^2}\right)$
      Obtain the shortest non-TS tasks:$\mathbb{T}\left\{0, t_c, \hat{\delta}\right\}$
    else
      $n_{TS} = n_{TS} + 1$
    Push the TS task to the computing resources
    $Num = 0$
    while $Num < \mathbb{C}_j$ **do**
      $\mathbb{C}_j = \mathbb{C}_j - n_{TS}$
      $Num = \sum_{n_i} \sum_k k \cdot Pr(n_{i_k}) (Pr(n_{i_k}) > Pr(m))$
      Obtain the shortest non-TS task in the waiting list
      if $\mathbb{C}_j - Num > 0$ **then**
        push this Non-TS to execute
      else
        hold
      if $\mathbb{C} = 0$ **then**
        push all the tasks into waiting list
        hold;
      else
        Deploy the shortest n_{non-TS} task in the waiting list

Algorithm 2 presents our proposed CRRS scheme in detail. Generally speaking, the CRRS reserves computing slots based on the probability distribution of TS tasks. Then, CRRS checks the remaining computing slots and sends a query to the non-TS task waiting list. The non-TS waiting list responds to the query and returns the shortest non-TS task on the waiting list. Then, the CRRS pushes the shortest non-TS task to the computing resource to execute. After that, if empty computing slots still

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2020.3044057, IEEE Internet of Things Journal

remain, the CRRS sends another query to the non-TS task waiting list to obtain the next shortest non-TS task. In addition, the CRRS projects upcoming TS tasks over the length of time required for the incoming non-TS task. If the empty computing slots are sufficient to execute the TS tasks, the CRRS pushes the non-TS task to the computing resource to execute; if not, the CRRS holds the non-TS task until the next time slot. Based on our analysis, the time complexity of this algorithm for each AP is $O(m \cdot \mathbb{C} + n^2) \approx O(2N^2)$, where m is the number of unmanned vehicles in the communication range of an AP, $\mathbb{C}$ is the computing capacity of this AP, and n denotes the length of the waiting list for the non-TS tasks. Again, N is the scale of problem.

IV. Implementation

In this section, we introduce the implementation to validate our scheme. We first define the simulation programs and identify the related parameters. Then, we set up an evaluation environment using the Common Open Research Emulator (CORE) [18], [19] to deploy the programs[1]. Finally, we present the settings of the environment used to validate the proposed scheduling algorithm.

A. Implementation and Parameter Settings

To validate our approach, we first implement the proposed scheduling scheme using Python. We design three components, written using Python 3.7, including AP Deployer, Unmanned Vehicle Routing Generator, and CRRS Scheduling Program. First, the AP Deployer calculates the specific locations for the APs, in order to deploy the APs to the evaluation environment. Here, without loss of generality, we set the shape of the warehouse as a square with side length l (say equal to 200 m). Also, we set the communication radius r of the APs (say equal to 20 m). Based on the scenario defined in Section III-B, the unmanned vehicles communicate with the computing resources via wireless networks. Therefore, wireless communication needs to fully cover the warehouse. Recall that Definition 1 defines the coverage efficiency τ, which is a number smaller than 1. A higher coverage efficiency represents smaller communication overlaps. In order to reduce the interference of the wireless network, it is necessary to find the maximum value of τ.

Fig. 3 shows the minimum overlap for three circles. To obtain the minimum overlap, the three circles must intersect at a point. We assume the radius of the circles is r. Since the intersection is a point, we obtain the distance between each center as $\sqrt{3} \cdot r$. Then, based on Equation (1), we compute the maximum coverage efficiency $\tau = \frac{R_1 \bigcup R_2 \cdots \bigcup R_i}{\sum_{1,2,3,\cdots i} R_i} = \frac{\pi r^2 + 3\sqrt{3}}{3\pi r^2} = 82.7\,\%$. Following the calculation, we obtain the locations for all APs. Then, we utilize the Unmanned Vehicle Routing Generator to randomly generate various routes

[1]Certain commercial equipment, instruments, or materials are identified in this paper in order to specify the experimental procedure adequately. Such identification is not intended to imply recommendation or endorsement by the National Institute of Standards and Technology, nor is it intended to imply that the materials or equipment identified are necessarily the best available for the purpose.

