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/TII.2018.2843802, IEEE Transactions on Industrial Informatics

Industrial IoT Data Scheduling based on Hierarchical Fog Computing: A key for Enabling Smart Factory

Djabir Abdeldjalil Chekired, *Student Member, IEEE,* Lyes Khoukhi, *Member, IEEE,* and Hussein T. Mouftah*, Fellow, IEEE*

Abstract—Industry 4.0 or Industrial Internet of Things (IIoT), has become one of the most talked-about industrial business concepts in recent years. Thus, to efficiently integrate Internet of Things (IoT) technology into industry, the collected and sensed data from IIoT need to be scheduled in real time constraints; especially for big factories. To this end, we propose in this paper a hierarchical fog servers' deployment at the network service layer across different tiers. Using probabilistic analysis models, we prove the efficiency of the proposed hierarchical fog computing compared with the flat architecture. In this work, IIoT data and requests are divided into both high priority and low priority requests; the high priority requests are urgent/emergency demands that need to be scheduled rapidly. Therefore, we use two-priority queuing model in order to schedule and analyze IIoT data. Finally, we further introduce a workload assignment algorithm to offload peak loads over higher tiers of the fog hierarchy. Using realistic industrial data from Bosch group, the benefits of the proposed architecture compared to the conventional flat design are proved using various performance metrics and through extensive simulations.

Index Terms— Industrial IoT, hierarchical fog architecture, priority queuing, linear optimization.

NOMENCLATURE

w_i	Received IIoT workload by a i-th tier-1 server
c_i	Computational capacity of i-th tier-1 server
s	Number of tier-1 servers
Cap	Computational capacity of tier-1 servers
O_i	Offloaded amount of one i-th tier-1 server
$\mathbb{P}(w_i \leq Cap)$	Probability that the i-th tier-1 server can successfully serve its received workload
$\mathbb{F}_{O_i}(Cap)$	Cumulative Distribution Function (CDF) of the offloaded workloads
ω	Total workload that the tier-2 can handle
$\mathcal{S}_1$	Set of servers for tier-1 (from 1 to $s-1$)
$\mathcal{S}_2$	Set of servers for tier-1 (only one server)
C	Computational capacity of tier-2 server
Ω	Offloaded amount for one server at tier-2
$\mathbb{P}_{\mathcal{S}_2}(Cap)$	Probability that the tier-2 server can serve its offloaded workload Ω
λ_1 and μ_1	Arrival rate and the service rate of demands with high priority
λ_0 and μ_0	Arrival rate and the service rate of demands with low priority
r	Number of waiting requests in each queue
π_0	Total probability condition
$E(W_0))$	The mean waiting time for low priority
$E(W_1))$	The mean waiting time for high priority
$\{c_1, c_i, \dots, c_x\}$	Amounts of computation capacities
$\{G_1, G_2, \dots, G_K\}$	K mutually groups of offloaded workload
$\{c_1, c_2, \dots, c_S\}$	Computation capacities of S servers in the Fog layer (CPU cycles per second)
$\{D_1, D_2, \dots, D_x\}$	Data size of each task
ψ	The server where task i is placed on
θ_i^*	The optimal solution to the MNIP problem
set_j	Set of computing tasks placed at server j
B_ψ	The network bandwidth allocated to ψ
m	Number of computing tasks for workload assignment problem.
y	Number of tasks for workload assignment Sub-problem
Υ_y	the lower bound on the total delays for the remaining m-y tasks

D. A. Chekired and L. Khoukhi are with the Autonomic Networking Environment (ERA), Charles Delaunay Institute (ICD), University of Technology of Troyes, France. (e-mail:djabir_abdeldjalil.chekired@utt.fr, lyes.khoukhi@utt.fr).

H. T. Mouftah is with the School of Electrical Engineering and Computer Science, University of Ottawa, Ontario, Canada. (e-mail: mouftah@uottawa.ca).

I. INTRODUCTION

THE Internet of Things (IoT) has been gaining attraction in many industry areas, such as logistics, manufacturing, retailing, and pharmaceutics [1]. On the other hand, with the advances in wireless communication and sensor network technologies, more and more networked things or smart objects are being involved in industrial IoT (IIoT). As a result, IIoT-related technologies have also made a large impact on information and communication technology (ICT) and industrial systems technologies [2].

Recently, the manufacturing industry has gained another transformation sparked by smart connectivity and advanced analytics. This is referred to as advanced manufacturing in North America and Industry 4.0 in Europe [3]. Because of its highly adapted properties and reduced delivery time, this smart connectivity and analytics based on IIoT are seen as key enablers to future industry. On the one hand, this transformation facilitates flexibility and scalability, particularly with respect to the high generation of data and information. On the other hand,

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/TII.2018.2843802, IEEE Transactions on Industrial Informatics

it also permits the delivery of products with improved quality at the same or lower cost by leveraging the data collected by various elements of a connected assembly line.

Consequently, a large amount of data is generated from IIoT devices installed over different physical factory components as assembly lines, human machine interface, and logistic structures. Besides, the data collected brings transparency about the machines' operations, the materials utilized, the facility logistics, and even about the human operators. However, much of the data collected are used for direct real-time feedback control and for forensic purposes. As presented in a case study from Bosch group in [3], during the manufacturing process, information about the status of the process, the states of machines, tools, and parts produced are uninterruptedly relayed, scheduled and stored in a real-time manner. The capacity, scale, and regularity of production in this traditional case study (i.e., Bosch group case) is so high that it requires the use of a big data tools scheduling to store, reprocess, and join the data. In these regards, an efficient service management for IIoT needs to be implemented, and the traditional industrial network should be enhanced and adapted in order to ensure real time analytic process, to minimize the end-to-end delay and to increase the data storage capacity.

In recent research works, cloud computing has acted as a key approach for big data processing with its ample computing, storage capacity and distributed aspects [4], [5]. However, most interaction between IIoT devices and back-end servers in the traditional system is done through large scale cloud data centers. Nevertheless, because of the delays induced on Wide Area Networks (WANs), and being far from IIoT devices, cloud-based IIoT systems face several challenges, including especially a high response time due to the high cost of communication bandwidth, and heavy loads on cloud servers due to the high redundancy of data.

In order to overcome the weaknesses of cloud computing, a local cloud computing architecture, called fog computing, has been introduced recently. According to Cisco [6], fog computing extends the cloud computing away from the cloud computing data centers and towards the edge of the network. Fog computing, considered as a new decentralized computing aspect of cloud architecture, is one of challenging industry topics for IoT. Fog computing is a newly introduced concept that aims to put the cloud services closer to the end users (things) for better quality of service (QoS). Fog is an intelligent server's layer sitting between cloud and IIoT that brings low latency, location awareness, and widespread geographical distribution for IoT [7]. Getting main concepts of cloud computing, fog provides computation and storage capacities to end users, but at the edge of the network. An open issue consists of exploring the enhancement of an optimal end-to-end delay network by having an intelligent layer of fog servers between IIoT devices and cloud data centers.

Our contributions are as follows:

1) We design a decentralized multi-tiers fog architecture for IIoT requests scheduling and data analytics. The architecture is based on hierarchical deployment of servers in the fog layer.
2) We propose a probabilistic model to compare the efficiency of fog resources utilization between flat and hierarchical fog architectures. We show, using an analytical model that the hierarchical fog architecture has better efficiency requests handling in terms of minimizing computation and communication delays.
3) We implement a priority queuing model for scheduling IIoT data. We divide IIoT requests into two priority levels (high and low); we favorite high priority requests (e.g., emergency requests) to be scheduled first.
4) We formulate the IIoT workload assignment problem as a mixed nonlinear integer programming (MNIP) problem, and propose a branch and bound approach to solve the problem. Then, the optimal results will be aggregated for high tiers using the Simulated Annealing Algorithm in order to ensure a global optimality of IIoT workload assignment.
5) We develop an IIoT offloading algorithm (IIoT_OFF) which offloads the workloads over different tiers of the hierarchy.
6) Finally, we compare our proposed model with relevant and recent works in the performance evaluation Section.

The remainder of this paper is organized as follows. In Section II, we briefly present the related works. Section III presents a probabilistic model for hierarchical fog servers. Section IV describes the priority queuing model for IIoT data scheduling. In Section V, we introduce the IIoT data assignment and offloading algorithms. The evaluation of the proposed architecture and algorithms is discussed in Section VI. Finally, Section VII concludes the paper.

II. RELATED WORKS

According to Xu et *al*. [8], the application of IoT in the industrial automation address to a very small extent. IoT solutions for the industrial automation are still evolving. As explained earlier, the fog-based approaches can meet the requirements of modern industrial systems. However, most of the existing works focus on centralized computation architectures that utilize cloud computing for the purpose of monitoring data and managing control processes in the industrial automation [7], [9]. The majority of existing approaches and solutions in the context of cloud computing in the industrial automation focus on the higher levels than the field-level. In [7], the authors investigate a prototype to explore the use of IoT devices to communicate with a cloud-based controller targeting the automation industry. Nevertheless, this work applies mitigation mechanisms to deal only with the communication delays that are caused by the networks and ignored the computation delays and capacities of cloud servers; also the mitigation model was not validated using a concrete mathematical model.

