Devote: Criticality-Aware Federated Service Provisioning in Fog-Based IoT Environments

Minu Tiwari, Sudip Misra, *Senior Member, IEEE*, Pradyumna Kumar Bishoyi, *Student Member, IEEE*, and Laurence T Yang, *Fellow, IEEE*

***Abstract*—In this paper, we present an efficient criticality-aware decision-making system, named Devote, for fog-based Internet of Things (IoT) environment. Devote introduces an intelligent algorithm for the service of data based on the criticality, while considering the current availability of the resources at the fog node. To cope with the dynamic IoT environment, we adopt a reinforcement learning-based algorithm for the processing of the IoT data based on time-varying conditions. Additionally, we propose an efficient online secretary-based algorithm for choosing the best suitable candidate fog node for offloading the data. To show the effectiveness of Devote, we obtained the numerical results for assessing its performance, while collating it with the benchmark schemes. We analyze different performance metrics such as service delay, economy, and user satisfaction, which show that Devote incurs less service delay, as compared to other systems, while achieving user satisfaction of 88.4%.**

***Index Terms*—Fog computing, SARSA, Q-learning, online secretary algorithm, decision-making, Markov decision process**

I. Introduction

Fog computing introduces a middle layer between IoT devices and the cloud, which reduces the volume of raw data transmitted to the cloud, thereby minimizing the bandwidth requirement and the response time [1]. Fog computing plays a crucial role in delay-sensitive IoT environments like the healthcare and mining industry [2], [3], where the the data processing delay may result in serious accidents. In fog architecture, the fog nodes (FNs) are physically located close to the IoT devices and therefore, are capable of serving the data faster [4]. However, the data should be classified to decide where it should be processed – fog or cloud.

To answer the trail of questions in the context of which data is to be processed at which layer, researchers [5], [6] proposed various offloading algorithms considering different aspects. The FNs can either serve the data by themselves or offload horizontally or vertically: to cloud or other FN based on the requirement of the data. However, the classification should be done for the data that need immediate action and those that can tolerate delay. Also, which algorithm is to be applied to which data and to be served by whom is a significant concern. This induces the need for a decision-making algorithm at the fog layer to help the FNs make this decision. Nonetheless, in the dynamic environments of IoT, the FNs must harmonize themselves according to the environment changes, so that they can update their knowledge with the change. Thus, learning the environment by the FNs to take efficient actions is necessary.

M. Tiwari and S. Misra are with the Department of Computer Science and Engineering, Indian Institute of Technology Kharagpur, 721302, India (e-mail:{minutiwari, sudipm}@iitkgp.ac.in).

P. K. Bishoyi is with Advanced Technology Development Center, Indian Institute of Technology Kharagpur, 721302, India (e-mail: pradyumna.bishoyi@iitkgp.ac.in).

L. T. Yang is with Department of Computer Science, St. Francis Xavier University, Antigonish, Canada (e-mail: ltyang@stfx.ca).

A. Motivation

The decision of what data is to be served at which layer is taken by considering different aspects such as the energy of the fog nodes, bandwidth, traffic of the network, and the delay of the task. The existing algorithms in the literature attempted to solve the questions – where to offload the data [6], [7] and when to offload the data [5], [8]. However, the IoT environment comprises of diverse data that makes it imperative to offer differential priority to data, which induces the need of data classification.

Furthermore, a singular algorithm for processing all the requests is not prudent. Since the delay demand for different data varies, it becomes important to find out the proper suitability of the algorithm according to the nature of the data. The decision becomes crucial in the environments, where a minor delay in the processing of the data may result in serious hazards. Therefore, the FNs should be intelligent enough to prioritize the data and decide the suitability of an algorithm for the processing of specific data.

Besides, the IoT environment changes with time and demands the FNs to update their knowledge according to the changes. This dictates the need for a system that will adapt itself to the changes. However, the nature of data cannot solely determine its processing, since the entity which has to perform the task should be capable enough to do so. Therefore, the location to process the data becomes important– whether to process the data by itself or offload it horizontally or vertically. There is a need for a system that would pave a way for – *how to classify the data according to their priority, which data needs to be processed using what algorithm, how to offload the data: horizontally or vertically, and whom to offload if selected horizontal offloading*.

To overcome the above-discussed problems, we present an intelligent reinforcement learning-based algorithm that decides the processing of the data based on its severity. The algorithm manifests a way to the FNs for suitable applicability of different algorithms, which is the first of its kind.

B. Contributions

To address the prior discussed problems, we design solutions combining reinforcement learning-based processing and

data offloading. Specifically, the main contributions of the paper are listed below.

- We propose a hierarchical structure that separates entities inside the FNs for proper categorization of the data based on its criticality and making the decision of either serving by self or through horizontal/vertical offloading.
- We formulate a Markov Decision Process (MDP)-based framework for the efficient allocation of resources of FNs while considering the criticality of incoming data traffic.
- We present an adaptive reinforcement learning algorithm-based solution for learning the optimal policies in the formulated MDP problem for the self-service of the data by the FN.
- We introduce an efficient algorithm based on online secretary problem for choosing the most suitable candidates among the group of members for the decision of offloading.