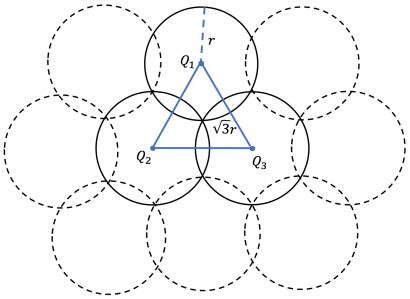

Fig. 3. The Maximum Coverage Efficiency

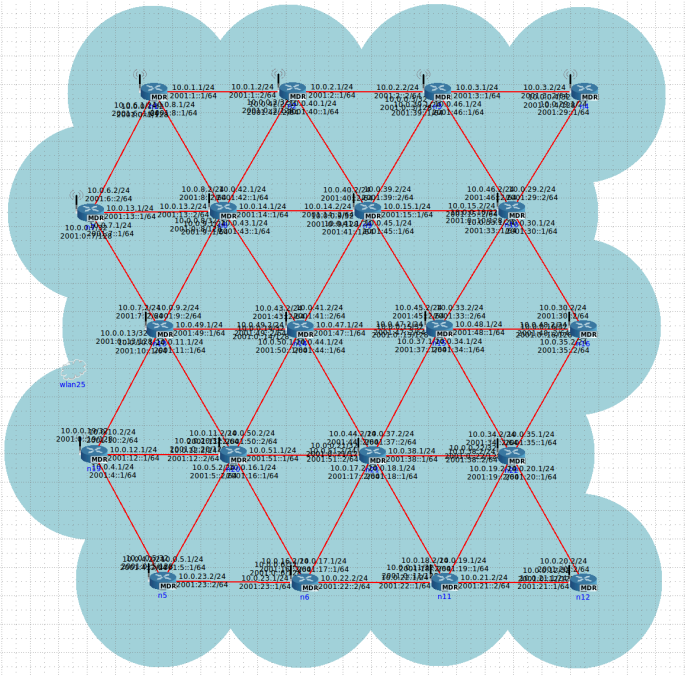

Fig. 4. Experimental Setup

for given unmanned vehicles. In this step, the program also generates the TS tasks and non-TS tasks in a given time period.

B. Environment Settings

We now introduce the evaluation environment settings in detail. To simulate the distributed IIoT environment, we leverage CORE. Developed by the U.S. Naval Research Laboratory, CORE is a network emulator [19]. The emulator allows users to leverage different network types, nodes, protocols, and structures that are defined inside of CORE to establish their own network. In addition, users are able to run lightweight virtual machines on network nodes. Based on the features of CORE, we first establish the APs, and then deploy the CRRS Scheduling Program on each AP, in order to evaluate the effectiveness of our proposed scheduling algorithm.

Access Points Configuration: We utilize the MANET Designated Routers (MDR) in CORE to simulate the APs. In our case, we deploy '20' MDRs in a $200 \times 200\,m^2$ area and assign the IP addresses of all the interfaces for each MDR, which is shown in Fig. 4. The location of the MDR is determined by Algorithm 1 discussed in Section III. We install

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2020.3044057, IEEE Internet of Things Journal

Python 3.7 to all MDRs and deploy the CRRS Scheduling Program to each MDR. In addition, we define an additional '100' MDRs as unmanned vehicles that are moving within this area. Then, we utilize the Unmanned Vehicle Routing Generator to randomly generate the routes and all tasks, which is shown in Fig. 5. According to the routes, we define the moving script for each unmanned vehicle.

Network Configuration: As we discussed in Section III, all unmanned vehicles communicate with the APs via wireless communication. Thus, we first assign all the APs and unmanned vehicles (MDRs) to a wireless WLAN and configure the OSPF protocol for each MDR. By doing this, the unmanned vehicles are able to communicate with the nearest AP. In our case, we set the communication radius as 20 m. Since the distance between each AP is larger than 20 m, we connect each AP by wired cable. Fig. 4 shows the network topology of the environment.

C. Experimental Design

By leveraging the described evaluation environment setup and configuration, we designed experiments to evaluate our CRRS scheme. In the following, we introduce the design of experiments in detail.

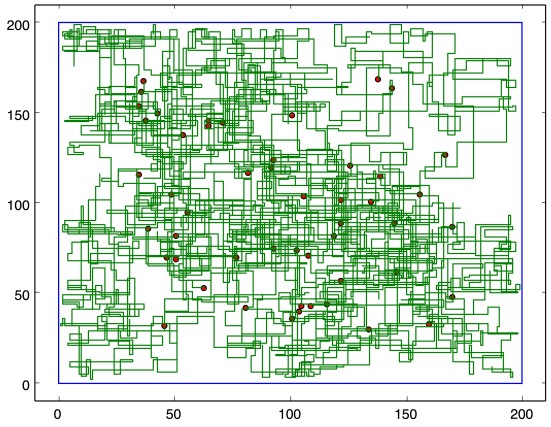

Fig. 5. Routes of Unmanned Vehicles

In our experiments, we assume the length of a TS task is 1 s. Based on the discussion in Section III, we utilize the *Poisson* distribution to generate TS tasks. In addition, we define the time duration of the experiments as 1200 s. First, we generate the routes of the unmanned vehicles and then the tasks. We record the data, including the coordinates of the unmanned vehicles, the start times of TS tasks, the start times of non-TS tasks, and the lengths of non-TS tasks. After that, according to the location data, we design the moving scripts for each unmanned vehicle. We deploy the CRRS Scheduling Program to the APs, the APs discover the unmanned vehicles within their communication range based on the location data. Then, we define the computing resources as two-thirds of the maximum number of concurrent tasks. In our case, there are '100' unmanned vehicles in the experiment, so that the maximum number of concurrent tasks is '100'. Further, we configure the total computing resources to be '100', and each AP has '5' computing resources. Finally, depending on the task information, which is generated by the Unmanned Vehicle Routing Generator, the CRRS Scheduling Program schedules the tasks.