In a recent work [9], the authors derive an exact expression for the performance of IIoT by combining computation with intelligence. They design an efficient way to obtain a threshold by approximating the performance of different computing manners, and show how to apply it to practical IIoT applications. The proposed work is interesting and gives good results (i.e., we will compare the computing efficiency of our work with this work [9] in the performance evaluation section). However, the authors ignore data transmission delay and synchronization time especially when using distributed computing manner.

Only few works (e.g., [10], [11]) consider the distributed computation feature. In [10], the authors proposed a computing

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/TII.2018.2843802, IEEE Transactions on Industrial Informatics

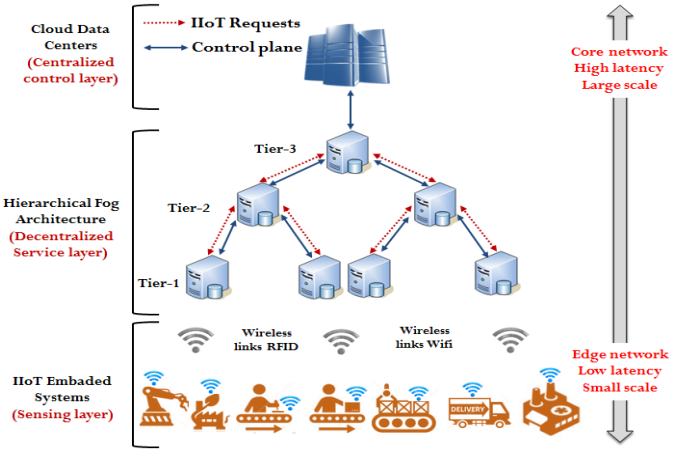


Fig. 1. Hierarchical fog architecture for Industrial internet of things

paradigm, named Edge Mesh, which distributes the decision-making tasks among edge devices within the network instead of sending all the data to a centralized server. Edge Mesh provides many benefits, including distributed processing, low latency and fault tolerance. However, the authors ignore the tasks allocation problem in such distributed architecture. A vision of distributed IIoT data processing using fog and cloud computing has also been discussed in a recent work [11]. The authors discuss the emerging challenges in the aspects of data processing in IIoT. They design a flexible framework by integrating the fog computing and cloud computing. Based on the time latency requirements, the collected data are processed and stored by the edge server or the cloud server.
We will compare our results with the results of these two (i.e., [09] and [11]) in the performance evaluation section.

In this paper, we address the aforementioned issues towards minimizing communication and data processing delay in industrial internet of things systems. Hence, as shown in Fig. 1, the fog servers are organized into hierarchical architecture, in order to serve industrial things requests and analyses information in a real time way. Our idea is to optimally aggregate and offload the IIoT peak loads that exceed the capacities of lower tiers of fog servers to other servers at higher tiers. In addition, we aim to allocate provisioned capacities at high tiers servers in order to schedule the offloaded workloads. Consequently, our solution will be able to handle large amounts of IIoT devices requests and data coming from different physical factory components. To the best of our knowledge, our work is one of the first attempts to design a new hierarchical fog servers' architecture for IIoT environment, and further improve its performance by minimizing the processing and communication delays.

III. PROBABILISTIC ANALYSIS MODEL OF FOG HIERARCHY

In this section, we formally prove the advantage of the hierarchical fog computing on improving the efficiency of cloud resource utilization when serving the peak load of IIoT. Without loss of generality, we first develop an analytical model for a two tier hierarchical fog. Then, the analysis results based

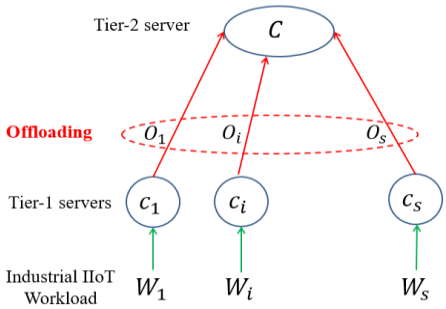


Fig. 2. Notation in two-tier fog hierarchy for IIoT workloads

on this two tier model are generalized to more complicated topology of the fog hierarchy.

As shown in Fig. 2, a two tier fog hierarchy consists of s servers in tier-1, and one server in tier-2. Let c_i denotes the computational capacity of the i-th tier-1 servers, and w_i be the amount of received IIoT workload of the i-th tier-1 servers; c_i and w_i are measured in units of CPU cycles. However, in tier-2 we have only one server which has a capacity of C.

A. Probabilistic model of tier-1 fog hierarchy

We assume that $w_1, \dots, w_i, \dots, w_s$ are independent and identically distributed random variables. We denote by $\mathbb{P}(w_i \leq Cap)$ the probability that the i-th tier-1 server can successfully serve its received workload w_i when Cap is its computational capacity. The probability that all tier-1 servers can schedule their received requests from different IIoT devices and sensors is:

$$\mathbb{P}(w_1 \leq c_1,\ w_2 \leq c_2, \dots, w_s \leq c_s) = \prod_{j=1}^{s} \mathbb{P}(w_i \leq c_i) \quad (1)$$

According to the capacity of each server, we consider two cases as follows:

1) If $w_i > c_i$, the workload amount denoted by O_i will be offloaded on the tier-2 server (see Fig. 2). The offloaded amount O_i of one server i is given by:

$$O_i = w_i - c_i \quad (2)$$

2) If $w_i \leq c_i$, the offloaded workload is null (i.e. $O_i = 0$).

As results, we define the Cumulative Distribution Function (CDF) $\mathbb{F}$ of the offloaded workloads as follows:

$$\mathbb{F}_{O_i}(Cap) = \mathbb{P}(O_i \leq Cap) = \begin{cases} \mathbb{P}(w_i \leq Cap + c_i) & if\ Cap \geq 0 \\ 0 & otherwise \end{cases} \quad (3)$$

B. Probabilistic model of tier-2 fog hierarchy

For now, we calculate the probabilistic model for tier-2 server, systematically, the probabilistic model of tier-2 server is related to tier-1 servers. As shown in Fig. 2, we define the total workload ω that the tier-2 server has revived from tier-1 servers as follows:

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/TII.2018.2843802, IEEE Transactions on Industrial Informatics

$$\omega = \sum_{i=1}^{s} O_i \quad (4)$$

To calculate the probability that one server in tier-2 can serve the received data from different tier-1 servers, we divide the s tier-1 servers into two groups $\mathcal{S}_1$ and $\mathcal{S}_2$ where the group $\mathcal{S}_1$includes servers from 1 to $s-1$, and $\mathcal{S}_2$ has server s (i.e., only one server).

1) We suppose that there is no workload offloaded by $\mathcal{S}_2$, so all the workloads come from $\mathcal{S}_1$, then the probability that tier-2 servers can handle their received workload ω is:

$$\mathbb{P}_{\mathcal{S}_1}(\omega) = \mathbb{P}(\sum_{i=1}^{s-1} O_i \le Cap)\, \mathbb{P}(O_s = 0) \quad (5)$$

$$\text{Where, } \mathbb{P}(O_s = 0) = \mathbb{F}_{s-1}(Cap)\, \mathcal{S}_s(0) \quad (6)$$

2) Now, we suppose that there is a workload of amount Ω ($\Omega > 0$) offloaded by $\mathcal{S}_2$. The workloads offloaded by $\mathcal{S}_1$ cannot exceed $(C - \Omega)$, because C is the computational capacity of the tier-2 server. When Ω is fixed, the probability that the tier-2 server can serve its offloaded workload Ω is given as follows:

$$\mathbb{P}_{\mathcal{S}_2}(C) = \mathbb{P}(\sum_{i=1}^{s-1} O_i \le C - \Omega)\, \mathbb{P}(O_s = \Omega) \quad (7)$$

Then, when $0 < \Omega < C$, this probability is:

$$\mathbb{P}_{\mathcal{S}_2}(C) = \int_{0^+}^{C} \mathbb{P}(\sum_{i=1}^{s-1} O_i \le C - \Omega) d\, \mathbb{P}(O_s \le \Omega) \quad (8)$$

From (5), (6), (7) and (8), we define the formula of the Cumulative Distribution Function $\mathbb{F}$ for tier-2 workload probabilistic model as follows:

Let $\mathbb{F}_s(C) = \mathbb{P}(\omega \le C)$, and $\mathcal{S}_s(C) = \mathbb{P}(O_S \le C)$, so we have:

$$\mathbb{F}_s(C) = \mathbb{F}_{S-1}(C)\, \mathcal{S}_s(0) + \int_{0^+}^{C} \mathbb{F}_{S-1}(C - \Omega)\, d\mathcal{S}_s(\Omega) \quad (9)$$

$$\text{Thus, } \mathbb{F}_s(C) = \mathbb{P}_{\mathcal{S}_1}(C) + \mathbb{P}_{\mathcal{S}_2}(C) \quad (10)$$

Eq. (9) and Eq. (10) show how to analytically derive the characteristics of ω, given the knowledge about workloads received by all tier-1 servers.