The rest of the paper is organized as follows. The study of the related literature is presented in Section II. Section III describes the system model of Devote. Section IV presents the formulation of MDP for the problem. The proposed decision-making system and the offloading solution are presented in Sections V and VI, respectively. Section VII discusses the performance evaluation of Devote. Finally, Section VIII concludes the paper.

II. Related Works

We study the existing works for task offloading in cloud-fog architecture. For efficient data processing, different algorithms are proposed [9], [10], considering various aspects such as energy of the FN, delay of the task, and computation workload. Bozorgchenani *et al.* [9] proposed a partial task offloading approach by considering the trade-off between the energy consumption and incurred service delay of the task computation. To efficiently utilize the resources of FNs, Mishra *et al.* [10] proposed a resource utilization scheme considering the load on FNs and network links. They aimed to minimize the overall system delay. However, considering the possibility of offloading the data to the other FNs in the vicinity is lacked in [9]. Further, both the works have not considered the priority of the data.

Several learning-based works [5], [7] were proposed in the literature to know the current availability of other FNs. Baek *et al.* [7] proposed a Q-learning based solution to know the most suitable candidate node for offloading the data. Another work for computation co-offloading is proposed by Wang *et al.* [5] in which they focused on the energy consumption and service delay of a task by modeling the willingness parameter of other FNs. However, different priorities of serving the data based on its nature is not addressed.

Synthesis: The study of research works unveils that the existing works adopt a particular method for achieving the objective of load balancing, processing, or resource management. However, the situation becomes worse when sensitive data and very normal data go through the same procedure of processing, incurring a major delay for sensitive data and a minor delay for a normal data, which is undesirable. Further, it is important for the FNs to be intelligent enough to decide what action is to be taken on which data according to which algorithm. Therefore, in this work, we present a novel decision-making algorithm adaptable to the dynamic environment and prioritizing the data based on its severity.

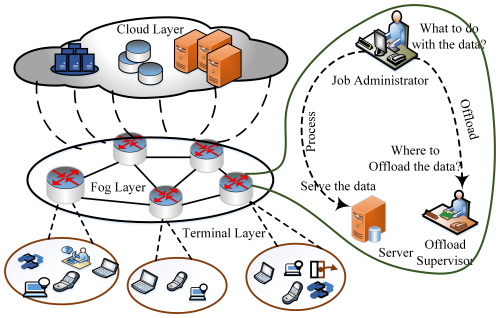


Fig. 1: System model architecture

III. Problem Formulation

A. System Model

We consider a three-layer fog computing architecture comprising the terminal layer, fog layer, and cloud layer as depicted in Figure 1. The terminal layer consists of N number of IoT devices, represented as the set, $\mathcal{N} = \{1, 2, \cdots, N\}$. The set of FNs is represented as $\mathcal{FN} = \{f_1, f_2, \ldots, f_P\}$. The devices in the terminal layer and the fog nodes share a surjective relation, i.e., each device is connected to one of the FNs, which can be defined as $\mathcal{FN} = \{f_{11}, f_{12}, \ldots, f_{1n_1}, f_{21}, f_{22}, \ldots, f_{2n_2}, \ldots, f_{P1}, f_{P2}, \ldots, f_{Pn_P}\}$, $\sum_{i=1}^{P} n_i = N$. The data/tasks generated by the IoT devices are transmitted to the FNs through wireless devices. Also, FNs can communicate with other FNs in their vicinity. For each of the data coming from the terminal layer, the FNs are responsible to make the decision whether it is to be forwarded to the cloud or is to be processed at the fog layer.

The FNs are penalized for rejecting a task or exceeding the expected delay for processing the task. However, for each of the task served, the FNs earn the respective task revenue. Notably, FNs are small devices with limited computational resources such as memory, power, and CPU. Let us denote the resources of the FNs pertaining to their memory (M), power (P), and CPU frequency (C) as a function of $\mathcal{R} = f(M, P, C)$. The computation requires the power which is proportional to the CPU frequency, defined as $P \propto (C)^3$ [11]. We consider that a constant R_{res} amount of resource is being reserved for the FNs for receiving the data from the IoT devices.

1) Entities in FN: The decision-making involves different functionalities of an FN. Therefore, the module design of *FNs* in our system consists of a *job administrator*, *offload supervisor*, and a *server*. The responsibilities of each of these modules are explained below.

i) Job administrator: The job administrator has the responsibility of receiving the data from the IoT devices, computing the criticality of the data, and making the decision of offloading

or serving. For these tasks, this module utilizes R_{res} amount of the total resources of the FN.

ii) Offload supervisor: If the decision taken by the job administrator is to offload then the *offload supervisor* is responsible for deciding where to offload – to the cloud or other FNs. The decision of *offload supervisor* is fully dependent on the severity of data and the availability of eligible FNs for offloading. The severity of data indirectly defines the delay requirement of particular data. Therefore, the *offload supervisor* considers the delay associated with each data before making the decision. The delay associated with the processing of each data is interpreted in terms of (a) *Migration delay* and (b) *Computation delay*.

The time required to transmit i^{th} task from one FN to another FN is the *migration delay*. Mathematically, $T^i_{migr}(t) = \frac{L_i}{r_{jk}}, \forall i \in [1, N]$. Here, L_i is the size of the task to be transmitted and r_{jk} is the data rate from FN j to FN k, which is computed using Shannon-Hartley equation as, $r_{jk} = Blog(1 + \frac{H_{jk}P_{tx}(j)}{BN_0})$, where, B is the bandwidth, H_{jk} is the channel gain, $P_{tx}(j)$ is the transmission power of node j, and N_0 is the noise power spectral density.