V. Evaluation Results

We now detail the evaluation results of the experiments outlined in Section IV. In the following, we first present the evaluation methodology and then show a performance comparison of the proposed scheme and representative scheduling schemes.

A. Methodology

As we discussed in Section IV, we utilize the simulation program to create both TS and non-TS tasks. The program records the information of each task, including arrival time, task length, task type, completion time, and locations. Then, we deploy the tasks to the evaluation environment. The task scheduling algorithms on each edge computing node detect the nearby tasks and process them.

First, we need to identify a threshold for the CRRS scheme. Based on the threshold, the CRRS scheme then determines how many computing slots should be reserved. Based on the *Poisson* distribution, different numbers of TS tasks appear. For instance, the probability of 5 TS tasks appearing is 80 % and the probability of 9 TS tasks appearing is 60 %. When selecting 80 % as the threshold, our scheme reserves 5 computing slots. Obviously, setting a lower threshold yields better performance only for TS tasks, because our scheme reserves more computing resources. Nonetheless, setting a lower threshold causes computing resources to be wasted, because the idle computing resources cannot execute any tasks while waiting.

After identifying the probability threshold for this particular scenario, we compare the performance of the proposed scheduling scheme and the existing schemes, which are based on the first-in-first-out (FIFO) and priority-based scheduling algorithms [20]. Note that the priority-based scheduling scheme has limitations for handling the low priority tasks. Specifically, the low priority tasks get pushed to the end of the waiting list and never have a chance to be executed. Thus, we set a dynamic priority value for the non-TS tasks that increases the priority value as the waiting time increases. Finally, to evaluate the performance of each edge computing node, since this is a distributed edge computing platform, we compare the throughput of each edge node under both the proposed scheduling scheme and the representative schemes.

Based on the outlined scope and experimental design, we define the following metrics to evaluate the effectiveness of our proposed scheme: (i) *Turnaround Time:* It is an important metric for evaluating task scheduling schemes and defined as the time interval between one process being submitted to the execution queue and the process being completed by the CPU. It can be calculated by the sum of waiting time and execution time. In our experimentation, we collect the turnaround time and obtain the sum for all TS tasks to

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2020.3044057, IEEE Internet of Things Journal

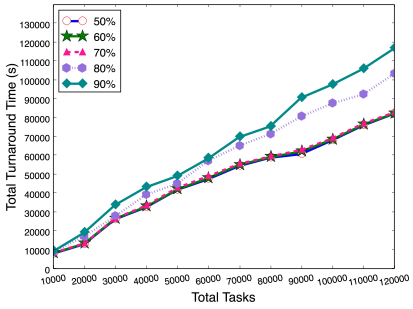

Fig. 6. Turnaround time vs. probability rate

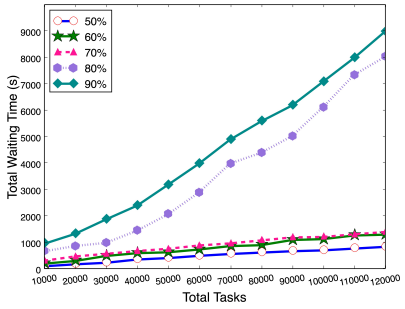

Fig. 7. Waiting time vs. probability rate

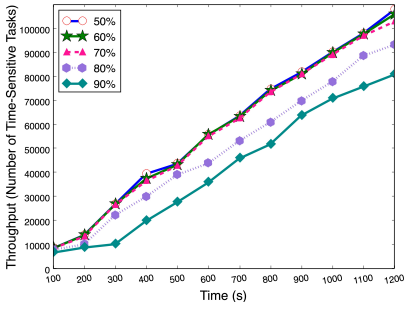

Fig. 8. Throughput vs. probability rate

evaluate the proposed scheme. (ii) *Waiting Time:* It is another important metric for evaluating task scheduling schemes. It refers to the time interval from the time when one process is submitted to the ready queue to the time when the process selected by CPU to execute. Based on the hardware limitations of available computing resources, it is difficult to improve execution speed. Thus, reducing the waiting time is the viable way to reduce the overall turnaround time. Just as with the turnaround time, we collect the waiting time for all TS tasks. (iii) *Throughput:* It measures how many tasks are completed in a given time period. It can clearly describe the performance of the scheduling scheme and, in the simulation, we compare the overall throughput of the edge computing system and the throughput of each edge computing node.

