RTOS: Response-aware Traffic Offloading Scheme using Regression Machine Learning for User-Centric Large-Scale Internet of Things

Gunasekaran Manogaran, Gautam Srivastava (Senior Member, IEEE), Bala Anand Muthu, S. Baskar, P. Mohamed Shakeel, Ching-Hsien Hsu, Ali Kashif Bashir (Senior Member, IEEE), Priyan M. Kumar

***Abstract*— Resource allocation and management in an Internet of Things (IoT) paradigm requires precise request and response processing irrespective of its scalability support. Unpredictable traffic pattern and user density demands reliable offloading for handling user request traffic and service response. Considering the need for large-scale IoT in account of its interoperability and heterogeneous support, this manuscript introduces response-aware traffic offloading scheme (RTOS) for delay-sensitive user requests. This offloading scheme is supported by multivariate spline regression machine learning model for classifying traffic for reducing the failure rate. The splines are adaptive based on the classified traffic for performing independent and shared offloading. The computation process for determining the offloading model is inherited from the cyber physical system (CPS) coupled with the IoT-Cloud architecture. The information from the knowledge base and event logs are exploited for decision-making in employing the offloading method for the classified traffic. The simulation analysis of this scheme shows that it is effective in improving the request processing ratio and reducing processing, response time, and delay. The simulation is performed for the varying user density and traffic flows.**

***Index Terms*— Basis Function, Cyber Physical System, IoT, Multivariate Spline Regression, Traffic Offloading**

I. Introduction

Internet of Things (IoT) has emerged as a smart object enabling technology for meeting the demands of increasing user requests. The exceptional features of the IoT paradigm endorse platform flexibility, service scalability and ubiquitous access to resources due to incorporation of diverse information communication technologies (ICT) [1, 2]. With the growth of human population and service requirements, the need for extending IoT classes from supporting a small scale to large-resource environment becomes necessary. Though the scalability and flexibility of the decentralized communication and computation paradigm is improved, service response and quality of service compliance needs to be retained in a more reliable manner [3, 4]. Smart objects and ICT jointly construct the platform for IoT users to improve the on-demand communication, information sharing and process computations. Besides, IoT paradigm provides global access to resources through interoperability with cloud, mobile edge computing (MEC), fog nodes, smart sensors, etc. [2, 4]. Envisioning the information and user requests in digital form enables machine-to-machine, device-to-device, human-computer interaction, etc., depending upon the application area ranging from residential to commercial and healthcare industries [5].

The density of request and traffic in a large-scale IoT is unpredictable due to which processing and resource allocation failures are common [6]. The conventional computation process and resource allocation systems inherit the advantages of the attached infrastructure support for handling the varying traffic patterns [7]. For handling varying traffic patterns, MEC, cloud, and fog based architectures are introduced for IoT paradigm. The computations are performed in a distributed and shared manner between the associated systems [5, 8]. Offloading is a prominent process for suppressing the traffic and congestion in the ubiquitous environment. Storage and computation, architecture based offloading schemes have been designed for smoothing IoT traffic in a view to improve the performance and reliability of resource allocation and sharing systems [6, 9]. Task or workflow offloading follows the conventional scheduling process for handling overloaded requests and service responses. Delay-sensitive application requirements in the large-scale environment are to be responded without time-lapse to retain the service quality of the users [5, 8, 10].

Cyber Physical System (CPS) has evolved as an integral support for improving the service quality of autonomous and distributed communication environments. Diverse computing, communication, and control systems are unified under CPS that

Gunasekaran Manogaran is with University of California, Davis, USA (e-mail: gmanogaran@ieee.org).