Computation delay is the time required to process a particular task i on a particular FN j, $\forall j \in \mathcal{FN}$, at a decision epoch t. Mathematically, computation delay is defined as: $T^i_{comp}(t) = \frac{\nu_i}{f_j}, \forall i \in [1, N]$, where ν_i represents the clock cycle required to execute i^{th} task and f_j is the CPU frequency of the FN j. We assume that the migration delay to transmit the data to other FNs is less than forwarding the same to the cloud. Therefore, it is always preferred to forward the data to other FNs, if available, rather than forwarding it to the cloud.

iii) Server: The data to be self-served by the FN is forwarded to the *server* for processing. However, this serving decision is taken based on the availability of the resources, and consequently on the severity of the data as well.

2) Criticality of Data: The nature of data becomes an important factor in the environment, where the applications are highly latency-critical and a minor delay in processing the data in such a scenario may result in serious havoc.

Definition 1. *Criticality: The criticality of the data is defined as the deviation of the current data value from its predefined threshold limit. Mathematically,*

$$\phi^i = \frac{(val^i_{th,up} - val^i_{curr})^2 - (val^i_{curr} - val^i_{th,low})^2}{(val^i_{th,up} + val^i_{th,low})^2} \quad (1)$$

where $val^i_{th,up}$ is the upper threshold value, $val^i_{th,low}$ is the lower threshold value, and val^i_{curr} is the current value of the data. The thresholds of the data vary for different applications from time to time.

Based on the criticality of the data, we consider three types of data in our system, namely, *too critical*, *critical*, and *normal*, and denoted as ϕ_{tc}, ϕ_c, and ϕ_m, respectively. The more the value of ϕ^i, the higher the data criticality.

3) Idleness of the FNs: The FNs are limited in resources, and therefore, they need to utilize their resources efficiently. Let the set $\mathcal{R}^m_t = \{0, 1, ..., \mathcal{R}^m_{max}\}$ represent the number of current unutilized resources of any FN at a decision epoch t, $\forall m \in \mathcal{FN}$. As stated earlier, R_{res} amount of resources are always reserved, and hence, the maximum number of unutilized resources for any FN at any t can be $(R^m_t)' = \{0, 1, \ldots, \mathcal{R}^m_{max} - R_{res}\}$. Since R_{res} is very small, it can be neglected, which thereby results in $(R^m_t)' = \mathcal{R}^m_t$. The FNs can accommodate any offloaded task, if they satisfy $\mathcal{R}^m_t > 0$. Eventually, any FN is considered as idle, whenever its number of unutilized resources is greater than 0, i.e., $\mathcal{R}^m_t > 0$. On the other hand, if all the resources of any FN at epoch t are involved in the processing of their data or any offloaded data then $\mathcal{R}^m_t = 0$. This is the busy state of the FN and is no longer available for handling any other offloaded data. Let the probability of any FN to be idle be denoted by P_{idle}. Therefore, at any decision epoch t, the probability of p number of FNs being idle is computed as:

$$\mathcal{P}^p(t) = \binom{P}{p}(P_{idle})^p(1 - P_{idle})^{P-p}, \forall p \in [1, P] \quad (2)$$

4) Utility of the FNs: We define the utility of FNs as a function of user satisfaction (ρ), service revenue (ζ), and the number of users being served (n), denoted as $U = \{\rho, \zeta, n\}$. The decision of the FN to offload or serve the data is dominated by the resource requirement for processing the current data, the criticality of the data, and the revenue it gets if the data is processed by itself. The revenue is affected by the number of users it can satisfy with proper decisions. The user satisfaction, in turn, depends on the delay caused in processing the data. Although, there is an expected delay with the processing of each data but the more the delay, the lesser the user satisfaction. At the same time, the FNs also want to efficiently utilize their resources by serving more users, which can increase the delay in processing the data; and consequently, affect the user satisfaction.

Adapting to the dynamic IoT environment, the objective of the FNs is to maximize their utility by serving more number of users, while earning more service revenue along with the efficient utilization of resources, expressed as:

Objective:

$$\max_{R^m_t} U$$

subject to

$$0 \leq \rho \leq 1;$$

$$T_{expected} \leq T_{tot} \leq T_{max};$$

where $T_{expected}$ is the expected delay associated with each task, T_{tot} is the total delay incurred by the task, T_{max} is the maximum delay, and $T_{max} < \infty$.

To maximize the utility of each FN, we formulate the problem as MDP. MDP allows an agent to follow a policy that will eventually lead to an optimal decision. In our discrete system, the FNs decide the action to be taken according to their current availability of resources, while achieving the maximum possible utility in that state at each time slot. This problem can be regarded as a sequential decision problem and can be formulated as MDP.

IV. MDP Formulation

We formulate the problem of finding an optimal policy at each state of the FN as an MDP. In an MDP, the next state is decided based on the actions taken in the current state and the transition probability of going to a particular state. This characterizes the behavior of the environment. An MDP is a 5-tuple consisting of states, actions, transition probabilities, and rewards defined as $\{D, H, X, P, R\}$. The detailed description of the entities of MDP is presented below.

