IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2017.DOI

Software Defined Mission-Critical Wireless Sensor Network: Architecture and Edge Offloading Strategy

FANGMIN XU[1], HUANYU YE [2], FAN YANG[3],AND CHENGLIN ZHAO[4] ,

[1]Key Laboratory of Universal Wireless Communications Ministry of Education, Beijing University of Posts and Telecommunications (e-mail: xufm@bupt.edu.cn)
[2]mcxd_799845315@bupt.edu.cn
[3]2013213320@bupt.edu.cn
[4]clzhao@bupt.edu.cn

Corresponding author: Fangmin Xu (e-mail: xufm@bupt.edu.cn).

ABSTRACT Mission Critical Wireless Sensor Network are attractive for information gathering in various complex environments and support many mission-critical applications such as industrial automation, security surveillance. However, in order to fully exploit these networks for such applications, agile and scalable network management for data transfer and computing task implementation are essential.Thus, in this paper, we propose a software defined mission-critical wireless sensor network (MC-SDWSN) which can solve the existing challenging issues in tradition WSN such as resource utilization, data processing, system compatibility and strict latency requirement. The architecture is based on the idea of SDN architecture, combining the hierarchical cloud and edge computing technologies. Based on the MC-SDWSN architecture, a novel centralized computation offload strategy in sensor network application are proposed to show the feasibility. The simulation results confirm the MC-SDWSN architecture and the edge offloading strategy could support the critical missions effectively.

INDEX TERMS Mission Critical Wireless Sensor Network; Software Defined Network; Edge Computing; Computing Offloading

I. INTRODUCTION

WIRELESS Sensor network (WSN) is a distributed intelligent network system that can complete assigned task according to the environment, which is composed of a large number of small sensor nodes with wireless communication and computing capability deployed in certain region. WSN could be used in traffic management, environmental monitoring, medical and health care, smart city areas. It could also be used in some application areas with critical missions such as military, disaster early warning, industrial manufacturing and emergency rescue areas.

Mission-critical WSN (MC-WSN) applications are defined as applications demanding data delivery bounds in the time and reliability domains [1]. In most of the application areas, sensors need to support variety critical missions such as battlefield environment detection, fire alarm and so on. To support MC-WSN applications, WSN tasks execution must be flexible with limited bandwidth and energy. Moreover, WSN must have high time efficiency that could handle critical missions in time. Finally, it should also be security enough to defense aggression.

Thus, traditional MC-WSN is still facing following challenges:

- Stringent low latency and high reliability guarantee. Ensuring the reliable and low-latency information transmission is important to achieve the reliability and efficiency of MC-WSN. Traditional WSN use flexible scheme such as CSMA(Carrier Sense Multiple Access). However, deterministic routing and scheduling schemes like TDMA(Time Division Multiple Access) are recommended in current MC-WSN standard.
- Low energy consumption and computation capability
  Generally, constrained by the limited physical size, sensor nodes have limited battery energy supply. Therefore, their memories, computational and communication capabilities are restricted. However, some critical missions need complex computation capability, this brings a challenge to MC-WSN.
- Scalability
  As the number of connected sensors increases, the large

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/ACCESS.2019.2890854, IEEE Access

amount of sensing data and the management of network entities rises as a significantly challenge to network administrator.
- Security issue
  The ensure of data confidentiality, authenticity, and integrity is essential to MC-WSN. With distributed sensor nodes attached to the network, the design of networking protocol should make the communication safe from external attacks and intrusion.

Therefore, Software Defined Mission-Critical Wireless Sensor Networking (MC-SDWSN) is introduced which is the combination of Software Defined Networking(SDN) and MC-WSN. Wireless sensor network control and management functions are logically centralized in MC-SDWSN to bring higher processing and operation performance. It could avoid most of the before-mentioned drawbacks. Specially, our contributions are:

- Propose a new Software Defined Mission-Critical Wireless Sensor Network (MC-SDWSN) architecture for MC-WSN, and analyze the control and management process by the example application of edge computation.
- Based on the MC-SDWSN architecture and the characteristics of computing tasks, we propose a centralized edge computation offloading strategy.

The rest of the paper is organized as follows. Section II introduces the architecture of MC-SDWSN, its advantages and related works. Section III analyses the application of MC-SDWSN in solving the computing offload problem and the unique features of typical missions. The system model, problem formulation and solutions are given in Section IV. We present the performance simulation in Section V. Finally, Section VI concludes the work.