C. Analytic comparison model

From the above workload models, we further analyze the advantage of hierarchical fog on improving the efficiency of servers' resource utilization. As shown in Figure 2, the amount of capacity C is provisioned to the tier-2 server. Similarly, when we provision the flat fog architecture using the same amount of capacity, C will be provisioned to tier-1 servers. Moreover, we define the difference between flat (i.e., one tier) and hierarchical (i.e., multi-tier) architectures of fog servers' deployment.

To analyze the difference between flat and hierarchical designs of fog, we first have the following lemma:

Theorem 1: For any computation capacities $c_1^*, c_2^* \in [0, C], c_1^* + c_2^* = C$, we have:

$$\mathbb{P}(O_1 + O_2 \le C) \ge \mathbb{P}(O_1 \le c_1^*)\, \mathbb{P}(O_2 \le c_2^*) \quad (11)$$

Proof: We use algebra of sets to proof this lemma. We define two sets of offloading workloads as follows:

$$\begin{cases} \mathbb{S}_1 = \{(O_1, O_2) | O_1 + O_2 \le C\} \\ \mathbb{S}_2 = \{(O_1, O_2) | O_1 \le c_1^*, O_2 \le c_2^*\} \end{cases}$$

It is evident that for any element $\{\Omega_1, \Omega_2\} \in \mathbb{S}_2$, we have also $\{\Omega_1, \Omega_2\} \in \mathbb{S}_1$, so $\mathbb{S}_2 \subseteq \mathbb{S}_1$, and then we have:

$$\mathbb{P}(O_1 + O_2 \le C) \ge \mathbb{P}(O_1 \le c_1^*)\, \mathbb{P}(O_2 \le c_2^*) \quad (12)$$

Now, we generalize the concept for all fog servers. To this, end, we extend Theorem 1 to lemma 1 as follows:

Lemma 1: For any $c_i^* \in [0, C]$ $(\forall\, i = 1,2, \dots, s)$ and $\sum_{i=1}^{s} c_i^* = C$ we have:

$$\mathbb{P}\left(\sum_{j=1}^{s} O_i \le C\right) \ge \prod_{i=1}^{s} \mathbb{P}(O_i \le c_i^*) \quad (13)$$

Proof: Lemma 1 can be proved recursively using Theorem 1:

$$\begin{aligned} \mathbb{P}\left(\sum_{i=1}^{s} O_i \le C\right) &\ge \mathbb{P}\left(\sum_{i=1}^{s-1} O_i \le \sum_{i=1}^{s-1} c_i^*\right) \mathbb{P}(O_s \le c_s^*) \\ &\ge \mathbb{P}\left(\sum_{i=1}^{s-2} O_i \le \sum_{i=1}^{s-1} c_j^*\right) \mathbb{P}(O_{s-1} \le c_{s-1}^*) \mathbb{P}(O_s \le c_s^*) \\ &\ge \cdots \ge \prod_{i=1}^{s} \mathbb{P}(O_i \le c_i^*) \end{aligned} \quad (14)$$

From Lemma 1, we can deduct that when using a fixed amount of processing capacity, the hierarchical fog computing always has a higher chance to successfully schedule the workloads.

Corollary 1: From Theorem 1 and Lemma 1, we can generate our analytic comparison model of fog hierarchy between multi-tier and flat fog architectures. To this end, we partition the offloaded workloads $O_i, (i = 1,,2, \dots, s)$ into K mutually groups $\{G_1, G_2, \dots, G_K\}$. Thus, for any $c_k^* \in [0, C] (k = 1, 2, \dots, K)$ and $\sum_{k=1}^{K} c_k^* = C$, we have:

$$\mathbb{P}\left(\sum_{i=1}^{s} O_i \le C\right) \ge \prod_{k=1}^{K} \mathbb{P}\left(\sum_{j \in G_k} O_k \le c_k^*\right) \quad (15)$$

Eq. (15) indicates that the efficiency of resource utilization in the 2-tier fog hierarchy will be maximized when there is only one tier-2 server. When $s = K$ in Eq. (15), the number of tier-1

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/TII.2018.2843802, IEEE Transactions on Industrial Informatics

and tier-2 servers are the same; as results, the flat and hierarchical designs of fog computing will be equivalent to each other. Clearly, we can also extend the results for hierarchical fog computing with more than 2-tier by repeating the above analysis recursively. For example, to construct a 3-tier fog hierarchy, we can simply use $\sum_{i=1}^{s} O_i$ as the workload of a tier-2 server.

IV. PRIORITY QUEUING MODEL FOR SCHEDULING IIoT DATA

As seen earlier, we consider a number of fog cells or data centers, where each data center contain servers deployed hierarchically. These servers handle the scheduling process and analysis data coming from different things installed inside the industrial factory. Incoming data, requests and information from different industrial things and sensors have different degrees of importance, e.g., emergency requests. The requests with high priority need to be scheduled and analyzed quickly.

We describe the queuing system with two priorities by making the following assumptions [12]:

Assumption 1(The input): the interval times of data of each class are mutually independent and identically distributed random variables. Thus, the input consists of a superposition of two independent renewal processes corresponding to these classes. However, it must be noted that the total input of IIoT data regardless of their class is not in general a renewal process.

Assumption 2 (The service mechanism): there are *s* servers in each fog tier. The services times of scheduling requests from IIoT are mutually independent random variables. their distribution being the same for requests in the same class, but they are different from that of other classes.

Assumption 3 (Queue discipline): the two classes are numbered 0 for low priority and 1 for high priority. Thus, a request of class 1 is always served prior to a customer of class 0. Within each class, the queue discipline is first come, first served (FIFO) with pre-emptive discipline. In the pre-emptive case, the request of class 0 (low priority) returns to the head of the queue of his class and waits until the newly arrived request of class 1 are served.

First, we describe the queuing system with two priorities [12]. The workload parameters are defined as follows:

$$\lambda = \lambda_1 + \lambda_0 \tag{16}$$

$$\mu = \mu_1 = \mu_0 \tag{17}$$

$$\rho_1 = \frac{\lambda_1}{\mu}, \rho_0 = \frac{\lambda_0}{\mu} \text{ and } \rho = \rho_1 + \rho_0 \tag{18}$$

Where λ_1 and μ_1 are respectively the arrival rate and the service rate of demands with high priority. λ_0 and μ_0 are the arrival and the service rate of demands with low priority, respectively. Definitely, when arriving, IIoT requests are placed in different queues of each server; each of which has two services priority, high and low.

As a stochastic model, the queuing process theory is represented using Markov chains by incorporating information in the state description. We consider that each state in the Markov chain corresponds to the number of IIoT requests in the two queues, and the state transitions occur when a new IIoT requests arrives or an IIoT requests is served. We use the birth-death process as a stochastic model to describe the evolution of our system.

The model considers two cases: the first is when the number of waiting requests, denoted by r, in the two queues is $r < s$, where s is the number of servers in each fog tier; the overall completion rate is then $r\mu$. The second case is when the number of waiting requests is $r \geq s$; this means that the entire servers are occupied, and the completion rate is $s\mu$.

We can obtain the general stationary distribution state using recursive demonstration to distinguish two cases according whether μ_r depends on r or not, as follow:

- if $r < s$:

$$\pi_r = \pi_0 \prod_{i=0}^{r-1} \frac{\lambda_0 + \lambda_1}{(i+1)\mu} = \pi_0 \left(\frac{\lambda_0 + \lambda_1}{\mu}\right)^r \frac{1}{r!} \tag{19}$$

- if $r \geq s$:

$$\pi_r = \pi_0 \prod_{i=0}^{s-1} \frac{\lambda_0 + \lambda_1}{(i+1)\mu} \prod_{j=s}^{r-1} \frac{\lambda_0 + \lambda_1}{r\mu} = \pi_0 \left(\frac{\lambda_0 + \lambda_1}{\mu}\right)^r \frac{1}{s!\, s^{r-n}} \tag{20}$$

The total probability condition is as follows:

$$\pi_0 = \left(\sum_{r=1}^{s-1} \frac{(\rho_H + \rho_L)^r}{r!} + \frac{(\rho_H + \rho_L)^s}{s!} \frac{1}{s!\, s^{r-n}} \right)^{-1} \tag{21}$$

Finally, from Eq. (19), (20) and (21) we define the mean waiting time (low $E(W_0)$) and (high $E(W_1)$) as follows [12]:

$$E(W_0) = \frac{\frac{\lambda_0}{{\mu_0}^2} + \frac{\lambda_1}{{\mu_1}^2}}{(1-\rho_1)(1-\rho_0-\rho_1)} \tag{22}$$

$$E(W_1) = \frac{\rho_1}{(1-\rho_1)\mu_1} \tag{23}$$

V. IIoT DATA ASSIGNMENT

In this section, we focus on determining which fog server's workloads are placed on to be scheduled. In addition, we define how much the computation capacity is provisioned to execute each request, so as to optimize the performance of executing all IIoT workloads and to take an optimal decision in a real time way to minimize the response delay.