A. Decision Epoch

In the described system model, there are $\mathcal{N}$ number of IoT devices that generate data. The decision on each data cannot describe the state of the system entirely. Therefore, the decision needs to be taken on the data collected over a time period, while concurrently managing the delay of each data by processing it within the expected delay. Thus, the maximum allowable value of the decision epoch is T_{max}. We define the set of total decision epochs as $\mathbf{T} = \{1, ..., T_{max}\}$.

B. System State

We define the state of the system as the state of FN by taking a particular action. This state is interpreted as the number of free resources of the FN at any decision epoch t. The resource is interpreted with the same definition as described in Section III. We consider the quantized value of resources as $\mathcal{R}_t^m = \{1, 2, ..., \mathcal{R}_m^{max}\}$. Therefore, the state space is defined as $H_t = \mathcal{R}_t^m$.

C. Action

The FN is responsible to take action at each decision epoch based on the nature of the data. We define five actions for an FN to take at each decision epoch. Thus, our action set consists of five components, represented as $\mathcal{X} = \{x_{sS}, x_{sQ}, x_{oQ}, x_{oT}, x_{uC}\}$, where x_{sS} defines *self-serve SARSA*, x_{sQ} defines *self-serve Q-learning*, x_{oQ} denotes the *offloading Q-learning algorithm*, x_{oT} indicates the action *offloading total task to other FNs*, and x_{uC} is *uploading the task to the cloud*.

D. Transition Probabilities

The FN transits from one state to other when a particular action is taken at a decision epoch t. The FN will change its state from h_t to the other state h_{t+1} by taking an action $x_t \in \mathcal{X}$, where h_t and h_{t+1} are the current and the next state, respectively. The probability associated in this particular transition is called the transition probability, denoted as, $Pr(h_{t+1}|h_t, x_t)$. The transition probability associated with different actions are defined as:

$$Pr_{cloud} = Pr(\phi^i = \phi_m)\mathcal{P}_t^p, \forall p = 0 \tag{3}$$

$$Pr_{off-QL} = Pr(\phi^i = \phi_c)\mathcal{P}_t^p, \forall p = \chi \tag{4}$$

$$Pr_{off-tot} = Pr(\phi^i = \phi_m)\mathcal{P}_t^p, \forall p > 1 \tag{5}$$

$$Pr_{self-QL} = Pr(\phi^i = \phi_c)\mathcal{P}_t^p, 1 \leq p < \chi \tag{6}$$

$$Pr_{self-SARSA} = Pr(\phi^i = \phi_{tc}) \tag{7}$$

where, $p \in [1, P]$ and χ denotes the minimum number of fog nodes required to execute the distributed Q-learning algorithm so that the delay does not exceed T_{max}.

E. Reward

Each of the transitions to a new state is associated with some reward. Mathematically, $R_t(h_t, x_t)$ represents the reward for the transition by taking action x_t on the current state h_t. We define the reward function as:

$$R_t(h_t, x_t) = \sum_{i=1}^{n} \rho_i \zeta_i \tag{8}$$

Definition 2. *User satisfaction (ρ): It is a parameter to measure the quality of experience of a user based on the delay for the processing of the data. There is an inverse relationship between the user satisfaction and the delay; the more the delay, the less is the user satisfaction. If ρ_i = 1, the user is fully satisfied as the task is completed with in the expected amount of delay of the task.*

$$\rho_i = \begin{cases} 1, & \text{if } T_{expected} \leq T_{tot} \\ \Upsilon_i \frac{T_{expected}}{T_{tot}}, & \text{otherwise} \\ 0, & \text{if } T_{tot} \geq T_{max} \end{cases} \tag{9}$$

where Υ_i is the user preference parameter associated with the i^{th} task and T_{tot} represents the total time taken for the processing of the task.

Definition 3. *Service Revenue: Every task is associated with a revenue, ζ. When the task is served, the revenue is paid back as the profit to the serving agent. The service revenue gained for a task $j, \forall j \in [1, N]$ depends on the action performed in the particular state.*

To solve the above formulated MDP and provide a solution to the decision-making problem, we propose a Reinforcement Learning (RL) based solution for making the FNs adapt to the change in the environment while maximizing their utility. However, at each particular state, the agent decides the reward maximizing action to be taken, which is defined by the policy that the agent adopts. Given the states, the policy Φ defines a distribution over the possible actions, i.e., $\Phi(x|h) = Pr[\mathcal{X} = x|H_t = h]$. The objective is to find an optimal policy to maximize the return in every state. However, a particular algorithm cannot be used for diverse data of the IoT environment. Nevertheless, since the resources of the FNs are limited, an intelligent decision of what data to be processed, how it is to be processed, and where it is to be processed plays a major role, keeping the objective always in mind. Therefore, we adopt a combination of RL algorithms to solve the problem.

V. Proposed Model

We describe the proposed model for addressing the problem of what algorithm to apply on which data. The system named *Devote* is founded on the interworking of different RL algorithms, where the method is decided with reference to the criticality of the data. The name Devote signifies the inclination of the system towards the service of critical data.