II. SOFTWARE DEFINED MISSION-CRITICAL WIRELESS SENSOR NETWORKING

A. THE ADVANTAGE OF MC-SDWSN

SDN is an emerging approach of networking paradigm that allows directly programmable network control. SDN makes basic infrastructures transparent to user by separating the control plane and data plane. It provides promising solutions to the problems of traditional WSN. There are some emerging architectures of SDN in the field of WSN. In part of the architecture, all sensor nodes have SDN functions, the flow table is strong to support routings in order to reduce the communications between controllers and sensors [2][3]. Some other research proposed the architecture that only a few sensor nodes have SDN function, and regular sensor nodes only run minimum transmission energy routing (MTE) [4][5]. Compared with traditional WSN, SDN based WSN are more energy efficiency, long lifetime and have higher performance. Most of the previous architecture focus more on energy saving but ignore the latency and security. However, in critical missions, latency and security are also important indicators. Therefore, this paper proposed a MC-SDWSN architecture and edge offloading strategy. It considers the requirements of MC-WSN, then combines the superiority of SDN and MEC with MC-WSN. Thus, it has following advantages:

- **Dynamic Resource management**
  The controller has a centralized control function to manage the distributed computing, storage and communication resource. As it is shown in later section, the distributed computing resource (including the edge computing resource and the local resource) in the network is globally managed by the central controller.
- **Fast deployment**
  SDN helps wireless sensors accelerate application deployment and delivery, dramatically reducing costs by providing automated, on-demand application delivery and mobility at any scale.
- **Abnormal message handling**
  In SDN structure, the abnormal message will trigger Packet_In event and an asynchronous message will be sent to the controller where the message will be treated by specific policies. Based on the centralized control plane in SDN, the flow table statistic is collected from the controller online, and the abnormal traffic is detected.
- **Reliable Security performance**
  Under the SDN architecture, the security policy configuration of 2-7 layers can be realized through the controller, providing more fine-grained security control. Network Managers can make changes effectively by quickly limiting and viewing capabilities within the network from a central point of view, and perform traffic shaping and packet QoS in a more secure manner. Any message which accords to OpenFlow protocol goes through the controller is using an TLS encrypted channel.

MC-SDWSN has not been confirmed can handle all the challenges that mentioned, but its abilities of powerful centralized management, concurrent data collection and high processing performance indicate that MC-SDWSN has great potential and value in the construction of MC-WSN.

B. RELATED WORKS

Now, using SDN architecture in wireless sensor network has been attracting significant attention in recent years. And there have been several works about SDWSN [6-11]. In [6], authors propose a low latency mobile edge computing framework base on SDN architecture. A general architecture of SDWSN has been designed in [4], it uses MTE routing protocol, with part of SDWSN deployed. Security and privacy are two of the main challenges to the IoT, so that [11] has proposed a few solutions and models for securing the IoT devices and the communications using SDN architecture. Meanwhile, the performance of SDWSN using in data offloading and edge computation in several scenarios such as cloud server and mobile tasks have discussed in [7][8].

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/ACCESS.2019.2890854, IEEE Access

