Software Defined Industrial Network: Architecture and Edge Offloading Strategy

Fangmin Xu[1], Huanyu Ye[1(✉)], Shaohua Cui[2], Chenglin Zhao[1], and Haipeng Yao[1]

[1] Key Laboratory of Universal Wireless Communications, Ministry of Education, Beijing University of Posts and Telecommunications, Beijing 100876, China
huanyuyebupt@gmail.com

[2] China Petroleum Technology and Development Corporation (CPTDC), Beijing 100028, China
shcui@163.com

Abstract. The integration of the internet and the traditional manufacturing industry has identified the "Industrial Internet of Things" (IIoT) as a popular research topic. However, traditional industrial networks continue to face challenges of resource management and limited raw data storage and computation capacity. In this paper, we propose a Software Defined Industrial Network (SDIN) architecture to address the existing drawbacks in IIoT such as resource utilization, data processing and system compatibility. The architecture is developed based on the Software Defined Network (SDN) architecture, combining hierarchical cloud and edge computing technologies. Based on the SDIN architecture, a novel centralized computation offloading strategy in industrial application is proposed. The simulation results confirm that the SDIN architecture is feasible and effective in the application of edge computing.

Keywords: Software defined industrial network · Industrial internet of things · Edge computing · Computing offloading · Time delay

1 Introduction

Intelligent manufacturing (IM), which has been driven by Information and communication technologies (ICT), greatly improves the automation level, production quality and efficiency of manufacturing industry. These information

Supported by Key Program of the National Natural Science Foundation of China (Grant No 61431008) and Project of intelligent manufacturing integrated standardization and new model application.

X. Liu et al. (Eds.): ChinaCom 2018, LNICST 262, pp. 46–56, 2019.
https://doi.org/10.1007/978-3-030-06161-6_5

and communication technologies (ICTs) provide reduced Capital Expenditure (CAPEX) and Operating Expense (OPEX) with higher efficiency and effectiveness. However, owing to the inherent limitations and complex network protocols in traditional industrial networks, traditional industrial Ethernets cannot manage distributed resources flexibly. Moreover, collected raw data are becoming increasingly more granular and voluminous.To address these drawbacks, bringing the cloud computing resources nearer to the underlying networks is attractive and promising.

In [1], the authors proposed a low latency mobile edge computing (MEC) framework based on the SDN architecture. Security and privacy are two of the main challenges to the IoT; hence, [2] proposed solutions and models for securing IoT devices and communications using the SDN architecture. Meanwhile, the performance of SDIN using in data offloading and edge computing in several scenarios including cloud server and mobile tasks was discussed in [3–5]. Computation offloading technology as discussed in [6,7]. In [8] and [9], mobile devices can extend the standby time by computation offloading.

Other previous research addressed special purposes such as energy saving and real-time communication; however, they did not focus on the details of the entire system architecture and operation. Therefore, we present a new software defined industrial network (SDIN) architecture that is the combination of Software Defined Networking (SDN) and IIoT. Industry network intelligence and control are logically centralized in SDIN to provide greater processing performance and avoid the majority of the aforementioned drawbacks. Particularly, our contributions are as follows:

- We propose a new SDIN architecture, and analyze the control and management process.
- Based on the SDIN architecture and the characteristics of industrial computing tasks, we propose a hybrid centralized edge computing offloading strategy.

The remainder of this paper is organized as follows. Section 2 identifies the architecture of SDIN. Section 3 analyses the application of SDIN in solving the computing offloading problem and the unique features. The system model, problem formulation, and solutions are provided in Sect. 4. We present a performance simulation in Sect. 5. Finally, Sect. 6 concludes the work.

2 Software Defined Industrial Network

We propose an SDIN architecture as displayed in Fig. 1. The SDIN architecture contains four layers: field devices layer, data transport layer, distributed control layer and cloud platform. Field devices include the basic infrastructures such as robot arms, conveyor belts, lathes and deployed sensors, etc.