A. Adjudgement on the data

When the IoT data arrives at the FN, data classification is done to prioritize the data according to its severity. Therefore, in our model, we proposed ϕ^i to classify the data, as discussed in Section III-A2. The data is classified as *too critical, critical*, and *normal*. If the computed ϕ^i classifies the data as too critical (ϕ_{tc}), the data needs immediate processing along with the efficient result, and hence, we adopt the SARSA algorithm [12] for processing the data. On the other hand, if the ϕ^i is classified as ϕ_c, the data is critical and the processing cannot be delayed much without compromising the result. Therefore, to process critical data, Q-learning [12] is adopted as the method of processing. Lastly, if the data do not belong to this category then there is no urgency with the data and its processing can be delayed. Therefore, it can be offloaded to another FN or to the cloud.

B. The final verdict

To ensure that the processing of too critical data does not get delayed, Devote renders the FN to reserve the resources required for implementing the SARSA and will be available for handling the other offloaded task only if its resources exceed this reserved amount. Let us consider that $\mathfrak{S}$ amount of resource is required for executing the SARSA algorithm, then an FN is available for offloading based on the following condition.

$$P_{i,t}^{avl} = \begin{cases} 1, & \text{if } \mathcal{R}_m^{i,t} - \mathfrak{S} > 0 \\ 0, & \text{otherwise} \end{cases} \tag{10}$$

where $P_{i,t}^{avl}$ is the probability that the i^{th} node is available for handling the offloaded task at a decision epoch t and ($\mathcal{R}_m^{i,t}$ - $\mathfrak{S}$) is the number of free resources of the i^{th} FN.

As explained in Section V-A, after finding the value of ϕ^i for categorizing the data, deciding a suitable algorithm also depends on the availability of the resources at the FN. If the data is categorized as *too critical*, the host FN serves the data itself. Otherwise, if the data is *critical* and the system has enough resources, it will use Q-learning on its own to process the data.

If the host FN is not capable of processing critical data, it tries to offload the data to other FNs for implementing Q-learning. Since every FN is busy serving its own data, it may not be able to implement Q-learning on its own. However, an FN may collaborate with other FNs to process the data. To do so, the host FN needs to choose the best possible set of candidate nodes. To accomplish the task, the set of best candidate nodes are chosen using the secretary algorithm as elucidated in section VI and requests them to implement distributed Q-learning [13] on the data.

If the data belongs to *normal* category, it is offloaded to the neighboring nodes that possess the resources and willing to accept the data. However, if no such node is found, the data is sent to the cloud for processing. It is assumed that a *normal* data can tolerate a viable amount of delay induced for sending to the cloud and the response to come back. Algorithm 1 describes the decision-making process.

VI. Data Offloading

In our problem, when an FN is overburdened with its data processing and faces critical data very frequently, it needs to offload its data. The FN first elects the candidates and then finds suitable candidates for offloading.

Algorithm 1: Algorithm for decision-making

```
Inputs: Sensed data from the sensors.
Output: Decision – offload/process.
Initialize F_can ⟵ {∅};
for i ∈ [1, N] do
    Compute φ^i using Equation 1;
    if φ^i == φ_tc then
        SARSA(R_M);
    else if φ^i = φ_c then
        if R_m > 𝔖 then
            Q-learning(R_m);
        else
            F_can = FindCandidate(FN);
            Offload i to F_can;
            Call distributed Q-learning;
    else
        F_can = FindCandidate(FN);
        if F_can == 0 then
            Offload i to cloud;
        else
            Offload i to F_can;
```

A. Electing the candidates

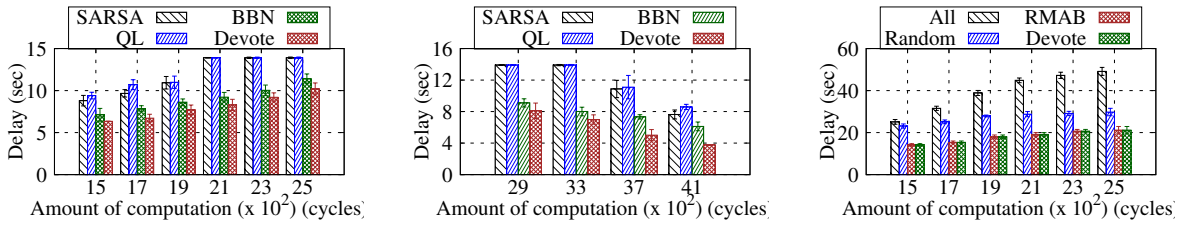
To find out the group of suitable candidate FNs in the vicinity of the host FN, where the offloading can be performed, the host broadcasts an offload request packet. The FNs that fulfill the minimum requirement of resources for handling the data, reply back with a beacon signal indicating that they are willing to handle the offloaded data. We denote this set of nodes by $\mathcal{F}(t)$. The problem of finding the most suitable nodes from the set $\mathcal{F}(t)$ is in line with the knapsack problem which is NP-complete. Various approaches are suggested in the literature for solving this problem such as online auction theory [14] and online secretary problem [15], that will yield an efficient set of nodes capable of handling the offloaded data. Since the number of candidate FNs are deterministic and the selection is completely based on the capability to handle the task, we adopt an online secretary algorithm for finding the solution to this problem.