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/TII.2018.2843802, IEEE Transactions on Industrial Informatics

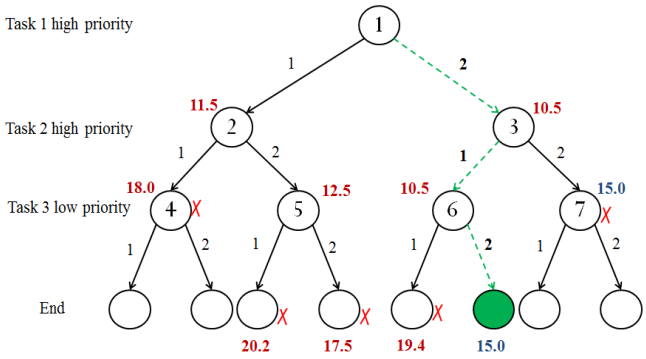


Fig. 3: The branch and bound process based on BnB algorithm

A. Assignment problem formulation

We first study a scenario which considers only one server at each tier of the fog. Then, we aggregate the decisions of workload offloading at different tiers. We formulate the workload placement problem as a mixed nonlinear integer programming (MNIP) problem of fog tier which has only one server, considering integers variables to indicate the locations of things requests being placed and non-integer variables to indicate the capacity allocated to execute each request.

We assume that there are m computing tasks with amounts of computations $\{w_1, w, \dots, w_m\}$, the data size of program states $\{D_1, D_2, \dots, D_m\}$, and n servers at the fog layer with capacities $\{c_1, c_2, \dots, c_n\}$ (i.e., CPU cycles per second).

The MNIP problem is given as follows:

$$\min \mathbb{f} = \sum_{i=1}^{m} \left(\frac{D_i}{B_\psi}(\psi - 1) + \frac{w_i}{\theta_i^\psi . c_\psi} + E\left(W_{(0,1)}{}^\psi\right) \right) \quad (24)$$

$$\text{Subject to } \sum_{i \in set_j} \theta_i^j = 1, j = 1, 2, \dots, n$$

Where ψ indicates the server where task i is placed on, θ_i^ψ is the percentage of server ψ's computational capacity being allocated for task i, and set_j denotes the set of computing tasks placed at server j. The rate $\frac{D_i}{B_\psi}(\psi - 1)$ measures the communication delay of transmitting task i states to server ψ, which is determined by the network bandwidth B_ψ allocated to ψ. The rate $\frac{w_i}{\theta_i^\psi . c_\psi}$ measures the computational delay of task i's execution. Finally, $E\left(W_{(0,1)}{}^\psi\right)$ measures the waiting time of task i in each ψ server's queue. (i.e., the waiting time is calculated in Eq. (22) or Eq. (23) according to the priority class of each task).

Theorem 2: To solve the MNIP problem in Eq. (24), we first study the optimization problem when ψ is known to each task i. When each server ψ is fixed, the optimization problem in Eq. (24) becomes a convex optimization problem.

Proof: According to [16], function $\mathbb{f}$ in Eq. (24) is strictly convex if its Hessian matrix H is positive definite for all the variables. So, we can prove that the Hessian matrix of function $\mathbb{f}$ is a positive definite matrix. To do this, we use θ_i to substitute θ_i^ψ, thus $\mathbb{f}$ in Eq. (24) becomes a function of variables $\{\theta_1, \theta_2, \dots, \theta_i\}$. For each pair of (θ_i, θ_j), we have:

$$\frac{\partial^2 \mathbb{f}}{\partial \theta_i \partial \theta_j} = \begin{cases} \frac{1}{2} w_i c_\psi^{-1} \theta_i^{-3}, & if\ i = j \\ 0 & otherwise \end{cases} \quad (25)$$

According to [13], the Hessian matrix $\mathrm{H} = \left(\frac{\partial^2 \mathbb{f}}{\partial \theta_i \partial \theta_j}\right)_{s*s}$ of $\mathbb{f}$ is symmetric and positive definite. Therefore, since the constraint in Eq. (24) is linear, the optimization problem in Eq. (24) is a linear convex optimization problem.

From Theorem 2, we further extend the MNIP problem as follows:

When each ψ is fixed, the optimal value of $\mathbb{f}$ is as follows:

$$\mathbb{f}_{min} = \sum_{j=1}^{n} \left[\sum_{i \in set_j} \frac{(j-1)D_i}{B_\psi} + \frac{\left(\sum_{i \in set_j} \sqrt{w_i}\right)^2}{c_j} + \sum_{i \in set_j} E\left(W_{(0,1)}{}^j\right) \right] \quad (26)$$

Hence, the corresponding optimal solution to the MNIP problem in Eq. (29) is:

$$\theta_i^* = \frac{\sqrt{w_i}}{\sum_{i \in set_j} \sqrt{w_i}} \quad (27)$$

Corollary 2: when the placement for each demand is fixed, the corresponding capacity provisioning problem for scheduling these demands is deterministic and can become a combinatorial optimization problem using the transformation lemma. Hence, for each possible assignment, we can solve the problem from theorem 2 and Eq. (26).

B. Branch and bound approach to design workload assignment algorithm

To determine the optimal value of $\{\theta_i\}$ and based on the above problem formulation (i.e., Eq. (26)), we develop in this subsection IIoT workload and data assignment algorithm. As seen earlier, we have m computing tasks and n servers, so the time complexity to exhaustively search the solution space is $O(n^m)$. As we can see, the time complexity is very important. However, to further decrease the searching complexity and obtain the optimal solution space, we propose to use a branch and bound approach [14], as shown in Fig. 3. In our work, the research space corresponds to the definition space of the optimal value of $\{\theta_i\}$. To this end, and without loss of generality, we assume that the IIoT workloads $\{w_1, w_2, \dots, w_m\}$ are successively determined to be assigned. The sub-problems of the workload assignment problem is denoted by w_y. Based on this sub-problem definition, we have the following lemma:

Lemma 2: Let $\mathbb{f}_y$ indicates the total delays for the y sub-problems having been branched, the lower bound on the total delays for the remaining $m - y$ tasks is:

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/TII.2018.2843802, IEEE Transactions on Industrial Informatics

$$\Upsilon_y = \mathbb{f}_y + \sum_{i=y+1}^{m} [\min_{1\le j\le n}\{\frac{w_i}{c_j} + (j-1)\frac{D_i}{B_j} + E(W_{(0,1)}{}^j)\}] \quad (28)$$

Proof: In Eq. (28), all the remaining tasks ($m-y$ tasks) are provisioned with 100% capacity of a server, consequently Υ_y provides a lower bound on the total amount of delays for the remaining $m-y$ tasks.

Based on Lemma 2, the branching process is a depth-first search over subproblems placing subsets of workloads, with the ones of the smallest lower bounds being searched first. Each time when a leaf node in the search tree is reached, all other branches with larger values of lower bounds are pruned. The optimal solution will be found until all the leaf nodes have been either searched or pruned.

Example: In order to explain more the branching process using BnB algorithm, we further illustrate an example in Fig. 3. To this end, we define the values of each variable as follows: the number of tasks $m = 3$, the number of servers $s = 2$, and the amounts of computations for each task is $\{{c_1}^H, {c_2}^H, {c_3}^L\} = \{2, 3, 4\}$ where tasks 1 and 2 are high priority tasks while tasks 3 is a low priority one. The capacity of each server is $\{c_1, c_2\} = \{1, 2\}$, and the communication delays between tier-1 and tier-2 servers (e.g., the rate $(j-1)\frac{w_i}{B_j}$) for the three tasks are $\{2, 3, 3\}$. Finally, the waiting times in high and low priority queues of the two servers are: $\{\left(\frac{1}{2}\right)^H, \left(\frac{3}{2}\right)^L\}$ for server number one, and $\{\left(\frac{1}{4}\right)^H, \left(\frac{3}{4}\right)^L\}$ for server number two. The number indicated by the side of each node represents the lower bound of this sub-problem computed and the number on each edge indicates the decision of workload placement. For example, when task 1 is placed on tier-1, the lower bound of total delay on node 2 is $\{(\frac{2}{1}+\frac{1}{2}) + (\frac{3}{1}+\frac{1}{2}) + (\frac{4}{1}+\frac{3}{2}) = 11.5$, however, when it is placed on tier-2, the lower bound on node 3 is $\{(\frac{2}{2}+2+\frac{1}{2}) + (\frac{3}{1}+\frac{1}{2}) + (\frac{4}{1}+\frac{3}{2}) = 10.5$. As $10.5 < 11.5$, we further branch the next sub-problem until we reach a leave node being colored as green in Fig. 3. Formerly, the leftmost and rightmost branches are automatically pruned because their lower bound is larger than 15.0. Therefore, the optimal solution of this example is $\{2, 1, 2\}$, with the total delay of 15.0.