(1) Data Transport Layer

This layer is composed of SDN switches, wireless access points (APs). The APs emphasize authentication where IoT devices access the network and for data

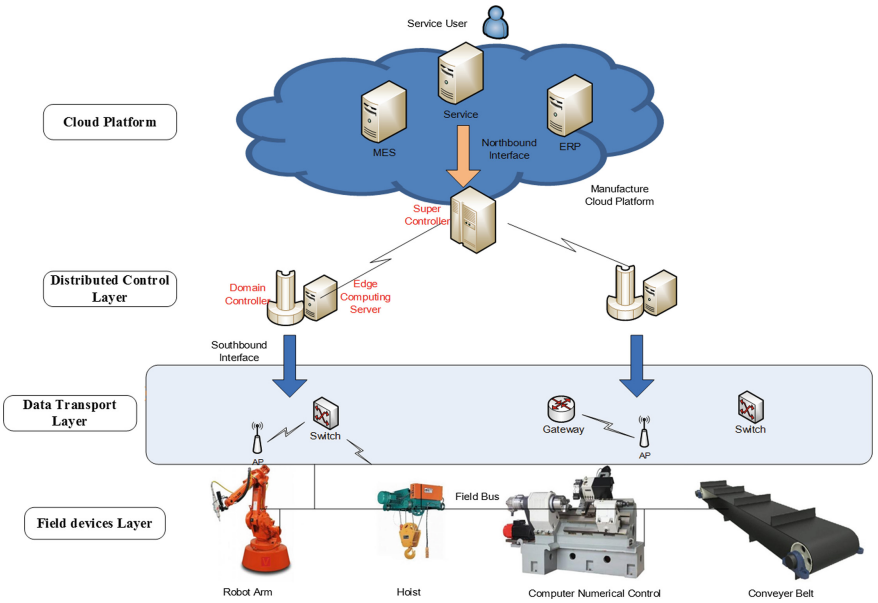

Fig. 1. Proposed software defined industrial network architecture

transporting. Data plane devices receive commands from control plane through southbound interface, such as computing decisions.

(2) Distributed Control Layer

The distributed SDN controllers are responsible for the centralized management of the edge computation servers and the authorization of multiuser access. Controllers receive requests from devices in the data transport plane and execute the offloading decision algorithm considering both the mission requirements and status of the edge computation servers. As displayed in Fig. 1, two-tier heterogeneous controller structure (domain controller and super controller) is one of the typical deployment solutions in large scale manufacture enterprise.

(3) Cloud Platform

The cloud platform includes a series of cloud service applications composed of industry application systems (Such as MES (Manufacturing Execution System), EPR (Enterprise Resources Planning)). The super controller places emphasis on resource management and the authorization of the distributed SDN controllers.

Besides, edge computing servers are more powerful than the local computing nodes in field devices. Due to the edge servers are located near the factories and production lines, it can provide lower and more stable latency than cloud computing server.

3 Computing Task and Offload in Industrial Scenario

In the concept of the intelligent factory, there are many computing tasks during the production process, such as Automatic Guided Vehicle (AGV) navigation, operation control of mechanical arms, and the identification of product imperfection. These services and applications can require significant computation resources and constrained time delay. However, the computational capabilities of the field devices are limited owing cost and size limitations.

3.1 Industrial Computing Task

After investigation, the industrial computing task has following unique features:

(1) Stringent computing delay tolerant
In industrial internet, the distributed sensors, actuators, machines, and other computing devices need to collaborate together to achieve real-time operation or complete the production tasks. In order to minimize the influence to the production line, the latency-sensitive industrial applications require delay from tens of milliseconds to hundreds of milliseconds.

(2) Diverse computing factors
As mentioned before, typical computing tasks in modern factory could be classified into following types with diverse characteristics and QoS requirements.

- Image or video recognition (such as quality inspection).
  Characteristic: huge amounts of raw input data, small size of computing result data.
- Localization and mapping (such as welding robot positioning guide).
  Characteristic: high computing accuracy, huge computing resource.
- Production planning and scheduling.
  Characteristic: multiple data sources, complex computing, low frequency.

To study the effects of the computation task characteristics on the design of offloading schemes, we classify the tasks into I types. Each type of computing task has various QoS requirements. According to computation task types, the field devices can be correspondingly classified into I types. Let us assume that one field device only generates one type of computing task.