The online secretary problem [15] aims at finding out the set of most suitable set of candidates for a position in the company. Moreover, the order in which the candidates will arrive for the interview is not known and the decision has to be made immediately after each interview. The most suitable candidates for offloading the task will be those which can offer the best possible service to the data keeping the utility optimal. Algorithm 2 describes the candidate FN selection procedure.

VII. Performance Evaluation

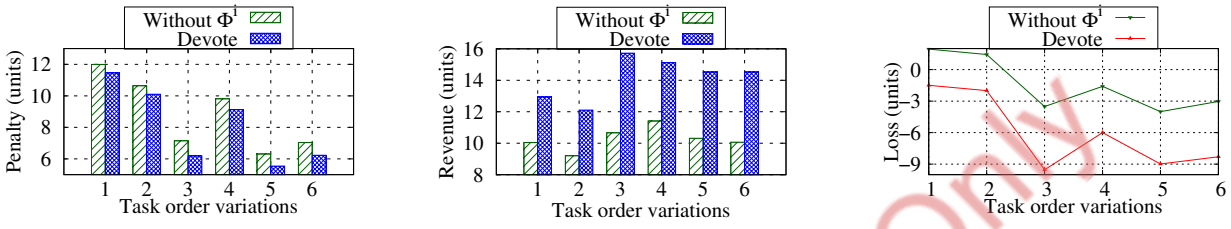
A. Numerical Analysis Setup

In this section, we analyze the performance of the proposed system by computing the numerical results using the MATLAB platform. We consider a network area of 1000 x 1000 m^2 in which the FNs are allocated randomly. The parameters used for numerical analysis along with their values is shown in



(a) Varying computation (b) Varying CPU cycle (c) Comparison of secretary algorithm

Fig. 2: Service delay



(a) Penalty (b) Revenue (c) Loss

Fig. 3: Economy of the system

Algorithm 2: Algorithm for finding candidates for offloading

Inputs: $\mathcal{FN}^P$.
Output: Candidate fog nodes.
function FindCandidate ($\mathcal{FN}$)
Initialize $\mathcal{F}_t \longleftarrow \{\emptyset\}$;
Broadcast an offload request;
for $j \in [1, P]$ **do**
  if $\mathcal{R}_m^j > \mathfrak{S}$ **then**
    Send beacon to the node i and add F_j to $\mathcal{F}(t)$, $\mathcal{F}(t) \longleftarrow \mathcal{F}(t) \cup \{F_i\}$;
    Apply secretary($\mathcal{F}(t)$) to find the candidate fog nodes;
  else
    Do not send the beacon;

Table I. We create an IoT environment based on the varying nature of data by considering three different types of tasks, based on the criticality of data – *too critical*, *critical* and *normal*. To compute the economy of the system, the number of tasks is variated in the proportion of 1:2:3. The overall performance is computed by averaging all the possibilities of task arrival, considering all the possible combinations. Moreover, we obtain the plots using 95% confidence interval.

TABLE I: Simulation Parameters

Parameters	Values
Discount factor (γ)	0.7
Learning rate (α)	0.01
Step size (n)	1
Number of resources	100-500
Amount of computation	1500-2500 Megacycle [16]
Average task size	500 MB [7]
CPU frequency	2.9-4.2 GHz [16]
Transmission power of fog node	20 dBm [16]
System bandwidth	20 MHz [16]

B. Performance metrics

The following metrics are used to examine the performance of Devote.

1) Service Delay: It is defined as the total delay incurred for processing the data, which includes the processing delay along with the migration delay, if offloaded, computed as: $T_{tot} = T^i_{comp}(t) + T^i_{migr}(t), \forall i \in [1, N]$.

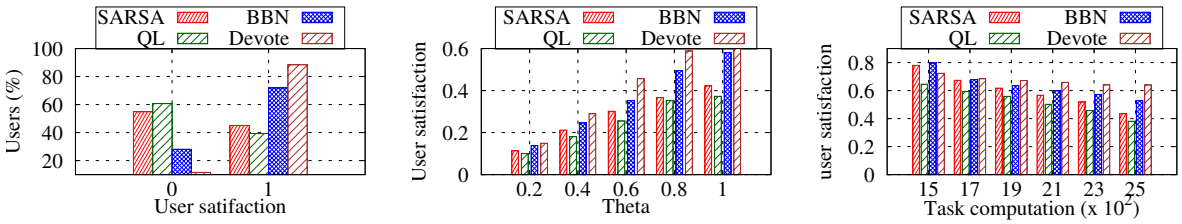
2) Loss: It is the value of difference between the total penalty incurred for rejecting the data and the total service revenue obtained for servicing the data. Mathematically, $Loss = \sum_{i=1}^{RT} \xi_r P_i - \sum_{i=1}^{ST} (1 - \xi_r)\zeta_i$, where ξ_r is the penalty factor for the rejection of the task, RT and ST are the total number of rejected and served tasks along with their respective penalty P_i and the service revenue ζ_i.

3) User Satisfaction: We calculate the percentage of user satisfaction for each case of the user preference parameter using Equation (9).