C. Aggregation of optimization results over high tiers

In order to ensure a global optimality of IIoT requests assignment, we aggregate the solutions over different high-tier servers. The idea is to allocate the computational capacity of high-tier fog server on demand, when it receives offloading requests from multiple low-tier servers. Since there is one and only one optimal solution for a branch and bound searching process, the optimal allocation at a server that minimizes the cumulative delay of executing IIoT workloads is also unique.

The key challenge of aggregating the optimization results is how to allocate computational capacity of a higher-tier server to each branch of the fog hierarchy. To do this, let z denotes the capacity allocation vector of a higher-tier server to its children; a direct solution is a numerical iteration. As in [15], we adopt the Simulated Annealing Algorithm (SA) to find out the optimal allocation using numerical iterations. We describe SA in Algorithm 1. The idea is to generate a new iteration state z_{new} from the previous state z_{old}. T is the simulated temperature; at the beginning, T equals to T_{max} and decreases at each iteration by a cooling parameter $\xi (0 < \xi < 1)$.

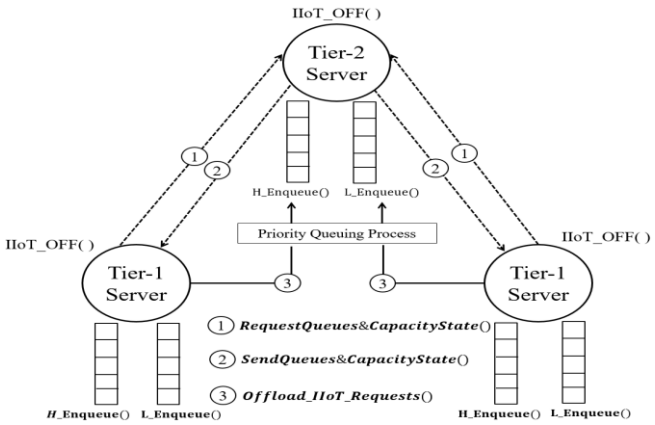


Fig. 4: Implementation of IIoT_OFF for two tiers fog hierarchy

Then, we calculate the difference of total delay, denoted as Δd, by solving the optimization problem in Eq. (26). If $\Delta d < 0$, z_{new} is accepted; otherwise, we calculate the probability $p = e^{(-\frac{\Delta d}{T})}$ to accept z_{new}. The SA algorithm iterations terminate when $T = T_{min}$. The simulated annealing algorithm is given as follows:

Algorithm 1: Simulated Annealing Algorithm (SA)

Input: Initial solution z_{old};
Output: Optimal allocation solution;

1. Randomly generate an initial solution z_{old};
2. $T \leftarrow T_{max}$; //initialization of the simulated temperature
3. **While** $T > T_{min}$ **do**
4. Generate a new solution z_{new};
5. $\Delta d = \mathbb{f}(z_{new}) - \mathbb{f}(z_{old})$; //by solving the problem in (24)
6. **If** $\Delta d < 0$ **then**
7. $z_{old} \leftarrow z_{new}$; // z_{new} accepted as optimal solution
8. **end**
9. $prob = e^{(-\frac{\Delta d}{T})}$; //calculate the probability to accept z_{new};
10. **else if** $rand(0,1) < p$ **then**
11. $z_{old} \leftarrow z_{new}$;
12. **end**
13. $T \leftarrow T.\xi$;//cooling parameter $(0 < \xi < 1)$;
14. **end**

D. IIoT offloading algorithm over k tiers of fog hierarchy

Finally, we implement the offloading algorithm considering k tiers in the fog hierarchy. Where each tier servers have two workload queues (high priority queue and low priority queue). When a task arrives from different factory devises to the first tier level, the server invokes functions $H\_Enqueue(IIoT\_Request)$ or $H\_Enqueue(IIoT\_Request)$

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/TII.2018.2843802, IEEE Transactions on Industrial Informatics

to put the workload into the appropriate queue according to its priority level.

If the queue is not empty and there is a spare computing capacity on the tier-k server, one task will be popped out from the queue and processed. Otherwise, if the tier-1 server is fully loaded, it will call $RequestQueues\&CapacityState()$ to check the availability of the $tier-(k+1)$ server. Only after having received the reply from the $tier-(k+1)$ server which invokes the $SendQueues\&CapacityState()$ function and confirmed that the $tier-(k+1)$ server has a spare computing capacity available and the optimal solution was founded, the $tier-(k+1)$ server will aggregate the optimal solution based on Algorithm 1 (i.e., SA Algorithm), and allocate the computational capacity of $tier-(k+1)$.

Finally, $tier-(k+1)$ receives the offloaded tasks from the lower tier servers which the $Offload\_IIoT\_Requests()$ function invokes and the tasks will be queued in the appropriate queue according to its priority levels. However, if the optimal solution was not founded, the branch and bound research is executed for higher tiers ($k = k+1$) until finding the optimal offloading solution with an optimal response delay for each task. The process of offloading tasks is presented in Algorithm 2 called IIoT_Off. Fig. 4 illustrates an example execution, considering the implementation of two fog hierarchies with three servers, two of which are used as tier-1 servers and another is used as the tier-2 server.

Algorithm 2: IIoT Offloading Algorithm (IIoT_Off)

Input: Number of tiers, Number of servers at each tier, $IIoT\_Requests(W, Priority)$;
Output: Optimal offloading solution, Optimal response delay;

1. Tier-1 servers receive $IIoT\_Requests(W, Priority)$;
2. $k = 1$;
3. **if** $IIoT\_Requests(W, Priority = 1)$ **then**
4. **For** each server i of Tier-k **do**
5. **if** $(\sum_{j=1}^{E_i(Q_1)} W_j \leq C_i)$ **then**
6. $H\_Enqueue(IIoT\_Request)$;
7. **else**
8. Calculate the offloaded workloads as in Eq. (2);
9. **While** $((\sum_{i=1}^{S} O_i)_{\boldsymbol{k}} \geq (\sum_{i=1}^{S} C_i)_{\boldsymbol{k+1}})$ **then**
10. $RequestQueues\&CapacityState()$from Tier-($k+1$);
11. $SendQueues\&CapacityState()$ to Tier- k servers;
12. Calculate optimal assignment using BnB as in Eq. (28);
13. **if** (optimal solution founded in Tier-($k+1$)) **then**
14. Aggregate the optimal solution using SA Algorithm;
15. Allocate the computational capacity of Tier-($k+1$);
16. $Offload\_IIoT\_Requests()$;
17. $H\_Enqueue(IIoT\_Request)$ in Tier-($k+1$) servers;
18. **else**
19. $k = k+1$;
20. **end**
21. **end**
22. **end**
23. **end**
24. **else** $IIoT\_Requests(W, Priority = 0)$ low priority requests;
25. We repeat the same instruction for low priority requests.
26. **end**

VI. SYSTEM PERFORMANCE EVALUATION

In this section, we implement extensive simulations to evaluate the performance of the proposed fog architecture, and compare it (i.e., hierarchical architecture) with the flat fog architecture and relevant recent works [9], [11]. We measure the proposed architecture performance using realistic industrial data from Bosch group case study presented in [3].

A. System settings and performance metrics

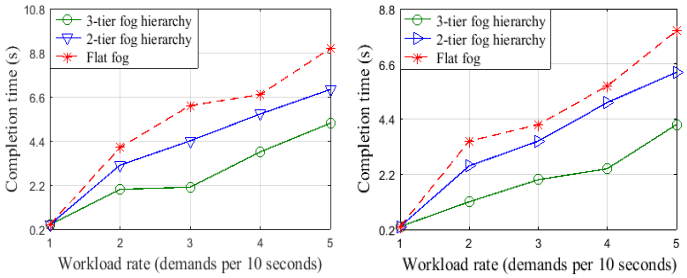
We use multiple virtual machines VM as fog servers, and each VM has a CPU capacity of 1.45 GHz and 2GB RAM. We use NS-2 to implement our network topologies [16]. In our experiments, industrial IoT tasks and datasets are received by 4 tier-1 fog servers. Our experiments are conducted over different settings of the hierarchical fog with different topologies and computational capacity being provisioned. Every two fog servers are connected via a 100Mbps network link. A total amount of 20 GHz computational capacity (in terms of the amount of CPU cycles per second) is provisioned. Each VM of the network topologies is running MATLAB to execute all the proposed algorithms. We evaluate the performance of each topology using the following metrics:

- *Workload rate:* The number of tasks, which indicates the workload rate, is determined every time following the same Poisson distribution.
- *Provisioned capacity:* The computational capacity provisioned to a fog server is measured by the number of computing threads running concurrently on the server.
- *Waiting time:* It is the waiting time at each priority queue.
- *End-to-end delay:* This is the time it takes for the first response to come after processing of IIoT dataset.
- *Completion time:* It indicates the amount of computational capacity being provided by fog servers.
- *Synchronization Time:* The time interval between the time point that the IIoT devices get the readings and that of the fog servers receive these readings.