(3) Regular task pattern
Generally, the computing tasks originate from the production line which has a fixed production cycle. For instance, the cycle of one product line is five products per minute, therefore the cycle of product imperfection identify task is also five times per minute.

To simplify the analysis, the arrival of single industrial computing task is modeled as regular arrival. However, considering the asynchronism of different field devices and production lines, the arrival of tasks at the edge server could be treated as poisson flow.

3.2 Computing Offload Procedure Based on the SDIN Architecture

The decision of whether local computing or offloading the computing task to the edge server is an important and difficult procedure. Figure 2 displays the computing decision process. The normal working process includes two stages: maintenance and update. In the maintenance stage, domain controllers broadcast the domain offloading strategy (the offloading probability of each computing task type, denoted as ξ_i, $i \in I = [1, 2, ...I]$) to field devices (① in Fig. 2). Field devices generate a random number between 0 and 1. If this number is less than ξ_i, then the computing will be offloaded to the edge server. Otherwise, the computing will be implemented in local computing unit. Update stage is trigged by the change of manufacturing environment and other factors, such as the increasing of computing task frequency, adjust of production scheduling. It contains three parts: requests collection, mode decision, computing, as follows:

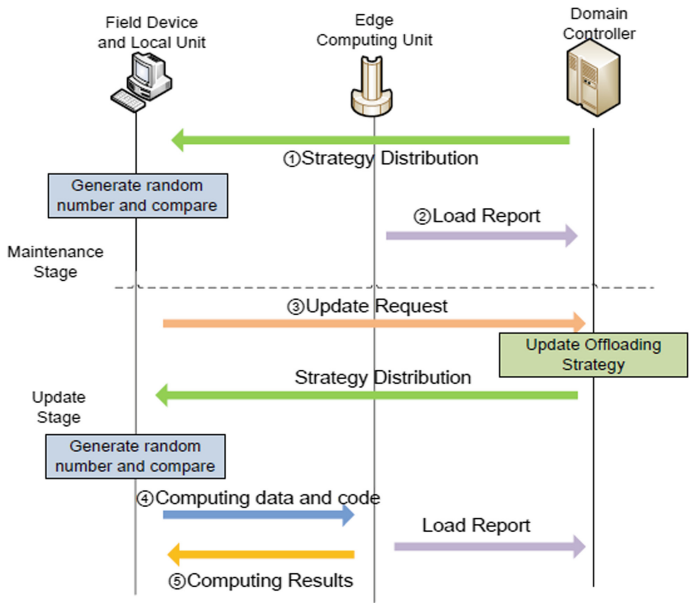

Fig. 2. The computing offloading decision process

(1) Update Request

The devices collect update data (information of computing tasks) and send it to the data transport plane through the APs. Then the update request message is send to its domain SDIN controller by southbound interface (③ in Fig. 2).

(2) Mode Decision

The domain controller exacts the computing capability parameters from the latest update request message and edge computing server (from the load report

message, as ② in Fig. 2), then it will execute the decision algorithm described in Sect. 4 and return the domain offloading strategy (the offloading probability of each computing task type, ξ_i).

(3) Computing

Similar with the maintenance stage, field devices decide whether offloading the task to the edge server or not after received the domain offloading strategy. For instance, if the device chooses the remote computing, field device will upload the necessary data and code to the edge server (④ in Fig. 2), and receive the computing result from the server afterward (⑤ in Fig. 2).

4 System Model and Offloading Algorithm

Considering most of computing devices in current factory are powered with electricity instead of batteries, the energy consumption of computing and data transfer is not a great issue. Therefore, the computing offload policy only considers the goal of minimizing computing delay.

The computing latency is divided into the following five aspects for decision analysis: local computing delay D_{Local}, data transmission delay from the local to the edge server T_{Trans}, task queuing delay for the edge computing server D_{queue}, edge computing server calculation delay D_{Remote}, and computing result return delay from the edge server to the local devices D_{Result}.

We denote the computing tasks of type i by $I_i = (D_i, C_i, T_i)$, where D_i denotes data size, C_i represents the size of computation data involved in the number of CPU cycles required to complete the type-i task, and T_i represents the maximum delay tolerance of the type-i task.