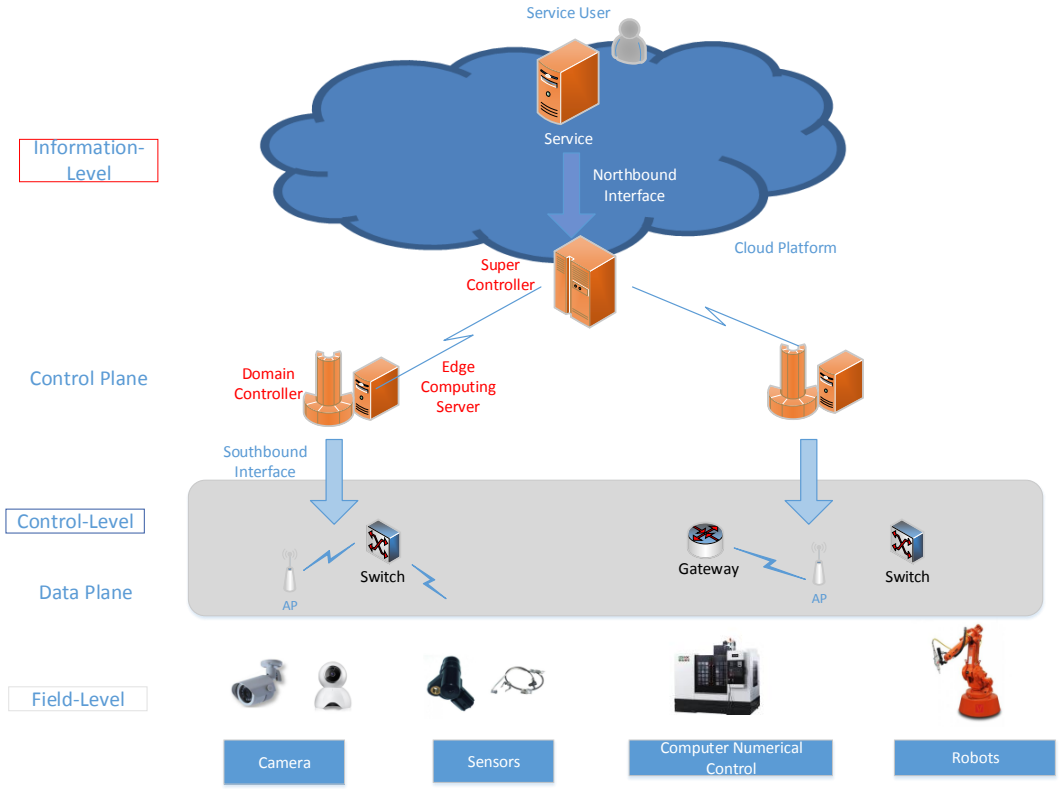

FIGURE 1: Architecture of Software Defined Mission Critical Sensor Network

Nowadays, most of the SDWSN researches are using the controller for resource management or big data analysis by combined with cloud computing. As for the computation offloading technology, Authors improves the computing time delay and reduces energy consumption of face recognition applications by uses the architecture [9]. In [6] and [10], Mobile devices can extend the standby time by computation offloading. Many previous researches paid more attention to the special purposes such as energy saving, but did not focus on the detail of the whole system architecture and operation.

Traditional WSN network use routing protocols such as Flooding, SPIN, HREEMR, LEACH. However, most of the protocols cannot support critical missions. Thus, in [11] a 2-hop neighborhood information based cover set selection to determine the most relevant cover sets was proposed to serve MC-WSN which could enable fast packet delivery and high transmit quality. In [12], a platform which benefits from both static and mobile sensors with low latency is proposed.

However, above protocols have no global view of the network and cannot realize the joint optimization of bandwidth, latency, energy and reliable. Moreover, security was considered incomplete. These cannot meet the requirement of MC-WSN.

C. MC-SDWSN ARCHITECTURE

Since MC-SDWSN has many attractive advantages that have been mentioned before, we propose a MC-SDWSN architecture as shown in Fig.1. On a top-down view, the MC-SDWSN architecture contains four layers:Field Device, Data transport plane, Distributed MC-SDWSN control plane and Cloud platform. Field devices include the basic devices such as sensors, actuator, cameras, etc.

1) Data transport plane

Data transport plane includes switches, wireless access points and gateways. All nodes could be controlled by receiving the commands(such as computing decision in the following computing offloading decision algorithm) from the control plane through southbound interface.

2) Distributed MC-SDWSN control plane

The distributed SDN controllers are responsible for centralized management of the data plane devices and edge computation servers. Controllers receive the requests from devices in data transport plane and run the offloading decision algorithm by considering both the mission requirements and the status of edge computation servers. As given in Fig.1, two-tier heterogeneous controller structure (domain controller and super controller) is one of the deployment solutions in MC-WSN.

3) Cloud platform

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/ACCESS.2019.2890854, IEEE Access

The cloud platform has series of cloud service applications that are composed of a bundle of application systems. The super controller lays more emphasis on resource management and the authorization of distributed SDN controllers.

Besides, edge computing servers are more powerful than the computing capability of local sensor nodes in MC-WSN. Due to the edge servers are located near the sensor network, it can provide lower and more stable latency than the cloud computing server. The offloading of computing task from local sensor network and cloud computing platform to edge servers is named as computing offloading in MC-WSN.

III. COMPUTING TASK AND OFFLOAD IN MISSION CRITICAL SCENARIO

In the concept of Mission critical network, there are lots of computing tasks during the operation, such as Automatic Guided Vehicle (AGV) navigation, operation control of mechanical arms, identify of fire alarm and so on. These services and applications may require significant computation resources and constrained time delays. However, the computational capabilities of the field devices are limited due to cost and size limitation. To meet the increasing computation demand of these applications, offloading the computation tasks to edge computing servers is an appealing idea, which is also named as edge computing offload.

A. CRITICAL MISSION COMPUTING TASK

After investigation, the critical mission computing task has following unique features:

1) Stringent computing delay tolerant

In MC-WSN, the distributed sensors, controllers, and edge computing devices need to collaborate together to achieve real-time operation or complete the critical missions. In order to minimize the influence to the sensor network, the latency-sensitive missions require delay from tens of milliseconds to hundreds of milliseconds.

2) Diverse computing factors

As mentioned before, typical computing tasks in MC-WSN could be classified into following types with diverse characteristics and QoS requirements.

- Data Preprocessing (such as sampling or averaging the data to reduce the processing pressure to the upper data center)
  Characteristic: continuous periodic incoming data packets with little size, small computing resource required.
- Image or video recognition (such as fire hazard monitor, invasion detection)
  Characteristic: huge amounts of raw input data, small size of computing result data.
- Localization and mapping (such as enemy positioning on the battlefield)
  Characteristic: high computing accuracy, huge computing resource required.

We consider the case where the network supports two types of computing tasks, which is defined as critical computing tasks (P_c) and normal computing tasks (P_n). P_c represents the time-critical computing task with emergency nature such as fire alarm image identify and emergency shutdown. This type of task is always critical and is bounded by strict deadlines. P_n task refers to non-critical periodic traffic such as periodic monitoring.