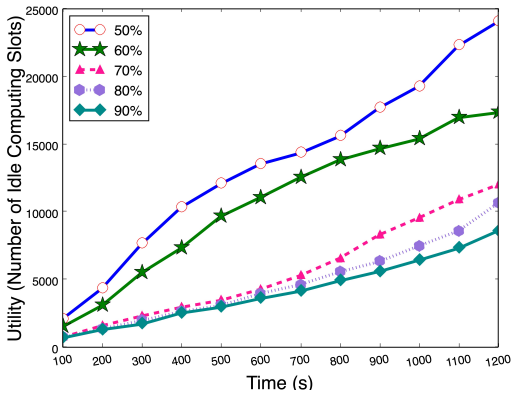

Fig. 9. Throughput on each edge computing node

B. Evaluation Results

We now present the evaluation results. First, we identify the threshold for the CRRS scheme. Fig. 6 shows the variation of turnaround time with threshold. Here, we set the probability threshold in 10 % increments, from 50 % to 90 %, in order to identify which probability has the best performance. The results show that setting the threshold to 50 % obtains the shortest turnaround time, while 90 % obtains the longest turnaround time. This is because setting a lower threshold causes the CRRS scheme to reserve more computing resources for the upcoming TS tasks. In addition, we set different thresholds in the evaluation. Fig. 6 shows that the turnaround time increases with the growth of the threshold.

Furthermore, we evaluate the waiting time and throughput in Figs. 7 and 8, respectively. Similar to turnaround time, we select the probability threshold from 50 % to 90 %, in 10 % increments. Then, we compare the performance. In detail, Fig. 7 shows the comparison of the waiting time for TS tasks with respect to different thresholds. As threshold increases, the waiting time for TS tasks increases as well. This is due to setting a high threshold causing the algorithm to reserve fewer computing resources. The reserved computing resources are not enough to execute all the upcoming TS tasks, leading to the increased waiting times for TS tasks. In our case, selecting 90 % as the threshold causes the waiting time for '120,000' tasks to be more than 9300 s. This approaches the waiting time of the FIFO algorithm.

In addition, Fig. 8 shows the variation of the throughput for TS tasks according with varying thresholds. As the threshold changes from 50 % to 70 %, the throughput performance drops 4.73 %. Moreover, continuously increasing the threshold to 90 % causes a throughput drop of 26.13 %. In summary, setting the threshold to 70 % is reasonable, as it has better overall performance and has less impact on non-TS tasks.

Finally, we obtain the utility of computing resources for different thresholds, as shown in Fig. 9, where the x-axis represents time, in seconds, and the y-axis represents the number of idle computing slots. The lower the number of empty computing slots, the higher the utility. From the figure, we can see that setting the threshold to 50 % can minimize the waiting time, whereas it maximizes idle computing slots that waste computing resources. Thus, based on the results shown in Figs. 6 and 9, we select 70 % as the threshold.

After obtaining the threshold for our algorithm, we utilize it to the proposed scheduling scheme and compare its performance with existing representative schemes. Fig. 10 shows the total turnaround time of all TS tasks between the proposed scheduling scheme and the existing schemes. From the figure, we can observe that the scheme based on FIFO algorithm achieves the largest turnaround time (more than 13,000 s) and the scheme based on priority-based algorithm is located in the middle (more than 9000 s). Our proposed scheduling scheme has the shortest turnaround time of 7792 s.

Fig. 11 shows the waiting time for TS tasks in the three

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2020.3044057, IEEE Internet of Things Journal

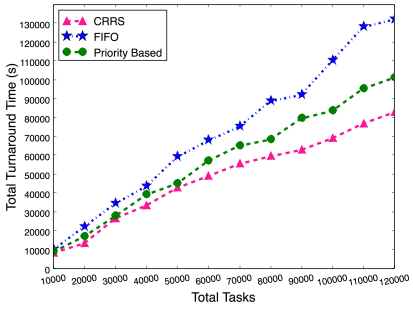

Fig. 10. Turnaround time comparison

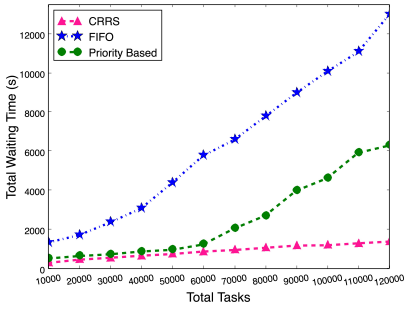

Fig. 11. Waiting time comparison

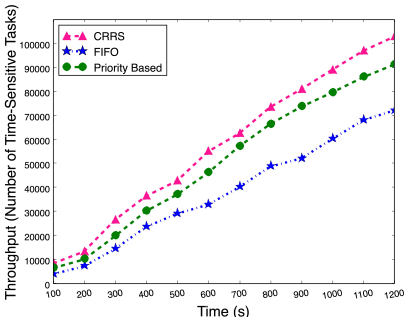

Fig. 12. Throughput comparison

scheduling schemes, where the x-axis represents the number of tasks and the y-axis indicates the waiting time. From the figure, we can see that, in the beginning, the waiting time of our proposed scheduling scheme is slightly shorter than the priority-based scheme. When the number of tasks increases, the waiting time for the priority based scheme increases very rapidly. This is because the priority value of non-TS tasks increases along with the waiting time to be able to execute all tasks. When the priority value of non-TS tasks is larger than TS tasks, the TS tasks will wait for the execution of the non-TS task. The scheme based on FIFO algorithm achieves the longest waiting time in the evaluation, as expected. In addition, we compare the throughput of the three selected scheduling scheme in Fig. 12. Clearly, our proposed scheduling scheme achieves the best throughput in all cases.