According to their computation task types, the proportion of the field devices with type-i tasks is given by π_i, where $i \in I$, and $\sum_{i \in I} \pi_i = 1$.

4.1 System Model

The local computing unit capability is defined as f_i^l, and f_i^r represents the CPU computation cycles per second that the edge server can provide. We assume that the transmission bandwidth is not constrained. Infinite buffer exists in the edge servers, and at most one computing task is served by the edge server simultaneously. Then the latency components could be calculated as (1)–(4).

$$D_{Local}(i) = \frac{C_i}{f_i^l} \tag{1}$$

$$D_{Remote}(i) = \frac{C_i}{f_r^l} \tag{2}$$

$$T_{Trans}(i) = \frac{D_i}{r_i} \tag{3}$$

$$D_{Result}(i) = \frac{D_i^r}{r_i} = \frac{K_i C_i}{r_i} \tag{4}$$

where r_i represents the transmission rate between the local node and the edge server node (either uplink or downlink), the unit is Kbps; D_i^r represents the size of computation result for type-i task. In general, its size is proportional to the amount of computation data C_i, the proportion factor is a constant K_i.

The queue delay D_{queue} is estimated by multiple class M/D/1 queue theory. Here we consider two typical cases: FIFO with equal priority (EP in short) and Non-preemptive priority queue (NPP in short). Denoted the arrival rate of all type-i tasks by λ_i, which is usually a known parameter at the domain controller. Therefore, the real arrival rate of type-i tasks at the edge server is $\xi_i\lambda_i$.

The mean service time of type-i $E(S_i)$ is deterministic, and equals to $D_{queue}(i)$. We define the probability that the server is busy and busy with a type-i task as ρ and ρ_i respectively. Obviously

$$\rho_i = \xi_i\lambda_i E(S_i) = \frac{\xi_i\lambda_i C_i}{f_i^r} \tag{5}$$

$$\rho = \sum_{i\in I}\rho_i = \sum_{i\in I}\frac{\xi_i\lambda_i C_i}{f_i^r} \tag{6}$$

$$E(S) = \sum_{i\in I}\frac{\xi_i\lambda_i}{\sum_{i\in I}\xi_i\lambda_i}E(S_i) = \sum_{i\in I}\frac{\xi_i\lambda_i}{\sum_{i\in I}\xi_i\lambda_i}\frac{C_i}{f_i^r} \tag{7}$$

Case A: Equal Priority (EP):

Based on Little theory and PASTA (Poisson arrivals see time averages) property, the average queueing latency is equal to the average queueing latency of each class, which could be estimated by

$$E(D_{queue}) = \frac{\sum_{i\in I}\rho_i\frac{E(S_i)}{2}}{1-\rho} = \frac{\sum_{i\in I}\frac{\xi_i\lambda_i(C_i)^2}{2(f_i^r)^2}}{1-\sum_{i\in I}\frac{\xi_i\lambda_i C_i}{f_i^r}} \tag{8}$$

Case B: Non-preemptive priority (NPP)

If type 1 has non-preemptive priority over type 2, then a type 2 task cannot be preempted once it enters service. Type 1 task still have priority over any type 2 task that are waiting but not being served. Let us assume that if $i < j$, then type i has non-preemptive priority over type j.

For type 1 task,

$$E(D_{Queue(1)}) = \frac{\sum_{i\in I}\rho_i\frac{E(S_i)}{2}}{1-\rho_1} = \frac{\sum_{i\in I}\frac{\xi_i\lambda_i(C_i)^2}{2*(f_i^r)^2}}{1-\frac{\xi_1\lambda_1 C_1}{f_1^r}} \tag{9}$$

For type $i > 1$, again using little and PASTA,

$$\begin{aligned} E(D_{Queue}(i)) &= \frac{\sum_{i\in I}\rho_i\frac{E(S_i)}{2}}{(1-\sum_{k=1}^{i-1}\rho_k)(1-\sum_{k=1}^{i}\rho_k)} \\ &= \frac{\sum_{i\in I}\rho_i\frac{E(S_i)}{2}}{(1-\sum_{k=1}^{i-1}\frac{\xi_k\lambda_k C_k}{f_k^r})(1-\sum_{k=1}^{i}\frac{\xi_k\lambda_k C_k}{f_k^r})} \end{aligned} \tag{10}$$