To study the effects of the computation task characteristics on the design of offloading schemes, we classify the normal

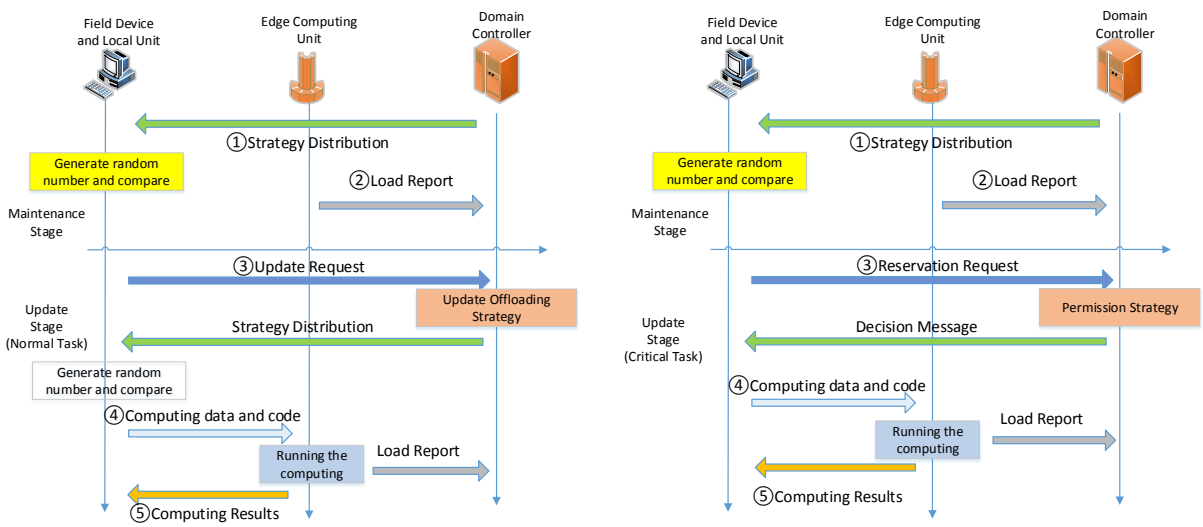

FIGURE 2: Computing Decision Progress Based on MC-SDWSN
(Left: Fig2(a) Non-critical Periodic traffic; Right: Fig2(b) Critical Mission traffic).

computing tasks into I types. Each type of normal computing task has various QoS requirements. According to normal computation task types, the sensors can be correspondingly classified into I types. Let us assume that one kind of sensors only generates one type of normal computing task.

3) Different task pattern

Generally, the computing task originate from the non-critical periodic traffic has a fixed cycle. To simplify the analysis, the arrival of single normal computing task P_n is modeled as regular arrival. However, considering the asynchronism of non-critical traffics, the arrival of normal tasks at the edge server could be treated as poisson flow.

B. COMPUTING OFFLOAD PROCEDURE BASED ON THE MC-SDWSN ARCHITECTURE

The decision of whether local computing or offloading the computing task to the edge server is an important and difficult procedure in MC-SDWSN. Generally, the computing latency in remote edge side is much shorter than that of local computing. However, the transmission latency and queue delay at the edge server will have great influence the offloading latency, which is vital performance factor for critical missions.

Fig.2 gives the computing decision process based on the MC-SDWSN architecture. 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 $\xi_i, i \in \mathcal{I} = [1, 2, ...I]$, to field devices (① in Fig.2). Field devices(type i) generate a random number ς 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.

$$Decision = \begin{cases} 1, if \varsigma < \xi_i \\ 0, otherwise \end{cases} \tag{1}$$

Update stage is trigged by the change of environment and other factors, such as the increasing of computing task frequency, adjust of task scheduling, triggering of emerging critical mission. It contains three parts: requests collection, mode decision, computing, as follows:

- Update Request
  The devices collect update data (information of computing tasks) and send it to the data transport plane through the access points. Then the update request message is send to its domain SDN controller by southbound interface (③ in Fig.2(a)).
  For critical task P_c, as in the Fig.2(b) device firstly transmits a reservation request message to the controller by the control channel. The reservation request message contains the type of the computing task, a payload which represents the start time of the task, a binary coded payload which corresponds to the decimal value d_i of the expire time (in ms) of the task. The range of of the d_i values is determined according to the critical application in the environment.
- Mode decision
  The domain controller exacts the computing capability parameters from the latest update request message / reservation request message and edge computing server (from the load report message, as② in Fig.2), then it will run the decision algorithm described in Section IV and return the domain offloading strategy. For normal task, the offloading strategy means the offloading probability of each computing task type, ξ_i. For critical task, the strategy is the permission or rejection message of the computing offloading.
- Computing and feedback
  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).

IV. SYSTEM MODEL AND OFFLOADING ALGORITHM

Considering most of devices in wireless sensor network are powered with batteries, generally the energy consumption of local computing is of the same order as the consumption in data transfer procedure in edge computing. And it is assumed that the edge computing servers have stable power supply, thus the energy consumption of edge computing server in the computing procedure could be ignored. Therefore, the designed computing offload policy only considers the goal of minimizing computing delay.