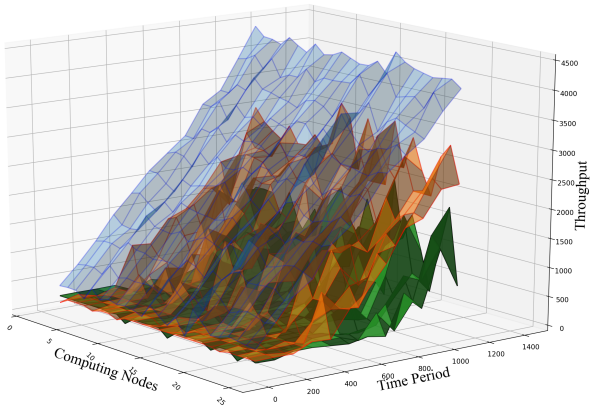

Fig. 13. Throughput comparison on each access point

Fig. 13 illustrates the throughput on each individual access point. Here, the x-axis represents the different computing nodes, the y-axis represents the time period, and the z-axis represents the throughput. Note that the blue surface on the top of the figure represents the throughput performance of our CRRS scheme. The yellow surface in the middle of the figure represents the throughput performance of the scheme based on priority-based algorithm. The green surface on the bottom of the figure represents the throughput performance of the scheme based on FIFO algorithm. The figure clearly indicates that the CRRS scheme could achieve the largest throughput of all other baseline schemes on each AP.

To summarize, our proposed CRRS scheduling scheme can improve the overall performance for TS tasks. In addition, in our proposed scenario, we deployed the proposed CRRS scheme in a distributed IIoT environment. The evaluation results show that our proposed scheme has the best performance in the distributed computing environment in comparison with the baseline scheduling schemes.

VI. Discussion

In this study, we proposed a computing resource reservation task scheduling scheme, which reserves computing resources for upcoming time-sensitive tasks. By doing so, it reduces the waiting time for TS tasks. As possible extensions of our work, we now consider some potential future directions toward improving performance in IIoT environments with respect to resource management, machine learning, and security.

Resource Management: We now discuss how to extend our scheme to manage resources in other IoT systems. On one hand, as the methodology of our proposed CRRS scheme is a generic one, it can be used in other IIoT systems to support fast response to time-sensitive applications, especially for some non-preemptive IIoT applications. As one example, the CRRS scheme can be used in smart manufacturing systems, in which some time-sensitive processes (e.g., real-time control, failure detection and recovery) require timely response and cannot be interrupted during manufacturing processes. In this case, the CRRS scheme can be used to improve the utilization of resources while reducing the response time of time-sensitive applications so that the strict performance requirements can be satisfied. As another example, our CRRS scheme can be used to assist in the smart grid system operations by reserving sufficient computing and networking resources for upcoming time-sensitive requests, which need timely process for urgent activities such as controlling critical power transmissions and dealing with failure detection and recovery, among others.

Second, besides the computing resource allocation, the networking resource allocation is another critical issue. The resources of networking is limited to support the growing demand for applications [21]. In this study, we proposed

a task scheduling algorithm for the TS tasks in IIoT. The proposed CRRS scheme focuses on reducing the turnaround time for TS tasks. Nonetheless, how to optimize networking resource allocation remains unsolved. It is necessary to design a network resource allocation algorithm to optimize network utilization. Furthermore, in distributed computing environments, because of the limitation of the computing capacity of distributed computing resources, one task may be assigned to different computing resources. Consider the complexity of the distributed computing structure, the cooperation and synchronization of different computing resources is another issue to consider. Further, how to deploy suitable computing and network resources to satisfy the demands of applications is critical [22]. This involves resource capacity and deployment allocation. Since IIoT is a dynamic environment, it brings new challenges for effective resource allocation and deployment in dynamic IIoT.

Machine Learning: In this study, we proposed a computing resource reservation task scheduling scheme. Obviously, improving prediction accuracy can improve the overall performance of the scheduling scheme. Moreover, since we did not have real IIoT data, we leveraged probability distributions in this study. Machine learning techniques can be applied to IoT systems to address different challenges [23], [24], [25]. For example, the recurrent neural network (RNN), as a typical machine learning model, can achieve accurate prediction in numerous cases. Based on real IIoT data, we could leverage and design an RNN model to predict upcoming TS tasks with better accuracy and precision [26]. Furthermore, as IIoT constantly generates new data, the RNN model needs to be retrained on the new data to maintain prediction accuracy. As the training cost is high, an online continuous learning strategy should be considered. Online continuous learning updates the learning model only utilizing a new data slice, which avoids retraining on the entire dataset, reducing the training overhead for the machine learning model [27]. In addition, IIoT is a distributed computing structure, and deploying the RNN model to the distributed computing nodes can reduce data transmission time and avoid network congestion. Thus, we shall design a distributed RNN model and leverage edge computing nodes to complete the RNN training, increasing the flexibility of the machine learning model in the distributed system.