C. Benchmark

To assess the performance of Devote, we compared it with the existing scheme [17] named *BBN* and two other schemes, namely, *SARSA* and *QL*. BBN uses a bayesian belief network at the fog layer for event classification, which would pave the way for showing the efficiency of using RL algorithms. The system adopting the SARSA algorithm uses SARSA as the primary algorithm and offloads the data if it does not have sufficient resources to process, while the system adopting QL uses Q-learning as the primary algorithm and offload the data to the cloud when there is a shortage of resources. These systems may be beneficial in some of the cases but fail when there is a varying type of data demanding different delays for the processing of the data.

Similarly, we compared Devote thatuses the secretary algorithm for choosing the best possible set of candidate nodes



(a) Yes/No (b) Different user preferences (c) Varying computation

Fig. 4: User satisfaction

for offloading the data with an existing system *RMAB* [18] that adopts exploration and exploitation for selecting a fog node. Additionally, Devote is compared with two traditional schemes, system choosing all the candidate nodes named *All* and the system choosing randomly some fixed number of nodes called *Random* from the candidate set. *All* expends its resources in transmitting data to all nodes while *Random* ends up doing the worst selection of nodes.

D. Results

1) Service Delay: The evaluation of service delay compared with different schemes is shown in Figure 2. Figure 2(a) depicts the variation in service delay with respect to the increase in the amount of computation. It is evident from the figure that the SARSA and QL methods suffer more delays in comparison to Devote since they offload high computation-intensive (above 2000 megacycle) tasks. BBN incurs high delay for critical data, which is transmitted to the cloud for processing. Devote outperforms the other methods by considering the criticality of data. The variation in service delay with the increase in CPU processor capacity is shown in figure 2(b). The figure shows that SARSA and QL systems offload most of the data, when the processor capacity is less, and hence, suffer high delay, whereas Devote owing to the consideration of the criticality of data obtains less delay. In the case of BBN, the delay increases due to the reason of offloading the critical data to the cloud. Therefore, Devote is efficient in terms of service delay.

In addition, to show the efficiency of Devote for using the online secretary algorithm, the system is compared with *RMAB*, *All*, and *Random*. Figure 2(c) shows that the algorithm choosing all the available FNs for offloading performs worst, while the secretary algorithm performs the best. This is because *All* incurs a high amount of migration delay for transmitting the request to all the available candidate nodes. *Random* always selects a fixed number of random candidate nodes from the total available nodes. The strategy of RMAB to explore and exploit for choosing FN for offloading leads to increased delay if a best FN is ignored. Devote always chooses the minimum best candidate nodes among all the available candidate FNs available for offloading.

2) Loss: The total loss/gain of the system in terms of penalty, revenue, and the overall loss is shown in Figure 3. The FNs are charged a penalty for each of the tasks they reject and offload to other FN or cloud and are awarded the revenue for task service. It is noteworthy to mention that the task with higher ϕ^i pays more service revenue and penalizes more for its rejection. The latency is increased with offloading and costs more if offloaded to the cloud in lieu of other FNs.

Figure 3(a) shows the total penalty obtained for rejecting the task by the two systems: the system which offloads the task without considering the criticality and Devote. The figure shows that the system not considering the criticality of data incurs more penalty compared to Devote. This is due to the fact that it keeps on processing the task till it has its resources left without considering criticality, while Devote offloads the *normal* (ϕ_m) task.

Figure 3(b) shows the total revenue obtained for each of the tasks served by each of the systems. Since the revenue associated with the higher ϕ^i task is more, Devote performs better in comparison to the other system. This is because Devote prioritizes the higher criticality task by getting the lower ϕ^i task done by others and saving its resources for the higher ϕ^i task which pays off more service revenue.

The total loss of Devote is lower as compared to the other system, as shown in Figure 3(c). The reason being the same is that the higher ϕ^i task pays off more service revenue and even costs more penalty if not served. Since Devote prioritizes the higher ϕ^i tasks, it obtains less penalty for lower ϕ^i tasks and attains more revenue for the higher ϕ^i task. Thus, a balanced economy is obtained with Devote, however, the other system undergoes more loss.

3) User Satisfaction: Figure 4 shows user satisfaction achieved by all the systems. The user satisfaction depends on the delay incurred to process data with respect to their criticality and the user preference parameter. Figure 4(a) presents the number of users satisfied and dissatisfied in different systems. It is evident from the figure that Devote achieves the satisfaction of 88.4% of users, BBN obtains 71.9%, while SARSA and QL achieve the satisfaction of 45.1% and 39.3%, respectively. Devote achieves better results with the consideration of the criticality of the data.

We also obtained the results by variating the user preference parameter from 0 to 1. Figure 4(b) shows that the satisfaction obtained in Devote is always higher for each of the values of the user preference and it keeps on increasing with the increase in the user satisfaction parameter. Since SARSA and QL do not consider the criticality of the data, they fail to achieve the expected delay of *too critical* data. Again, BBN offloads the critical data incurring high delay and possesses less user satisfaction. Therefore, Devote achieves higher satisfaction

with the consideration of ϕ^i.

In Figure 4(c), we obtained the results by taking a fixed user preference parameter of 0.5 and varying the amount of computation and criticality for the data. In this case, Devote excels QL, however, at times, BBN and SARSA have performed better. This is the case when a *too critical* data is served with in its expected delay by BBN and critical data is served with the high preference by SARSA incurring a lesser delay. Yet, BBN lags to achieve the satisfaction for critical data and SARSA fails in the long run when all the resources are exhausted serving the critical data and it is forced to offload the too critical data, which is undesirable. Also, QL always lags behind because the system spends a lot of time processing data neglecting the criticality.