A. SYSTEM MODEL

For the computing latency, it is divided into the following five aspects for decision analysis: local computing delay D_{Local}, data transmission delay from the local device to the edge server T_{Data}, the task queuing delay in edge computing server D_{Queue}, edge computing server processing delay D_{Remote} and computing result transmission delay from edge server to local devices T_{Result} .

We represent the computing tasks of type i by I_i=(D_i, C_i, T_i), where D_i represents the size of computation data involved in the type-i task, that is the total number of bits. C_i represents 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 normal computation task types, the proportion of the field devices with type-i tasks is given by π_i, where $\sum_{i \in \mathcal{I}} \pi_i = 1$.

The local computing unit capability is defined as f_i^l, and f^r represents the CPU computation cycles per second that the edge server can provide. We assume that 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 (2) - (5).

$$D_{Local}(i) = \frac{C_i}{f_i^l} \tag{2}$$

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/ACCESS.2019.2890854, IEEE Access

$$D_{Remote}(i) = \frac{C_i}{f^r} \tag{3}$$

$$T_{Data}(i) = \frac{D_i}{r_i} \tag{4}$$

$$T_{Result}(i) = \frac{D_i^r}{r_i} = \frac{K_i * D_i}{r_i} \tag{5}$$

$$r_i = Wlog(1 + \frac{ph_i}{WN_0}) \tag{6}$$

where r_i represents the transmission rate between the local node and the edge server node (either uplink or downlink), the unit is bps; r_i is related with channel bandwidth W, transmission power p and signal-to-noise ratio (SNR) of the channel $\frac{ph_i}{WN_0}$; h_i represents the gain of the channel from local node i to edge server, N_0 represents the noise power spectral density in the channel. 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.

Octets:1	3	3
Task Type	Start Time	Expire Time

FIGURE 3: Format of Reservation Request Packet

As shown in Fig.3,reservation request packet contains a binary coded payload, which corresponds to the task expire time (in ms) of the P_c task i, denoted by decimal value d_i.

For the critical task, the controller will always insert the incoming critical task into the header of the computing queue to guarantee the computing task will be implemented immediately. In the case that multiple critical tasks arrival at the edge computing server, the Earlier Expire Time(EET) scheduling will be used to schedule the critical tasks. The task with earliest expire time will be processed firstly. Such that, the node with the lowest d_i will gain the highest priority in the EET schedule and immediately access the computing resource. This is to prioritize the most urgent task to complete its computation within its deadline bound.

B. LATENCY ANALYSIS OF NORMAL COMPUTING TASK

For the normal task, without considering the influence of critical task, the queue delay D_{queue} is estimated by multiple class M/D/1 queue theory. FIFO(First In, First Out) scheduling will be used to schedule normal tasks. 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 normal tasks by λ_i, which usually could be estimated at the domain controller by historical data. Therefore, the real arrival rate of type-i normal tasks at the edge server is $\xi_i\lambda_i$.

Assuming at the schedule time, the set of accepted critical missions that has not been served at the domain controller is represented by $\mathcal{J}$. The service time of critical mission j is calculated as Equation (3).

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

$$\rho_i = \xi_i\lambda_i E(S_i) = \frac{\xi_i\lambda_i C_i}{f^r} \tag{7}$$

$$\rho = \sum_{i\in\mathcal{I}} \rho_i = \sum_{i\in\mathcal{I}} \frac{\xi_i\lambda_i C_i}{f^r} \tag{8}$$

$$E(S) = \sum_{i\in\mathcal{I}} \frac{\xi_i\lambda_i}{\sum_{i\in\mathcal{I}}\xi_i\lambda_i} E(S_i) = \sum_{i\in\mathcal{I}} \frac{\xi_i\lambda_i}{\sum_{i\in\mathcal{I}}\xi_i\lambda_i} \frac{C_i}{f^r} \tag{9}$$

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

$$\begin{aligned} E(D_{Queue}) =& \frac{\sum_{i\in\mathcal{I}} \rho_i \frac{E(S_i)}{2}}{1-\rho} + \sum_{j\in\mathcal{J}} \frac{C_j}{f^r} \\ =& \frac{\sum_{i\in\mathcal{I}} \frac{\xi_i\lambda_i C_i^2}{2*(f_i^r)^2}}{1-\sum_{i\in\mathcal{I}} \frac{\xi_i\lambda_i C_i}{f^r}} + \sum_{j\in\mathcal{J}} \frac{C_j}{f^r} \end{aligned} \tag{10}$$

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,

$$\begin{aligned} E(D_{Queue(1)}) =& \frac{\sum_{i\in\mathcal{I}} \rho_i \frac{E(S_i)}{2}}{1-\rho_1} + \sum_{j\in\mathcal{J}} \frac{C_j}{f^r} \\ =& \frac{\sum_{i\in\mathcal{I}} \frac{\xi_i\lambda_i C_i^2}{2*(f^r)^2}}{1-\frac{\xi_1\lambda_1 C_1}{f^r}} + \sum_{j\in\mathcal{J}} \frac{C_j}{f^r} \end{aligned} \tag{11}$$