Security and Privacy: Security and privacy is critical to IoT systems, given the massive amounts of investment invested into industrial processing, as well as the potential for disruption, destruction, and harm available through IIoT systems [28], [29], [30], [31], [32]. The specific environments and the distributed structure bring more security and privacy risks for IIoT. In particular, as mentioned above, in an IIoT system, an untrusted computing resource is a potential risk for the entire system. The automation of IIoT systems depends on data analysis. Adversaries could launch attacks and compromise computing resources and interfere with data to disrupt analytical results and infer sensitive information. Since the IIoT system is controlled and managed based on data analysis, incorrect analytical results could pose serious system failures. Furthermore, adversaries could insert fake or duplicated data to affect response efficiency for IIoT tasks. There are numerous aspects to protecting computing resources (e.g., filtering suspicious tasks, selecting trusted computing resources). From the computing resource perspective, when receiving a task from a client, the trust of the client should be evaluated, avoiding untrusted clients so that the computing resources can be protected. From the client perspective, trust-based computing resource allocation strategies should protect clients. If clients send their data to a compromised computing infrastructure, it can obviously disclose the client's sensitive information. Thus, it necessary to have an evaluation strategy to inspect the computing resources before sending computing tasks.

VII. Related Works

We now review some existing studies closely to IIoT and task scheduling that are much relevant to our study.

Based on IoT devices that are deployed in industrial environments, industrial systems are able to collect large amounts of data. Based on the collected datasets, operators are able to analyze the generated data and make assessments to manage the industrial systems effectively. There are numerous issues related to the deployment of IoT devices, resource allocation, and task scheduling, among others, which must be resolved. To deal with these issues, an emulation platform is necessary to enable evaluation without the potential to manipulate a real system and cause some unexpected results. Some existing research efforts have focused on developing emulation platforms for IIoT systems. For example, Boschert *et al.* investigated a digital model called digital twin (i.e., a digital model of the real industrial system) [33], which can abstract the physical industrial system to a digital model. By leveraging digital twins, operators can easily emulate the different statuses of the system to identify resource allocation solutions for the specific IIoT system under investigation. Likewise, Zhang *et al.* [34] proposed a digital twin based real-time scheduling scheme to handle the specific case of the hollow glass production line. Specifically, they utilized a digital model to emulate the production process and optimize the scheduling algorithm.

Since a large number of industrial processes are TS, cloud computing cannot satisfy the fast response needed. Edge computing has been considered as a viable computing infrastructure to handle TS tasks in IIoT systems [14], [15], [35], [36]. Nonetheless, as edge computing is a distributed computing infrastructure, how to select proper computing resources to accomplish tasks remains an unsolved issue. To this end, Tran *et al.* [37] proposed a heuristic algorithm to optimize task offloading, in order to obtain the maximum use of edge computing resources. Broji *et al.* [38] proposed a QoS-based resource allocation strategy to assist IoT devices in selecting the best computing resource. Likewise, Chen *et al.* [39] proposed an efficient task offload scheme in the mobile edge computing environment. Specifically, the task offloading was formulated as a mixed-integer non-linear program and the non-linear program was separated into two sub-problems in order to solve it.

Related to task scheduling in IIoT, existing research has focused on optimized scheduling algorithms to avoid network

congestion and improve computing performance. For instance, Pham *et al.* [40] considered task scheduling in a cloud-fog computing system and proposed a heuristic-based algorithm to fully utilize the cloud and fog computing resources. Similarly, Basu *et al.* [41] proposed a hybrid algorithm GAACO, combining the Genetic Algorithm (GA) and Ant Colony Optimization (ACO). The GAACO algorithm balances the computing payloads for each edge computing resource. Furthermore, as a cognitive or intelligent model, GAACO evolves itself based on the historical data, leading to a better computing payload balancing solution. Likewise, Turjman *et al.* [42] proposed a fully informed particle swarm optimization (FIPS) algorithm based on robust canonical particle swarm optimization (CPSO). In their study, the throughput and delay were optimized based on the different data traffic categories.

In this paper, we mainly focused on the task scheduling problem in IIoT environments. To be specific, we first classified the tasks and defined the problem space. We then focused on the reduction of the response time of TS tasks in IIoT, since those tasks have more critical response time requirements than others. We also designed a representative IIoT scenario and system model. Based on the model, we proposed our CRRS scheme and carried out extensive experiments to evaluate the effectiveness of our scheme in comparison with some existing schemes.

VIII. Final Remarks

In this paper, we identified the problem space for IIoT and focused on optimizing the computing performance for time sensitive (TS) tasks in a typical IIoT environment. To achieve this goal, we first defined a representative IIoT scenario. Based on the proposed scenario, we designed the system model and proposed Computing Resource Reservation Scheduling (CRRS) scheme for task scheduling. In detail, based on the probability distribution of TS tasks, our proposed scheme can reserve computing slots for upcoming TS tasks. In this way, the TS tasks can be executed immediately without any waiting time. Our scheme is capable of reducing the turnaround time for TS tasks in a computing capacity constant IIoT system. To evaluate the proposed scheme, we designed a comprehensive evaluation environment on CORE, a typical emulation environment. We also implemented a set of programs to simulate the process of the tasks generation of unmanned vehicles and utilized CORE to establish the network environment. Finally, we deployed the proposed scheme to CORE and conducted an evaluation of the proposed CRRS scheme against some representative scheduling schemes. Our extensive experimental results indicate that our proposed scheduling scheme can reduce the overall turnaround time for TS tasks in the IIoT environment, comparing with existing scheduling schemes.