B. Experiment results

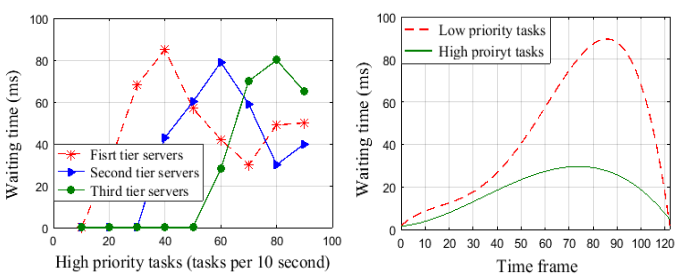
In order to study the performance of the proposed fog architecture, we conduct simulations experiments over the three topologies of fog architecture, i.e., 3-tiers fog hierarchy, 2-tiers fog hierarchy and flat architecture. First, we evaluate the completion time performance of each topology by fixing the provisioned capacity and change the workload rate from 1 to 5 tasks per 10 seconds. The results are shown in Fig. 5. As we observe in Fig. 5(a), when the provisioned capacity is 2, there is large difference between the performance of flat and hierarchical designs (e.g., 3-tier fog hierarchy and 2-tiers fog hierarchy); the 3-tiers hierarchy performs better and gives an optimal completion time. Rationally, when the provisioned capacity is 4, the completion time is more optimal as we can see in Fig. 5(b). The major reason for this difference is that the tier-3 server is able to efficiently aggregate the peak loads from all tier-2 servers, and tier-2 servers are able to efficiently offload the IIoT peak loads from tier-1 servers. Hence, each hierarchical tier (e.g., 3-tiers and 2-tiers) utilizes the provisioned capacity to process the tasks at a much higher

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/TII.2018.2843802, IEEE Transactions on Industrial Informatics



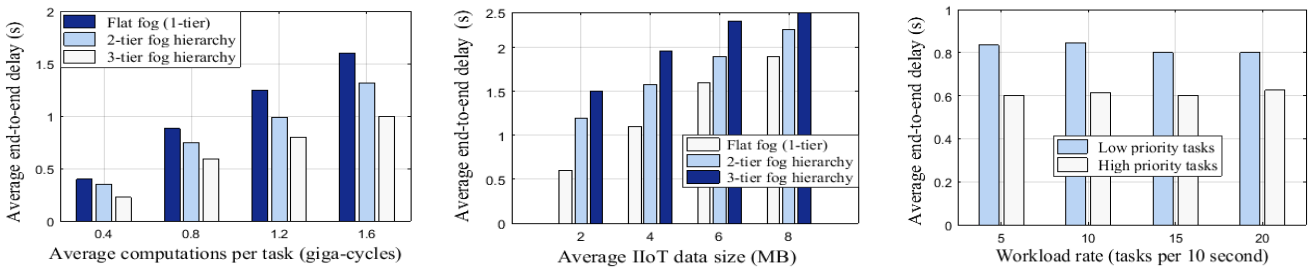
(a) provisioned capacity = 2 (b) provisioned capacity = 4

Fig. 5: Average completion time over different amounts of workloads



(a) Over dififfrent tiers hierarchy (b) Over dififfrent time frames

Fig. 6: Average waiting time comparison



(a) Over different computations tasks (b) Over different IIoT data sizes (c) For high and low priority tasks

Fig. 7: Average end-to-end delay over different topologies

efficiency by executing the proposed IIoT Offloading Algorithm (IIoT_Off).

Now, we evaluate the proposed priority queuing mechanism implemented over each tier server. To do this, we examine the waiting time at each queue; the results are shown in Fig. 6 over 3-tier hierarchy topology. Fig. 6(a) shows the comparison of waiting time at high priority queue (i.e., dedicated for high priority tasks) over the three tier servers of the fog hierarchy. We observe that for each tier, the waiting time increases and decreases periodically; the average waiting time in tier 1 servers increases first and starts decreasing when the waiting time in tier 2 servers' increases. Similarly, when the waiting time in tier 3 server increases, the waiting time in tier 2 servers' decreases. This is explained by the offloading process (IIoT_Off algorithm) of high priority tasks over different tiers as follows: First, tier 1 servers receive the workloads from IIoT; formerly the workloads are queued in the queue, so the waiting time increases until $\sum_{j=1}^{E_i(Q_1)} w_j > c_i$ (i.e., the sum of queued workloads computation becomes superior to the capacity of each tier 1 server). As results, the next received workloads will be offloaded to tier 2 servers which leads to a decrease in the waiting time for tier 1 servers and an increase in the waiting time in tier 2 servers. In the same way, we can explain the increase/decrease relationship of waiting time between tier 2 servers and tier 3 servers.

Fig. 6(b) shows the average waiting time comparison between high and low priority tasks over different time frames (e.g., 50% of high priority tasks and 50% of low priority tasks). In this experimentation, we suppose that the time frames from 60 to 90 contain a high amount of IIoT computation tasks where 80% of them are high priority tasks and 20% are low priority ones. We can see an increase in the waiting time for both high and low priority tasks from frames 60 to 90. However, the waiting time for high priority workloads is more stable than for low priority workloads, and the difference is very big between frame 60 and 90 in Fig. 6(b). This is because there are around 80% of high priority tasks in this frames interval (i.e., 60 and 90), and the proposed queue discipline is pre-emptive (the low priority tasks returns to the head of the queue of its class and waits until the newly arrived high priority tasks is served). As results, from Fig. 6 we can conclude that the proposed priority queuing mechanism helps the IIoT_Off algorithm to balance the high workloads over all tier servers in order to minimize the waiting time at each queue.

Now, we study why the end-to-end delay is an important parameter to consider in Industry 4.0 and smart factories based IoT. We propose a comparison study between each tier topology over various computation tasks (e.g., Fig. 7(a)), different IIoT data size (e.g., Fig. 7(b)), and several priority levels of tasks (e.g., Fig. 7(c)). As shown in Fig. 7(a), the 3-tier fog architecture can reduce the end-to-end delay by about 40% compared to that of the flat architecture. In addition, the 2-tier hierarchy reduction is around 20%, this is due to its incapability of aggregating enough peak loads from lower tiers. Indeed, the 3-tier fog hierarchy provides more computational capacity on the higher tiers servers and less capacity on the first tier, leading to a more iterative aggregation of devices demands. In Fig. 7(b), we evaluate the end-to-end delay of the proposed architecture over different IIoT data sets sizes.

As known, larger data sizes increase the communication delay for transmitting workloads to high tiers; however, our proposed scheduling scheme can still optimize the offloading of industrial things requests by reducing the total delay by 30% with different data sizes. Mainly, the 3-tier fog hierarchy can also outperform other topologies, because of its efficient iterative aggregations of the peak loads using IIoT_Off algorithm. Fig. 7(c) shows the average end-to-end delay for

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/TII.2018.2843802, IEEE Transactions on Industrial Informatics

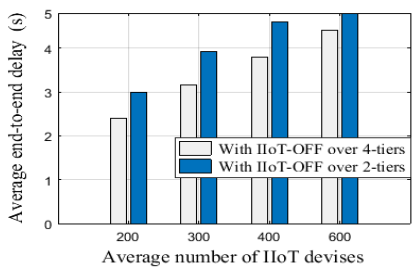
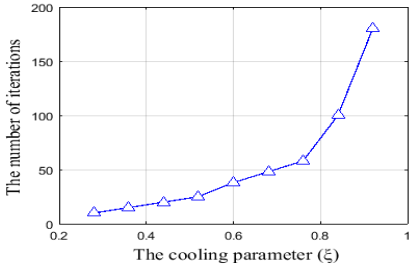
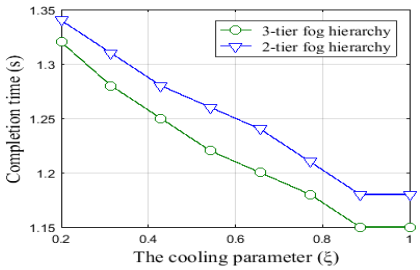