Gautam Srivastava is with the Dept of Math and CSC, Brandon University, Canada (e-mail: srivastavag@brandonu.ca)

Bala Muthu is with VRS College of Engineering and Technology, India (e-mail: balaanand@vrscet.in)

S. Baskar is with Karpagam Academy of Higher Education, India (e-mail: basker.s@kahedu.edu.in)

P. Mohamed Shakeel is with the University of Technical Malaysia Melaka. (e-mail: shakeel@utem.edu.my)

Ching-Hsien Hsu is with Asia University, Taiwan (e-mail: robertchh@asia.edu.tw)

Priyan M K is with Middlesex University, UK (e-mail: P.Malarvizhikumar@ieee.org)

Ali Bashir is with the Department of Computing and Mathematics, Manchester Metropolitan University, UK (e-mail: dr.alikashif.b@ieee.org)

is coupled to the IoT paradigm [11, 12]. Integration of such control system aids multi-faced benefits for IoT by aiding dynamic and reconfigurable service support and reliable resource oriented functions. The benefits of CPS are directly inherited in the IoT environment for improving the service response, delay-sensitive processing, resource allocation, and heterogeneity and scalable service support [7, 13]. The benefit of the integration of CPS into IoT-cloud is the independent and concurrent processing ability and distributed access. CPS is employed for resource allocation, traffic smoothing, data analytics and representation, visualizing events, data management, service compliance, etc. [12, 14].

II. Related Works

Kim et al. [15] introduced adaptive job allocation scheduler (AJAS) for redistributing overloaded IoT tasks. Tasks are distributed on the basis of user behavior, resource performance and energy exhaustion rate. This kind of allocation improves the availability of devices for processing the overloaded tasks.

Energy-cost dependent task offloading in heterogeneous IoT is introduced by Jaddoa et al. [16]. Based on the cloud/ edge node state and available network connections, dynamic decisions are made for offloading the arriving task. This decision-making process is reliable in improving the energy-efficiency and ratio of processed tasks in the IoT-cloud-edge architecture.

Fair and energy-minimized task offloading (FEMTO) is designed by Zhang et al. [17] for improving the performance of fog-assisted IoT networks. Offloading of tasks is performed by considering energy utilization, average energy exploitation in the previous states, and the priority of the fog nodes. This offloading method improves the fairness index along with probability of o overloading under controlled energy consumption.

Sun et al. [18] projected energy and time efficient task offloading and resource allocation algorithm for IoT-fog-cloud architecture. This algorithm is designed to reduce the energy and cost of through local computation and transmission power allocation. This algorithm achieves less energy utilization and computation time with better energy efficiency cost.

Misra and Saha [19] proposed dynamic task offloading for software-defined network assisted fog paradigm supporting IoT. The offloading method follows reliable decision-making, node discovery and route selection for aiding better performance. Offloading problem is addressed as an integer linear program in order to reduce delay and energy usage.

In order to handle computation demanding IoT tasks, Stavrinides and Karatza [20] designed hybrid flow-flow scheduling method. This method is effective for fog assisted IoT architectures, to reduce the computation cost and to smooth the offloading process. The proposed offloading method achieves better cost and resource utilization.

Kim et al. [21] introduced adaptive packet scheduling using Q-learning (APA-QL) for handling variable traffic flows in IoT environment. Q-learning helps to gain knowledge of the gateway queue size for scheduling the unpredictable traffic packet flow. Therefore, the processing time, computation process is reduced in this method. This method requires update of Q-table that increases the stagnancy in processing packets.

To handle deadline and cost dependent cloud workflows, Ma et al. [22] designed and IoT based effective task scheduling scheme. This task scheduling scheme identifies the performance of the virtual machine for satisfying the cost and delay constraints of the workflows. The allocation and scheduling process is facilitated through genetic process in order to improve the diversity of scheduling process.

Zhang et al. [23] employed cuckoo search algorithm for resource scheduling in IoT. The process of resource scheduling and allocation is differentiated from the premature convergence issue through Cauchy mutation. This helps to accelerate the number of local search, and the global solutions for scheduling. The heterogeneous resource scheduling algorithm improves resource utilization and relativity of user requests.

Boveiri et al. [24] employed min-max ant system (MMAS) for optimal task scheduling in cloud assisted IoT applications. This system employs traditional ant colony optimization for scheduling tasks over multi-processor systems. The design aim of this system is to identify the priority of the tasks for allocating appropriate systems for processing.

For edge assisted IoT services, Alameddine et al. [25] designed dynamic task offloading and scheduling (DTOS) method. This scheduling method is reliable in reducing the complexity in allocating resources through logic-based benders decomposition (LBBD). In this problem solving method, the primary and secondary computation models are useful in achieving optimal solutions for delay-sensitive applications.