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

$$\begin{aligned} E(D_{Queue}(i)) =& \frac{\sum_{i\in\mathcal{I}} \rho_i \frac{E(S_i)}{2}}{(1-\sum_{k=1}^{i-1}\rho_k)(1-\sum_{k=1}^{i}\rho_k)} + \sum_{j\in\mathcal{J}} \frac{C_j}{f^r} \\ =& \frac{\sum_{i\in\mathcal{I}} \rho_i \frac{E(S_i)}{2}}{(1-\sum_{k=1}^{i-1}\frac{\xi_k\lambda_k C_k}{f^r})(1-\sum_{k=1}^{i}\frac{\xi_k\lambda_k C_k}{f^r})} \\ &+ \sum_{j\in\mathcal{J}} \frac{C_j}{f^r} \end{aligned} \tag{12}$$

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

$$\begin{aligned} E(L_i) =& (1-\xi_i)D_{Local}(i) + \xi_i(D_{Remote}(i) + T_{Data}(i) \\ &+ E(D_{Queue}(i)) + T_{Result}(i)) \end{aligned} \tag{13}$$

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/ACCESS.2019.2890854, IEEE Access

C. OFFLOADING POLICY OF CRITICAL COMPUTING TASK

The computing latency for each critical computing task could be estimated by the controller. Similar with the normal computing task case, at that moment (t) that reservation request message of critical computing task j arrived at the controller, the set of critical computing tasks that has lower expire time than task j is denoted by $\mathcal{K}$, the remaining processing time for normal computing task that is running in the edge computing sever is represented by $R(t)$. the queue delay D_{queue} of critical computing task j is estimated by

$$D_{queue}(j) = \sum_{k \in \mathcal{K}} \frac{C_k}{f^r} + R(t) \tag{14}$$

Following algorithm is implemented at the controller to make the offloading decision for critical mission when new reservation request message arrived at the controller. The output $Decision_j$ is contained in the computing offloading permission or rejection message.

Algorithm 1 Offloading strategy for critical mission

Input : $C_k, f^r, f_j^l, R(t), d_j$
Initiate : $Decision_j = 0$
Calculate: $D_{LocalDelay}(j) = \frac{C_j}{f_j^l}$
Calculate: $D_{EdgeDelay}(j) = \sum_{k \in \mathcal{K}} \frac{C_k}{f^r} + T_{Data}(j) + T_{Result}(j) + R(t)$
if $D_{EdgeDelay}(j) < D(LocalDelay) \vee D_{EdgeDelay}(j) \leq d_j$ **then**
  $Decision_j = 1$
end if
Output $Decision_j$

D. OFFLOADING POLICY FOR NORMAL COMPUTING TASK

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:

$$\begin{aligned} O_1 : & \min_{\xi_i} \sum_{i \in \mathcal{I}} \pi_i E(L_i) \\ & \text{s.t.} E(L_i) \leqslant T_i, 0 \leqslant \xi_i \leqslant 1, i \in \mathcal{I} \end{aligned} \tag{15}$$

It is known from [9] 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,where ε is the predefined iteration termination value.

Algorithm 2 Proposed iterative algorithm based on BCD

Initiate : random choose $(\xi_i, i \in \mathcal{I})$, $k = 0$
Repeat $k = k + 1$
  for $i \in \mathcal{I}$
  update ξ_i^k with all ξ_j^k (for all $j \neq i$) fixed by $\xi_i^k = \xi_i^{k-1} + \nabla_i \sum_{i \in \mathcal{I}} \pi_i E(L_i)$
Until $|\sum_{\xi_i^k, i \in \mathcal{I}} \pi_i E(L_i) - \sum_{\xi_i^{k-1}, i \in \mathcal{I}} \pi_i E(L_i)| \leqslant \varepsilon$, or maximum number of iterations is reached.
End Repeat
Output $(\xi_i^k, i \in \mathcal{I})$

V. SIMULATION AND RESULTS 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 $\pi_i : \{0.1, 0.3, 0.4, 0.2\}$.

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

TABLE 1: Parameters of various computation tasks

Parameter	Critical Computing Task Value	Normal Computing Task Value	Unit
D	$\{0.1\}$	$\{0.2, 0.5, 3, 6\}$	Mbits
C	$\{6 * 10^6\}$	$\{10^8, 2 * 10^8, 3 * 10^8, 5 * 10^8\}$	Cycles
T	$\{0.025\}$	$\{0.05, 0.1, 0.2, 0.4\}$	Seconds

The arrival of critical computing task are modelled as beta distribution with parameter(α=3,β=4) in the duration of $[3-4]$ and $[7-8]$seconds. The total simulation time is 10 seconds.