Fig. 8: Average end-to-end delay over IIoT-OFF Algorithm for 4-tiers and 2-tiers



(a) The number of iterations



(b) The completion time

Fig. 9: The effect of the cooling parameter

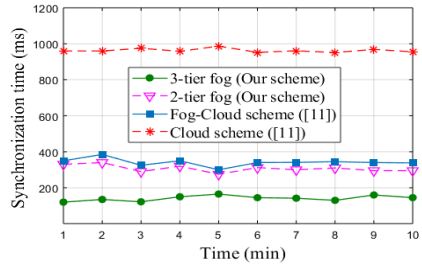
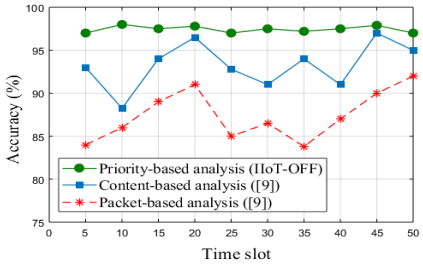
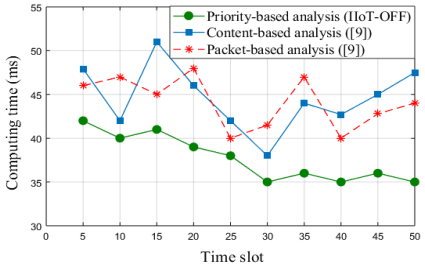


Fig. 10: Synchronization time comparison



(a) Accuracy analysis



(b) Computing time analysis

Fig. 11: Comparison of the perfromences of the proposed scheme

high and low priority tasks. As explained earlier, the end-to-end delay for high priority requests is better than that of low priority requests; this is because the waiting in high priority queues is smaller compared to that in low priority ones (see Fig. 6(b)).

Fig. 8 evaluates the performance of the offloading algorithm over 2-tiers and 4-tiers hierarchy, where $k = 2$ and $k = 4$ in IIoT_Off algorithm. We increase the number of IIoT devices from 200 to 600 where each device sends the same amount of data size (2 MB). As shown in the figure, with the increase in the number of interconnected industrial devices, the end-to-end delay increases for the two topologies, however it is smaller when we offload workloads over 4-tiers than over 2-tiers. This is mainly because the 4-tier fog hierarchy provides more computational capacity on the higher tiers servers and less capacity on the first and second tiers.

Finally, we study the impact of the SA algorithm cooling parameter (ξ) on the performance and time complexity in Fig. 9. The experiment results are shown in Fig. 9(a). Since a larger value of ξ helps the SA algorithm process to converge, it further reduces the total delay of task completion time. On the other hand, when the value of ξ is too small (< 0.6), the iterations may not be enough for the SA algorithm to converge, incurring higher delay of task completion. Fig. 9(b) shows that the 3-tier fog has the lowest delay compared to other 2-tier fog under different cooling parameters values, even though the result has not reached the convergent value. The reason is that although the cooling process accelerates, the 3-tier fog hierarchy can still aggregate more peak loads from lower tiers compared to the 2-tier hierarchy.

C. *Comparison with exiting works*

Now, we compare the proposed hierarchical fog computing and the scheduling approach of IIoT requests with two recent works (i.e., [11] and [9]), based on three principal metrics (i.e., synchronization time, accuracy, computing time, and end-to-end delay). First, we consider the same simulation parameters used in each work. Then, we adjust our simulation scenario with the same parameters and we run simulations.

Fig. 10 illustrates the performance of the proposed hierarchical fog architecture compared with two schemes (i.e., fog-cloud scheduling scheme and cloud scheme) proposed in [11] (see related work section). As in [11], in our proposed model, there is an interesting tradeoff between the data preprocessing time and data transmission time (due to the hierarchical deployment of the servers). We employ the synchronization time to measure the time efficiency of our model and compare it with the two schemes in [11] (i.e., traditional scheme which uses only cloud computing, and the framework that uses edge servers and cloud). We can observe that our two schemes (3-tier fog and 2-tier fog) are much more time-efficient than the traditional scheme (only cloud [11]) and the edge framework scheme [11]. This is reasonable considering that multiple links between the fog servers' hierarchy are built in our architecture, while only one link is built in [11] scheme which can provide an important overhead. Our proposed model ensures both offloading IIoT requests based on priority queuing discipline and optimal capacity allocation and aggregation over different fog tiers; this make it consumes very little time compared to [11].

Finally, we compare in Fig. 11 the performance of our proposed scheduling scheme (i.e., IIoT_Off algorithm) with a recent work [9] (see related work section). In [9], the authors design an efficient and simple content-aware data processing rule, in order to define how to design an optimal computing manner for a general IIoT (i.e., decentralized computing or centralized computing). We first examine the correctness of the proposed priority-based analysis method compared with the content-aware analysis and packet-based analysis methods

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/TII.2018.2843802, IEEE Transactions on Industrial Informatics

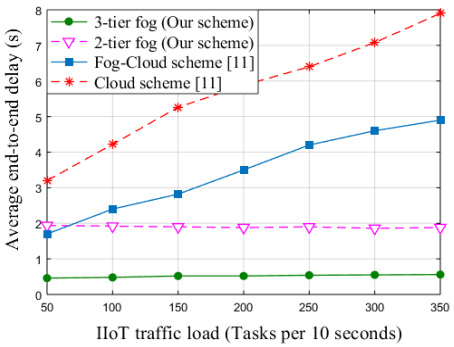


Fig. 12: Average end-to-end delay comparison with existing work [11]

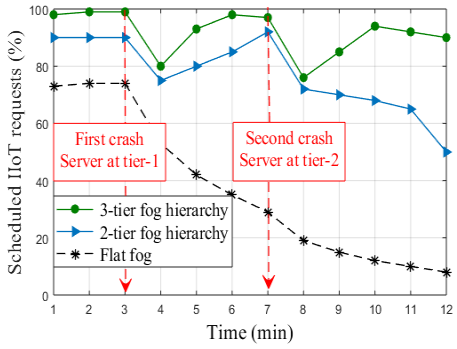


Fig. 13: Performances of our scheme when servers' crashes at different tiers of the hierarchy

(used as the benchmark). The comparison results are shown in Fig. 11(a) from the aspects of analysis accuracy, and in Fig. 11(b) from the aspect of computing time.

As we can see in Fig. 11(a), the priority-based analysis using IIoT_Off algorithm can achieve at least 6% on the analysis accuracy compared to content-aware method and more than 15% compared to the conventional one. The priority curve is remarkably stable during all time slots; this is due to the optimal offloading of IIoT requests over different fog tiers using the proposed queuing system at each server. Similarly, the computing time when using the proposed scheduling scheme is more optimal, reduced to 20 (ms) compared with the content analysis scheme, and to 30 (ms) compared with the packet analysis scheme. We observe also that the computing time curve of priority analysis is stable, and is decreased over highest time slots. We can explain this by the optimal resource allocation and workloads assignment over different fog tiers using the brand and branch method to solve the assignment problem in Eq. (26).

Fig. 12 illustrates the performance of the proposed hierarchical fog architecture with two topologies (i.e., 3-tier hierarchy and 2-tier hierarchy) compared with two schemes (i.e., fog-cloud scheduling scheme and cloud scheme) proposed in [11] (see related work section). We aim to examine the average end-to-end delay over different amounts of IIoT traffic loads (we increase the number of tasks at each 10 seconds). As we can see in Fig. 12, the two topologies of our proposed scheme (i.e., 3-tier, and 2-tier) outperforms the other schemes where the average end-to-end delay is stable between 0.5 and 0.7 second for 3-tier topology, and between 1.8 and 2.0 second for 2-tier topology. However, the average end-to-end delay increases significantly for fog-cloud scheme (it can reach 5 seconds) and cloud scheme (it can reach 8 seconds) when we increase the number of tasks. it is well-known that the end-to-end delay includes the computing time, the transmission time of each requests, and the synchronization time (in the case of offloading of requests). As we have seen earlier, our proposed scheme proposes an appropriate mechanism to allocate the computation capacities for each server located at different tiers of the fog hierarchy; however, the work in [11] did not propose a mechanism to reduce the computing time. On the other hand, as we compared in Fig. 10, our two schemes are much more time-efficient (in terms of synchronization time) than the traditional scheme (only cloud [11]) and the edge framework scheme [11]. This is reasonable considering that multiple links between the fog servers' hierarchy are built in our architecture, while only one link is built in [11] scheme which can provide an important overhead, which may increase significantly the end-to-end delay.

Given these results from the comparison with relevant existing works, we find that the proposed scheduling scheme based on hierarchical fog servers' deployment and priority queuing outperforms the works [9] and [11] in terms of synchronization time (i.e., data transmission over different servers), computing time, processing accuracy, and average end-to-end delay.