Therefore, the average computing latency for each type of task could be obtained as:

$$E(L_i) = (1-\xi_i)D_{Local}(i) + \\ \xi_i(D_{Remote}(i) + T_{Data}(i) + E(D_{Queue}(i)) + T_{Result}(i)) \tag{11}$$

4.2 Offloading Policy

The optimization goal is minimizing the total computing latency of all computing units, including the edge offloading units and local computing units under the constraints of allowed maximum delay tolerant of each unit T_i, the optimization problem is mathematically modeled as:

$$\begin{aligned} &\min_{\xi_i} \sum_{i \in I} \pi_i E(L_i) \\ &\mathtt{s.t.} E(L_i) \leq T_i, 0 \leq \xi_i \leq 1, i \in I \end{aligned} \tag{12}$$

It is known from [8] that the optimization problem above is convex optimization. Thus, one can use the block coordinate descent (BCD) approach to deal with it as in the following iterative algorithm.

Algorithm 1 Proposed iterative algorithm based on BCD

1: Initiate : random choose $(\xi_i, i \in I)$
2: Repeat
3: for $i \in I$
4: update ξ_i with all ξ_j (for all $j \neq i$) fixed by
5: $\xi_i = \xi_i + \nabla_i \sum_{i \in I} \pi_i E(L_i)$
6: Until $|\sum_{\xi_i, i \in I} \pi_i E(L_i) - \sum_{\xi_i, i \in I} \pi_i E(L_i)| \leq \varepsilon$, or maximum number of iterations is reached.
7: End Repeat
8: Return $(\xi_i, i \in I)$

5 Simulation and Result Analysis

In this section, we use MATLAB simulation to evaluate the performance of proposed edge computation offload scheme. The computation tasks of field devices are classified into four types with the probabilities $\pi : \{0.1, 0.3, 0.4, 0.2\}$.

The incoming computing flow of each type obeys poisson distribution of parameter λ. Other parameters are listed in Table 1. In addition, the computational capability $f_i^l = 2\,\text{GHz}$, $f_i^r = 10\,\text{GHz}$, $r_i = 20\,\text{Mbps}$, $K_i = 10^{-5}$. The link bandwidth bottleneck and transmission error are ignored.

Table 1. Parameters of various computation tasks

Parameter	Value	Unit
D	{0.2, 0.5, 3, 6}	Mbits
C	$\{10^8, 2*10^8, 3*10^8, 5*10^8\}$	Cycles
T	{0.05, 0.1, 0.2, 0.4}	Seconds

Figure 3 evaluates the percentage of various types of the tasks that are offloaded under different computing task density λ in equal priority case. Because the tasks of higher priority are more sensitive to the delay constraints, the equal priority case cannot improve the probability of processing for high-priority tasks. We found that the offloading percentage of the type-1 and the type-4 tasks is nearly 0% and 100% in all the λ values, respectively. With the increase of task density, the offloading percentage decrease due to higher queuing latency at the edge server.

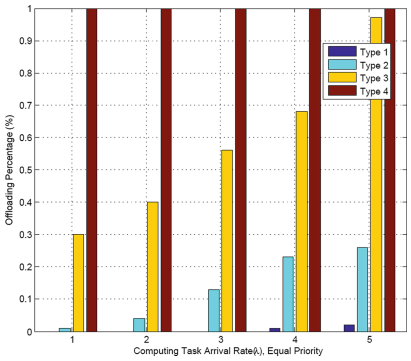

Fig. 3. EP offloading percentage

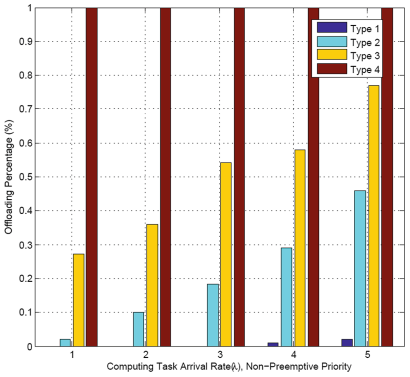

Fig. 4. NPP offloading percentage

Figure 4 shows the offload probability for each type of tasks under different computing task density λ in non-preemptive priority case. Compared with EP case, type-2 tasks will increase the offload probability slightly due to higher priority in high load region. Priority is given to high-priority tasks, which have less impact on the delay of subsequent tasks and easier to be flexibly choreographed.So Non-preemptive priority case can improve the service rate of high-priority tasks and further reduce average delay overall service.