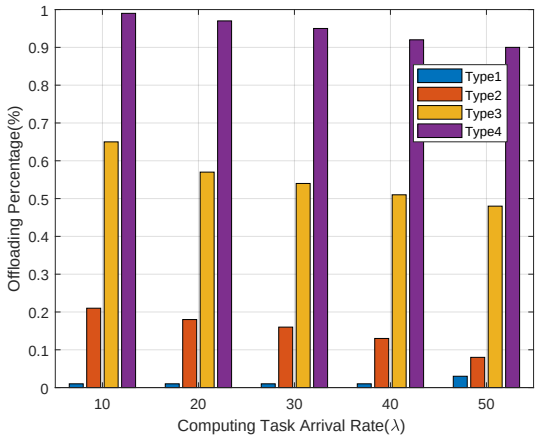

FIGURE 4: The offloading percentage of Periodic Computing tasks(EP case)

Fig.4 evaluates the percentage of various types of the tasks that are off-loaded under different computing task density λ in equal priority case. 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.

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/ACCESS.2019.2890854, IEEE Access

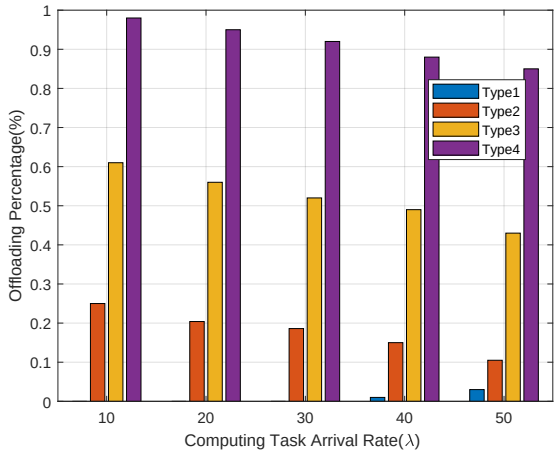

FIGURE 5: The offloading percentage of Periodic Computing tasks(NEP case)

Fig.5 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.

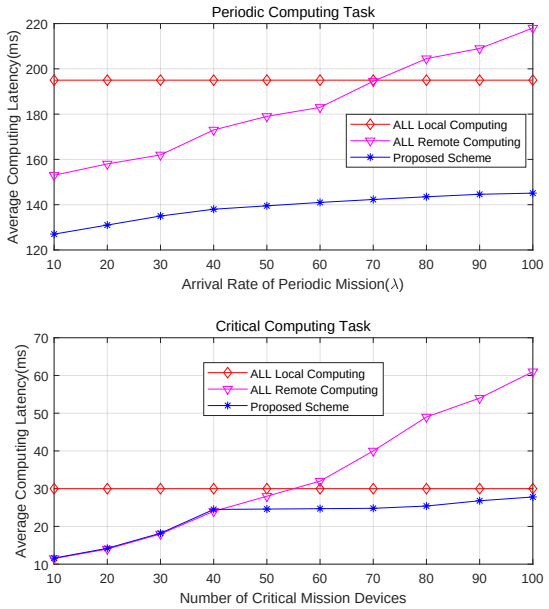

FIGURE 6: Average Computing Latency of different computing schemes(Above:6a, Below:6b)

Fig.6(a) depicts the average delay of periodic missions of proposed central offloading scheme, All-local computing and All-remote computing scheme in EP case. In the case where arrival rate of periodic mission is 10, Fig.6(b) shows the average delay of critical missions by changing the number of critical mission devices. Fig.7 shows the probability of outage (The probability that the computing latency larger

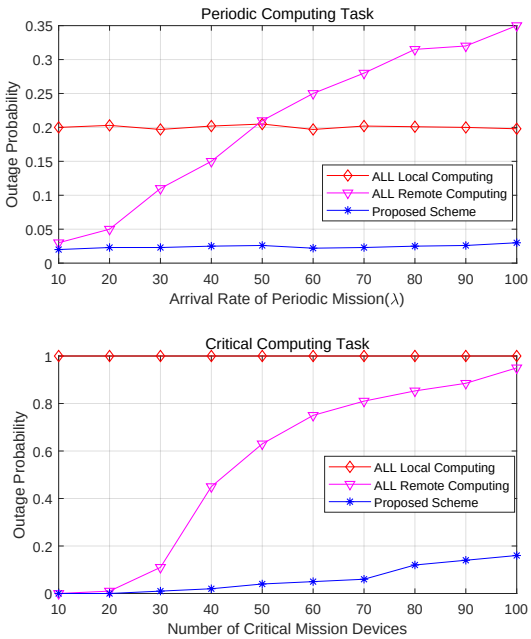

FIGURE 7: Outage Probabilities of different computing schemes(Above:7a, Below:7b)

than the maximal allowed latency) of those schemes in EP case.Fig.7(a) depicts the outage probabilities of periodic missions.In the case where arrival rate of periodic mission is 10, Fig.7(b) shows the outage probabilities of critical missions by changing the number of critical mission devices. The delay and outage probability prove the feasibility of proposed offloading scheme. With the growing 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 critical mission tasks and periodic tasks.

CONCLUSION

The area of Mission critical wireless sensor network has drawn many attention from the research community, especially the control and resource management architecture and methods. The mission-critical applications demand data delivery and computing implementation bounds in both the time and reliability.