D. *Performances of our scheme when servers' crashes at different tiers of the hierarchy:*

Finally, we examine the performance of our scheme when one of servers at different tiers of the hierarchy crashes. For this end, we study the percentage of scheduled IIoT requests during a simulation time of 12 minutes. We implement the three topologies (3-tier, 2-tier, and flat fog), and we generate the same load of IIoT traffic for each topology. Indeed, we simulate two crashes of servers; the first is caused at a server at tier-1 of each topology at 3 min, and the second is at a server located at tier-2 of each topology. The results are shown in Fig. 13; we investigate the results as follows:

- *Before the first crash (from 1 to 3 min):* we observe that the percentage of scheduled requests is stable for all topologies, and the 3-tier fog hierarchy topology outperforms the author topologies where it can reach 99% of scheduled requests, 91% for 2-tier hierarchy, and 72% for flat topology.

- *After the first crash (from 3 to 7 min):* we observe that the percentage decreases for the tier topologies from 3 to 4 min (i.e., to 80% for 3-tier, and to 70% for 2-tier). After 4 min, we observe that the percentage increases for the two topologies quickly to reach 98% for 3-tier and 90% for 2-tier. To loss of these requests can be explained by the crash of the server, where the requests scheduled in this server are not handled. However, we see that the problem is rapidly solved by offloading the incoming request to other servers of the hierarchy using IIoT_Off algorithm, and reallocating the computation capacities according to the new situation of the fog hierarchy (the proposed algorithms can be adapted rapidly to any situation change in the architecture by modifying the initial parameters as the number of servers at each tier). However, for flat architecture, we can see the percentage still increase until 35%

at 7 min, because we have only one tier (we can offload the requests to other tiers).

- *After the second crash (from 7 to 12 min):* we observe that the percentage of the scheduled IIoT requests for 3-tier topology decreases to reach 70% at 8 min, and it increases to be stable between 90% and 95%. However, for 2-tier topology, the percentage decreases significantly to reach 50% at 12 min. This is because we have only one sever at tier-2 for 2-tier topology, and by crashing the server at tier-2, the topology becomes flat where we cannot offload the IIoT requests, also, we cannot aggregate the optimal results, this is way the percentage did not increase. Nevertheless, for 3-tier hierarchy, even a server is crashed at tier-2, we still have a hierarchical topology with three tiers. Hence, we can adapt and execute our algorithms according to this situation.

VII. CONCLUSION

In this paper, we propose an efficient fog architecture for industrial internet of things applications, and introduce a new scheduling model for IIoT data processing. The proposed architecture is based on multi-tier servers' deployment at the fog layer. To efficiently schedule different industrial devices requests in a real time way, we developed an optimal workload assignment algorithm named IIoT-OFF by solving a mixed nonlinear integer programming (MNIP). Then, the optimal solution is aggregated over different tiers using a new Simulated Annealing Algorithm (SA). The advantages of the proposed hierarchical fog architecture compared to the conventional industrial system and two relevant recent works are proved over different performance metrics and through extensive simulations using realistic industrial data from Bosch group.

REFERENCES

[1] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash, "Internet of things: A survey on enabling technologies,protocols, and applications," IEEE Comm. Surv. Tutor.,vol. 17, no. 4, pp. 2347–2376, 2015.

[2] T. Qiu, X. Liu, L. Feng, and Y. Zhou, "An efficient tree-based selforganizing protocol for Internet of Things," IEEE Access, vol. 4, no. 6, pp. 3535–3546, Jun. 2016.

[3] P. Lade, R. Ghosh, and S. Srinivasan, "Manufacturing Analytics and Industrial Internet of Things," IEEE Intel. Sys. vol. 32, pp. 74-79, 2017.

[4] C. Alippi, R. Fantacci, D. Marabissi and M. Roveri, "A Cloud to the Ground: The New Frontier of Intelligent and Autonomous Networks of Things," in IEEE Commun. Mag., vol. 54, no. 12, pp. 14-20, Dec. 2016.

[5] D. A. Chekired and L. Khoukhi, "Smart Grid Solution for Charging and Discharging Services Based on Cloud Computing Scheduling," in IEEE Trans. on Indus. Inform., vol. 13, no. 6, pp. 3312-3321, Dec. 2017.

[6] Cisco white paper, "Fog Computing and the Internet of Things: Extend the Cloud to Where the Things Are," accessed on Nov. 2017. [Online]. Available:https://www.cisco.com/c/dam/en_us/solutions/trends/iot/docs/computing-overview.pdf.

[7] S. Mubeen, P. Nikolaidis, A. Didic, H. Pei-Breivold, K. Sandström, and M. Behnam, "Delay Mitigation in Offloaded Cloud Controllers in Industrial IoT," In IEEE Access, vol. 5, pp. 4418-30, April 2017.

[8] L. D. Xu, W. He, and S. Li, "Internet of Things in Industries: A Survey, "In IEEE Trans. on Indus. Inform., vol. 10, pp. 2233-43, Nov. 2014.

[9] L. Zhou, D. Wu, J. Chen and Z. Dong, "When Computation Hugs Intelligence: Content-Aware Data Processing for Industrial IoT," in IEEE Internet of Things Journal, vol. PP, no. 99, pp. 1-1. doi: 10.1109/JIOT.2017.2785624.

[10] Y. Sahni, J. Cao, S. Zhang and L. Yang, "Edge Mesh: A New Paradigm to Enable Distributed Intelligence in Internet of Things," in IEEE Access, vol. 5, no. , pp. 16441-16458, 2017.

[11] J. Fu, Y. Liu, H. C. Chao, B. Bhargava and Z. Zhang, "Secure Data Storage and Searching for Industrial IoT by Integrating Fog Computing and Cloud Computing," in IEEE Transactions on Industrial Informatics, vol. PP, no. 99, pp. 1-1. doi: 10.1109/TII.2018.2793350.

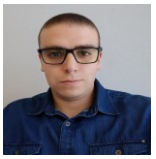
[12] N.U.Prabhu, "Fondations of Queuing Theory", Kulmar Academic Publishers, Massachusetts, 80-112, 1997.

[13] S. Boyd, and L. vandenberghe, "Convex optimization". Cambridge university press, 2009.

[14] J. C. Bazin, H. Li, I. S. Kweon, C. Demonceaux, P. Vasseur, and K. Ikeuchi, "A Branch-and-Bound Approach to Correspondence and Grouping Problems," In IEEE Trans. on Pattern Analysis and Machine Intelligence, vol. 35, no. 7, pp. 1565-76, Jul. 2013.

[15] S. Kirkpartrick, C. Gelatt, and C. Vecchi, "Optimization by Simulated Annealing". In Science, New Series, vol. 220, 1983.

[16] D. A. Chekired, L. Khoukhi and H. T. Mouftah, "Decentralized Cloud-SDN Architecture in Smart Grid: A Dynamic Pricing Model," in IEEE Trans. on Indus. Inform., vol. 14, no. 3, pp. 1220-1231, March 2018.



Djabir Abdeldjalil Chekired (S'17) received the B.S. degree in computer science and the M.S. degree in computer networks and distributed systems from University of Science and Technology Houari Boumediene (USTHB), Algiers, Algeria, in 2010 and 2012, respectively. He is currently working toward the joint Ph.D. degree in computer science with the Environment and Autonomous Networks Laboratory (ERA), University of Technology of Troyes, France, and the School of Electrical Engineering and Computer Science, University of Ottawa, Canada. His research interests include new cloud computing design for smart grid system and analysis in green electric vehicles networks, smart grid energy management, Fog-SDN architectures, and Internet of Things.

Lyes Khoukhi (M'09) received the Ph.D. degree in electrical and computer engineering from the University of Sherbrooke Canada, in 2006. In 2008, he was a researcher with the Department of Computer Science and Operations Research, University of Montreal, Canada. Currently, he is professor in computer science with the University of Technology of Troyes, France. He is the author or coauthor of more than 100 publications in reputable journals, conferences, and workshops. His research interests include IoT, cloud, smart grid and mobile communication protocols. Dr. Khoukhi has participated as a general Chair or TPC Member of many conferences.

Hussein T. Mouftah (S'74–M'76–SM'80–F'90) joined the School of Electrical Engineering and Computer Science, University of Ottawa in 2002 as a Tier 1 Canada Research Chair Professor, where he became a University distinguished professor in 2006. From 1979 to 2002, he was with the Department of Electrical and Computer Engineering, Queen's University, where he was prior to his departure a Full Professor and the Department Associate Head. He has six years of industrial experience mainly at Bell Northern Research of Ottawa (now Nortel Networks). He is the author or coauthor of 9 books, 63 book chapters, more than 1300 technical papers, 12 patents, and 142 industrial reports. He served as an Editor-in-Chief of the IEEE Communications Magazine (1995–1997) and IEEE ComSoc Director of Magazines (1998–1999), chair of the Awards Committee (2002–2003), director of Education (2006–2007), member of the Board of Governors (1997–1999 and 2006–2007), and member of the Nomination Committee (since 2012). He has been a distinguished speaker of the IEEE Communications Society (2000–2007). He is a fellow of the Canadian Academy of Engineering (2003), the Engineering Institute of Canada (2005), and the Royal Society of Canada RSC Academy of Science (2008).