References

[1] H. Xu, W. Yu, D. Griffith, and N. Golmie, "A survey on industrial internet of things: A cyber-physical systems perspective," *IEEE Access*, vol. 6, pp. 78 238–78 259, 2018.

[2] H. Xu, X. Liu, W. Yu, D. Griffith, and N. Golmie, "Reinforcement learning-based control and networking co-design for industrial internet of things," *IEEE Journal on Selected Areas in Communications*, pp. 1–1, 2020.

[3] F. Liang, W. Yu, X. Liu, D. Griffith, and N. Golmie, "Toward edge-based deep learning in industrial internet of things," *IEEE Internet of Things Journal*, vol. 7, no. 5, pp. 4329–4341, 2020.

[4] Z. Cai and T. Shi, "Distributed query processing in the edge assisted iot data monitoring system," *IEEE Internet of Things Journal*, pp. 1–1, 2020.

[5] Q. Chen, X. Ma, S. Tang, J. Guo, Q. Yang, and S. Fu, "F-cooper: feature based cooperative perception for autonomous vehicle edge computing system using 3d point clouds," in *Proceedings of the 4th ACM/IEEE Symposium on Edge Computing*, 2019, pp. 88–100.

[6] R. Rajkumar, C. Lee, J. P. Lehoczky, and D. P. Siewiorek, "Practical solutions for qos-based resource allocation problems," in *Proceedings 19th IEEE Real-Time Systems Symposium (Cat. No. 98CB36279)*. IEEE, 1998, pp. 296–306.

[7] T. Kuo, J. Chen, Y. Chang, and P. Hsiu, "Real-time computing and the evolution of embedded system designs," in *2018 IEEE Real-Time Systems Symposium (RTSS)*, 2018, pp. 1–12.

[8] T. Amudha and T. Dhivyaprabha, "Qos priority based scheduling algorithm and proposed framework for task scheduling in a grid environment," in *2011 International Conference on Recent Trends in Information Technology (ICRTIT)*. IEEE, 2011, pp. 650–655.

[9] M. Ghobakhloo, "The future of manufacturing industry: A strategic roadmap toward industry 4.0," *Journal of Manufacturing Technology Management*, vol. 29, no. 6, pp. 910–936, 2018.

[10] S. Hu and G. Li, "Dynamic request scheduling optimization in mobile edge computing for iot applications," *IEEE Internet of Things Journal*, vol. 7, no. 2, pp. 1426–1437, 2020.

[11] B. Cheng, G. Solmaz, F. Cirillo, E. Kovacs, K. Terasawa, and A. Kitazawa, "Fogflow: Easy programming of iot services over cloud and edges for smart cities," *IEEE Internet of Things Journal*, vol. 5, no. 2, pp. 696–707, 2018.

[12] Q. Fan and N. Ansari, "Application aware workload allocation for edge computing-based iot," *IEEE Internet of Things Journal*, vol. 5, no. 3, pp. 2146–2153, 2018.

[13] F. Zezulka, P. Marcon, Z. Bradac, J. Arm, T. Benesl, and I. Vesely, "Communication systems for industry 4.0 and the iiot," *IFAC-PapersOnLine*, vol. 51, no. 6, pp. 150–155, 2018.

[14] W. Yu, F. Liang, X. He, W. G. Hatcher, C. Lu, J. Lin, and X. Yang, "A survey on the edge computing for the internet of things," *IEEE access*, vol. 6, pp. 6900–6919, 2018.

[15] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," *IEEE Internet of Things Journal*, vol. 3, no. 5, pp. 637–646, 2016.

[16] B. Omoniwa, R. Hussain, M. A. Javed, S. H. Bouk, and S. A. Malik, "Fog/edge computing-based iot (feciot): Architecture, applications, and research issues," *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 4118–4149, 2019.

[17] X. Liu, J. Cao, Y. Yang, and S. Jiang, "Cps-based smart warehouse for industry 4.0: a survey of the underlying technologies," *Computers*, vol. 7, no. 1, p. 13, 2018.

[18] CORE, https://www.nrl.navy.mil/itd/ncs/products/core.

[19] W. Gao, J. H. Nguyen, W. Yu, C. Lu, D. T. Ku, and W. G. Hatcher, "Toward emulation-based performance assessment of constrained application protocol in dynamic networks," *IEEE Internet of Things Journal*, vol. 4, no. 5, pp. 1597–1610, Oct 2017.

[20] P. B. G. Avi Silberschatz and G. Gagne, *Operating System Concepts Ninth Edition*, 2012.

[21] R. Kumar, M. Hasan, S. Padhy, K. Evchenko, L. Piramanayagam, S. Mohan, and R. B. Bobba, "End-to-end network delay guarantees for real-time systems using sdn," in *2017 IEEE Real-Time Systems Symposium (RTSS)*, 2017, pp. 231–242.