In this paper, we propose a SDN based Mission critical wireless sensor network architecture and a kind of centralized computing resource management strategy. The strategy provide timely and reliable computing and communication resource for mission-critical applications.

Simulation results showed that it can achieve better performance in terms of delay guarantee which indicated the feasible and effective of the proposed architecture.

Future work will include the investigation of how the proposed architecture can be extended for more complex

scenarios.

COMPETING INTERESTS

The authors declare that they have no competing interests.

ACKNOWLEDGMENT

This work is supported by the Key Program of the National Natural Science Foundation of China (Grant No 61431008).

REFERENCES

[1] P. Suriyachai, U. Roedig and A. Scott, "A Survey of MAC Protocols for Mission-Critical Applications in Wireless Sensor Networks," in IEEE Communications Surveys and Tutorials, vol. 14, no. 2, pp.240-264, Second Quarter 2012.

[2] T. Luo, H. Tan and T. Q. S. Quek, "Sensor OpenFlow: Enabling Software-Defined Wireless Sensor Networks," in IEEE Communications Letters, vol. 16, no. 11, pp. 1896-1899, November 2012.

[3] J. Zhou, H. Jiang, J. Wu, L. Wu, C. Zhu and W. Li, "SDN-Based Application Framework for Wireless Sensor and Actor Networks," in IEEE Access, vol. 4, pp. 1583-1594, 2016.

[4] S. Tomovic and I. Radusinovic, "Performance analysis of a new SDN-based WSN architecture," 2015 23rd Telecommunications Forum Telfor (TELFOR), Belgrade, 2015, pp. 99-102.

[5] L. Galluccio, S. Milardo, G. Morabito and S. Palazzo, "SDN-WISE: Design, prototyping and experimentation of a stateful SDN solution for WIreless SEnsor networks," 2015 IEEE Conference on Computer Communications (INFOCOM), Kowloon, 2015, pp.513-521.

[6] Mustafa Y. Arslan, Karthikeyan Sundaresan, Sampath Rangarajan. Software-Defined Networking in Cellular Radio Access Networks: Potential and Challenges. IEEE Communications Magazine, January 2015. Vol 53, Issue 1, pp: 150-156.

[7] Chengchao Liang, F. Richard Yu. Wireless Network Virtualization: A Survey, Some Research Issues and Challenges. IEEE Communications Surveys & Tutorials,Vol 17,Issue 1,2015,pp: 358-380.

[8] Kok-Kiong Yap, Masayoshi Kobayashi, Rob Sherwood, Nikhil Handigol, Te-Yuan Huang, Michael Chan and Nick McKeown. OpenRoads: Empowering Research in Mobile Networks. ACM SIGCOMM, Barcelona, Spain, August 2009.

[9] Aditya Gudipati, Daniel Perry, Li Erran Li, Sachin Katti. SoftRAN: Software Defined Radio Access Network. HotSDN 2013.

[10] Bernardos, C.J.;De La Oliva, A.;Serrano, P.;Banchs, A.;Contreras, L.M.;Hao Jin;Zu?niga, J.C. An Architecture for software defined wireless networking. IEEE Wireless Communications. June 2014.Vol 21, Issue 3, pp: 56-61.

[11] M. Diop, C. Pham and O. Thiare, 2-hop neighborhood information for cover set selection in mission-critical surveillance with Wireless Image Sensor Networks, 2013 IFIP Wireless Days (WD), Valencia, 2013, pp. 1-7.

[12] Erman A T, van Hoesel L, Havinga P, et al. Enabling mobility in heterogeneous wireless sensor networks cooperating with UAVs for mission-critical management[J]. IEEE Wireless Communications, 2008, 15(6): 38-46.

[13] Kostas Pentikousis, Yan Wang, Weihua Hu. MobileFlow: Toward Software-Defined Mobile Networks. IEEE Communications Magazine, Vol 51 ,Issue 7, pp: 44-53.

[14] Tao Chen;Honggang Zhang;Xianfu Chen;Tirkkonen, O. SoftMobile: Control Evolution for Future Heterogeneous Mobile Networks. IEEE Wireless Communications, Vol 21, Issue 6, pp 70-78.

[15] Rui Wang, Honglin Hu and Xiumei Yang. Potentials and Challenges of C-RAN Supporting Multi-RATs Toward 5G Mobile Networks. IEEE Access.Vol 2, 2014, pp: 1187-1195.

[16] Imran, A.;Zoha, A.Challenges in 5G: How to Empower SON with Big Data for Enabling 5G. IEEE Network, Vol 28, Issue 6, pp: 27-33.

[17] OpenFlow-Enabled Mobile and Wireless Networks. [Online]. Available: https://www.opennetworking.org/images/stories sdn-resources/solution-briefs/sb-wireless-mobile.pdf.

[18] Network Functions Virtualisation in ETSI. [Online]. Available: http://www.etsi.org/technologies-clusters/technologies/nfv.

[19] 3GPP TS36.101 Specification. [Online]. Available: http://www.3gpp.org/ftp/Specs/archive/36_series/36.101/36101-am0.zip