Luo et al. [26] introduced adaptive task offloading (ATO) method for improving the performance of mobile edge computing paradigm in industrial applications. Aided by CPS, this offloading method is based on auction to meet the adaptability and security requirements of the edge server. The offloading method is found to achieve high resource utilization and controlled delay.

Energy cost and congestion issues in CPS based edge computing are jointly optimized by Yang et al. [27]. Promoted-by-probability (PBP) scheme is used in this optimization model for overcoming the fore-mentioned issues by classifying packets based on priority. Krill Herd Meta heuristic optimization algorithm used in the PBP helps to reduce the complexity of edge computing systems.

Yin et al. [28] designed a real-time task processing method using the collaboration of edge computing and CPS. High speed processing and response are the design objective of this collaborative architecture where task and control flow between the communication and CPS layers are handled independently. Queue aware scheduling and task assignment of this method helps to reduce delay and to improve the resource utilization rate.

III. Response-aware Traffic Offloading Scheme (RTOS)

The design goal of RTOS is to improve the reliability of request processing and thereby to improve the response service

quality in large-scale IoT. The computation model required to concise requests and response is distributed among the CPS-coupled IoT systems. CPS assists recommendation in handling responses for the overloaded and congested user requests. Divided by time and service based user demands, response is spontaneous satisfying the delay-less and backlogs-free compliance. The request traffic is in an independent and shared manner as distinguished by the capacity of the IoT computing system. The process of RTOS is portrayed in Figure 1.

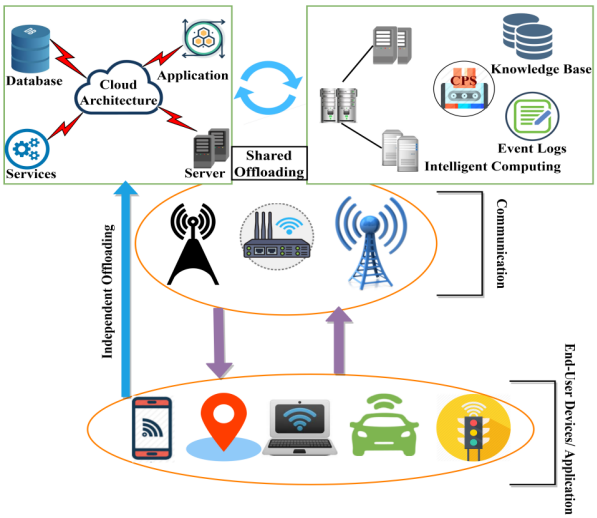

Figure 1 RTOS Process in IoT-Cloud-CPS Architecture

In Figure 1, RTOS process is illustrated in an IoT-Cloud-CPS architecture. The process of Traffic offloading is independent between the end-user devices and cloud architecture. In case of a shared offloading, CPS offers traffic handling features. Therefore, the offloading is determined with the aid of two-assistive architectures of IoT. End-user devices/ applications interact with the cloud/ CPS through the communication infrastructure. This architecture is designed for large-scale IoT request-response management, where traffic due to request and resource (service) is common, unattended/ non-smoothed traffic results in failure of request processing, delayed response and less feasible quality of service. In the following section, the rate of traffic generated and the appropriate offloading process is discussed in detail.

Traffic Generation Rate

The density (ρ) of users vary with different instance of time where in $\{r_1, r_2, .., r_\rho\}$ is the number of requests generated. The transmission time (t_t) of r_p varies in the processing time (p_t) of the IoT service provider. In a large-scale IoT, definite time slots for r_p transmission cannot be predicted or formulated as the status (completed/pending) of the previously generated requests relay on the processing capacity (p_c) of the $'m'$ service providers. Therefore, some requests from the previous t_t are carried forwarded in the successive processing time intervals. The maximum rate of traffic (requests) generated in $\sum t_t$ is $(\rho \times r_p \times t_t)$ for which the rate of response needs to be allocated in precise time. The maximum time until a user can wait is the validity time (t_v) of the request. If the response time (t_r) exceeds t_v, then it is said to be a failure. Therefore, allocating appropriate resources for the received request traffic with excess wait time or failure defaces the quality of service. The traffic flow of the requests in a time t_t is computed by equation

$$\left.\begin{aligned} \emptyset_r = \left\lceil \frac{\rho \times r_\rho \times t_t}{\left(r_\rho - \frac{\rho}{r_p}\right) \times t_v} \right\rceil \\ such\ that \\ (t_t + t_r) \leq t_v \end{aligned}\right\} \quad (1a)$$