[22] T. Amert, N. Otterness, M. Yang, J. H. Anderson, and F. D. Smith, "Gpu scheduling on the nvidia tx2: Hidden details revealed," in *2017 IEEE Real-Time Systems Symposium (RTSS)*, 2017, pp. 104–115.

[23] W. G. Hatcher and W. Yu, "A survey of deep learning: Platforms, applications and emerging research trends," *IEEE Access*, vol. 6, pp. 24 411–24 432, 2018.

[24] F. Liang, W. G. Hatcher, W. Liao, W. Gao, and W. Yu, "Machine learning for security and the internet of things: the good, the bad, and the ugly," *IEEE Access*, vol. 7, pp. 158 126–158 147, 2019.

[25] D. Wu, H. Shi, H. Wang, R. Wang, and H. Fang, "A feature-based learning system for internet of things applications," *IEEE Internet of Things Journal*, vol. 6, no. 2, pp. 1928–1937, 2018.

[26] R. Han, F. Zhang, L. Y. Chen, and J. Zhan, "Work-in-progress: Maximizing model accuracy in real-time and iterative machine learning," in *2017 IEEE Real-Time Systems Symposium (RTSS)*, 2017, pp. 351–353.

[27] F. Liang, W. G. Hatcher, G. Xu, J. Nguyen, W. Liao, and W. Yu, "Towards online deep learning-based energy forecasting," in *2019 28th International Conference on Computer Communication and Networks (ICCCN)*. IEEE, 2019, pp. 1–9.

[28] X. Zheng and Z. Cai, "Privacy-preserved data sharing towards multiple parties in industrial iots," *IEEE Journal on Selected Areas in Communications*, vol. 38, no. 5, pp. 968–979, 2020.

[29] H. Xu, W. Yu, X. Liu, D. Griffith, and N. Golmie, "On data integrity attacks against industrial internet of things," in *2020 IEEE Intl Conf on Dependable, Autonomic and Secure Computing, Intl Conf on Pervasive Intelligence and Computing, Intl Conf on Cloud and Big Data Computing, Intl Conf on Cyber Science and Technology Congress (DASC/PiCom/CBDCom/CyberSciTech)*, 2020, pp. 21–28.

[30] W. Yu, D. Griffith, L. Ge, S. Bhattarai, and N. Golmie, "An integrated detection system against false data injection attacks in the smart grid," *Security and Communication Networks*, vol. 8, no. 2, pp. 91–109, 2015.

[31] X. Liu, C. Qian, W. G. Hatcher, H. Xu, W. Liao, and W. Yu, "Secure internet of things (iot)-based smart-world critical infrastructures: Survey, case study and research opportunities," *IEEE Access*, vol. 7, pp. 79 523–79 544, 2019.

[32] Y. Song, C. Ma, X. Wu, L. Gong, L. Bao, W. Zuo, C. Shen, R. W. Lau, and M.-H. Yang, "Vital: Visual tracking via adversarial learning," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 8990–8999.

[33] S. Boschert and R. Rosen, "Digital twinthe simulation aspect," in *Mechatronic Futures*. Springer, 2016, pp. 59–74.

[34] H. Zhang, Q. Liu, X. Chen, D. Zhang, and J. Leng, "A digital twin-based approach for designing and multi-objective optimization of hollow glass production line," *IEEE Access*, vol. 5, pp. 26 901–26 911, 2017.

[35] K.-D. Thoben, S. Wiesner, and T. Wuest, "industrie 4.0 and smart manufacturing-a review of research issues and application examples," *International Journal of Automation Technology*, vol. 11, no. 1, pp. 4–16, 2017.

[36] M. Papazoglou, W.-J. van den Heuvel, and J. Mascolo, "Reference architecture and knowledge-based structures for smart manufacturing networks," *IEEE Software*, 2015.

[37] T. X. Tran and D. Pompili, "Joint task offloading and resource allocation for multi-server mobile-edge computing networks," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 1, pp. 856–868, 2018.

[38] A. Brogi and S. Forti, "Qos-aware deployment of iot applications through the fog," *IEEE Internet of Things Journal*, vol. 4, no. 5, pp. 1185–1192, 2017.

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

[40] X.-Q. Pham and E.-N. Huh, "Towards task scheduling in a cloud-fog computing system," in *2016 18th Asia-Pacific network operations and management symposium (APNOMS)*. IEEE, 2016, pp. 1–4.

[41] S. Basu, M. Karuppiah, K. Selvakumar, K.-C. Li, S. H. Islam, M. M. Hassan, and M. Z. A. Bhuiyan, "An intelligent/cognitive model of task scheduling for iot applications in cloud computing environment," *Future Generation Computer Systems*, vol. 88, pp. 254–261, 2018.

[42] F. Al-Turjman, M. Z. Hasan, and H. Al-Rizzo, "Task scheduling in cloud-based survivability applications using swarm optimization in iot," *Transactions on Emerging Telecommunications Technologies*, p. e3539, 2018.