Figure 5 depicts the average delay of proposed central offloading scheme, All-local computing and All-remote computing scheme in EP case. Figure 6 shows the probability of outage (The probability that the computing latency larger than the maximal allowed latency) of those schemes in EP case. The delay and outage probability prove the feasibility of proposed offloading scheme. With the growing

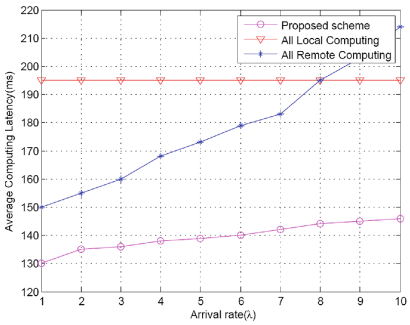

Fig. 5. Average delay

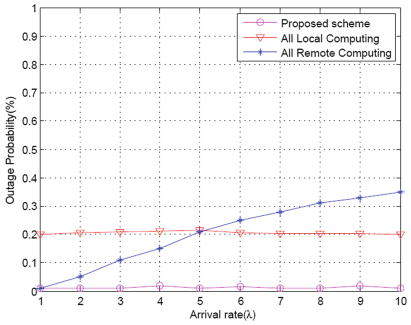

Fig. 6. Outage probability

of computing load, the performance of All-remote computing solution will grow worse due to the increasing queuing delay. The offloading scheme proposed in the paper can greatly reduce the computing latency and improve the computing QoS for different users.

6 Conclusion

In this paper, we propose a new SDIN architecture and a kind of centralized computing offloading strategies based on our SDIN architecture. The simulation results have indicated that the proposed SDIN architecture is feasible and effective in computing offloading. And to a certain extent, our architecture can provide traditional industries a better resource management solution and more flexible production scheme which means the production efficiency can possibly be improved.

References

1. Schweissguth, E., Danielis, P., Niemann, C., Timmermann, D.: Application-aware industrial ethernet based on an SDN-supported TDMA approach. In: 2016 IEEE World Conference on Factory Communication Systems (WFCS), Aveiro, Portugal (2016)
2. Aggarwal, C., Srivastava, K.: Securing IoT devices using SDN and edge computing. In: 2016 2nd International Conference on Next Generation Computing Technologies (NGCT), Dehradun, India (2016)
3. Sun, X., Ansari, N.: EdgeIoT: mobile edge computing for the internet of things. IEEE Commun. Mag. **54**(12), 22–29 (2016)
4. Dama, S., Pasca, T.V., Sathya, V.: A feasible cellular internet of things enabling edge computing and the IoT in dense futuristic cellular networks. IEEE Consum. Electron. Mag. **6**(1), 66–72 (2017)
5. Pengfei, H., Ning, H., Qiu, T.: Fog computing based face identification and resolution scheme in internet of things. IEEE Trans. Ind. Inf. **13**(4), 1910–1920 (2017)

6. Li, D., Zhou, M.-T., Zeng, P.: Green and reliable software defined industrial network. IEEE Commun. Mag. **54**(10), 30–37 (2016)
7. Zhao, P., Tian, H., Qin, C., Nie, G.: Energy-saving offloading by jointly allocating radio and computational resources for mobile edge computing. IEEE Access **5**, 11255–11268 (2017)
8. Miettinen, A.P., Nurminen, J.K.: Energy efficiency of mobile clients in cloud computing. HotCloud **10**, 4–4 (2010)
9. Li, M., Richard Yu, F., Si, P., Yao, H.: Energy-efficient M2M communications with mobile edge computing in virtualized cellular networks. In: 2017 IEEE International Conference on Communications (ICC), Paris, France (2017)