VIII. Conclusion

In this paper, we presented an efficient decision-making system, which provides intelligence to the fog nodes for deciding a suitable algorithm to process the data. Devote is designed using reinforcement learning algorithms, which help it to adapt to the dynamic environment IoT environment. The decision of a suitable algorithm is based on the nature of the data – *too critical, critical*, and *normal*. Further, to choose a suitable candidate fog node for offloading the critical data, we proposed an online secretary-based algorithm. The algorithm manages the trade-off between the efficient service of data along with the delay of processing. Numerical analysis depicted that Devote incurred less service delay while achieving higher user satisfaction.

The mobility of IoT devices as well as the fog nodes is typical in many applications such as vehicular networks. In this work, we did not consider the aspect of mobility. Thus, we plan to consider the mobility and heterogeneity aspect of devices as the future extension of this work. In addition, we will extend the proposed system by implementing it in a real fog network in our future work.

References

[1] G. Kumar, R. Saha, M. K. Rai, R. Thomas, G. Geetha, T. Hoon-Kim, and J. J. P. C. Rodrigues, "A novel framework for fog computing: Lattice-based secured framework for cloud interface," *IEEE Internet of Things Journal*, Apr 2020.

[2] M. Aazam, K. A. Harras, and S. Zeadally, "Fog computing for 5G tactile industrial Internet of Things: QoE-aware resource allocation model," *IEEE Transaction on Industrial Informatics*, Mar 2019.

[3] M. Thibaud, H. Chi, W. Zhou, and S. Piramuthu, "Internet of Things (IoT) in high-risk environment, health and safety (ehs) industries: A comprehensive review," *Decision Support Systems*, vol. 108, pp. 79–95, Apr 2018.

[4] R. J. Tom, S. Sankaranarayanan, and J. J. P. C. Rodrigues, "Agent negotiation in an IoT-Fog based power distribution system for demand reduction," *Sustainable Energy Technologies and Assessments*, vol. 38, p. 100653, Apr 2020.

[5] Y. Wang, K. Wang, H. Huang, T. Miyazaki, and S. Guo, "Traffic and computation co-offloading with reinforcement learning in fog computing for industrial applications," *IEEE Transaction on Industrial Informatics*, vol. 15, pp. 976–986, Nov 2019.

[6] C. Swain, M. N. Sahoo, A. Satpathy, K. Muhammad, S. Bakshi, J. J. P. C. Rodrigues, and V. H. C. de Albuquerque, "Meto: Matching theory based efficient task offloading in iot-fog interconnection networks," *IEEE Internet of Things Journal*, Sep 2020.

[7] J.-y. Baek, G. Kaddoum, S. Garg, K. Kaur, and V. Gravel, "Managing fog networks using reinforcement learning based load balancing algorithm," *arXiv preprint arXiv:1901.10023*, Apr 2019.

[8] M. Ammad, M. A. Shah, S. U. Islam, C. Maple, A. A. Alaulamie, J. J. P. C. Rodrigues, S. Mussadiq, and U. Tariq, "A novel fog-based multi-level energy-efficient framework for iot-enabled smart environments," *IEEE Access*, vol. 8, pp. 150 010–150 026, Jul 2020.

[9] A. Bozorgchenani, D. Tarchi, and G. E. Corazza, "Centralized and distributed architectures for energy and delay efficient fog network-based edge computing services," *IEEE Transaction on Green Communications and Networking*, vol. 3, no. 1, pp. 250–263, Dec 2018.

[10] S. Mishra, M. N. Sahoo, S. Bakshi, and J. J. P. C. Rodrigues, "Dynamic resource allocation in fog-cloud hybrid systems using multicriteria ahp techniques," *IEEE Internet of Things Journal*, vol. 7, no. 9, pp. 8993–9000, Jun 2020.

[11] A. Wierman, L. L. Andrew, and A. Tang, "Power-aware speed scaling in processor sharing systems," in *In Proc. of IEEE INFOCOM 2009*, Apr 2009, pp. 2007–2015.

[12] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 1998.

[13] M. Lauer and M. Riedmiller, "An algorithm for distributed reinforcement learning in cooperative multi-agent systems," in *Proc. of the Seventeenth International Conference on Machine Learning*, 2000.

[14] G. Iosifidis and I. Koutsopoulos, "Challenges in auction theory driven spectrum management," *IEEE Communications Magazine*, vol. 49, no. 8, pp. 128–135, Aug 2011.

[15] J. Preater, "On multiple choice secretary problems," *Mathematics of Operations Research*, vol. 19, no. 3, pp. 597–602, Aug 1994.

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

[17] P. Verma and S. K. Sood, "Fog assisted-IoT enabled patient health monitoring in smart homes," *IEEE Internet of Things Journal*, vol. 5, no. 3, pp. 1789–1796, Feb 2018.

[18] M. Yang, H. Zhu, H. Wang, Y. Koucheryavy, K. Samouylov, and H. Qian, "An online learning approach to computation offloading in dynamic fog networks," *IEEE Internet of Things Journal*, Aug 2020.