The traffic of the requests $(\emptyset_r)$ is estimated for a single flow observed in t_t. Determining the traffic flow with respect to $\left(r_\rho - \frac{\rho}{r_\rho}\right)$, (the variation with respect to ρ and r_ρ is validated as a single IoT user/application is capable of generating multiple requests) helps determine the offloading type. Now, the objective of the proposed scheme is defined by equation (1b) as

$$\left.\begin{aligned} \forall \emptyset_r\ and\ p_c > 0\ \in t_t, \\ minimize\ (t_t - t_r) \forall\ independent\ offloading \\ minimize\ (t_t - t_r + t_w) \forall\ shared\ offloading \\ such\ that, \left(r_p - \frac{\rho}{r_p}\right) > 0\ in\ all\ t_t \end{aligned}\right\} \quad (1b)$$

In equation (1b), the factor t_w denotes the wait time of the r_p for a shared offloading process. The factor $\left(r_\rho - \frac{\rho}{r_\rho}\right) < 0$ indicated that no requests are generated from ρ users and therefore the device traffic needs not be accounted. The objective in equation (1b) emphasizes the minimization of $(t_t + t_r)$ and $t_t + t_r + t_w)$ to improve response rate. Therefore, the objective focusses to improve the response of all $\left(r_\rho - \frac{\rho}{r_\rho}\right) < 0\ \emptyset_r$ generated in t_t. The t_t rate of $\emptyset_r$ if balanced to be served within the next t_t sequence, and then response time is less. The decision making in balancing $\emptyset_r$ and satisfying equation (1b) is aided by adaptive regression spline learning. The following subsections describe the role of spline learning for independent and shared offloading process.

Independent Offloading

The process of independent offloading is carried out between the cloud resources through local decision-making. The decision-making focuses to maximize the response rate and to retain $(t_t + t_r) \leq t_v$. If the above conditions are satisfied, then the ratio of processed requests is enhanced. The available m service providers are to be scheduled to accept r_ρ in time t_t. Irrespective of the condition of $\left(r_\rho - \frac{\rho}{r_\rho}\right) > 0$, the arrived requests are to be responded. The p_c of the available m plays a vital role in determining t_r of r_ρ. However, there exists a tradeoff between p_c and r_ρ (allocated for m) due to the process start time (t_s) and t_r. This trade off factor (γ_r) is defined

using equation (2) as

$$\gamma_r = argmin\left[\frac{\left(r_\rho - \frac{\rho}{r_\rho}\right)\times(t_s - \frac{t_s}{r_\rho}\times m)}{(t_r - t_s) - \left(\frac{t_s}{r_\rho}\times m\right)}\right] + 1 - \left[\frac{\left(r_\rho - \frac{\rho}{r_\rho}\right)\times(p_c - \frac{t_r}{p_t}\times m)}{(t_r - t_s) - \left(\frac{t_s}{r_\rho}\times m\right)}\right] \quad (2)$$

In equation (2), the tradeoff between p_c and r_ρ is estimated with respect to the t_s and p_t. Hence, the decision for identifying r_ρ that exceeds t_v(in terms of time) is necessary. The conventional limit of requests for shared offloading lies between $\left[\left(\frac{t_r}{p_t}\times m\right), r_\rho\right]$ for $\emptyset_r$ received in t_t. The range of r_p is the worst-case in handling $\emptyset_r$ as the failure increases. However, reducing $\left(\frac{t_r}{p_t}\times m\right)$ and r_ρ increases the changes of reliable request processing. The sensitivity of retaining $(t_t + t_r) \le t_v$ is achieved by reducing γ_r and $\left(\frac{t_r}{p_t}\times m\right)$. The case of reducing r_ρ is achieved through shared offloading as $\left(\frac{t_r}{p_t}\times m\right) < r_\rho$. Instead, if $r_\rho = \frac{t_r}{p_t}\times m$, then u=independent offloading is sufficient. The decision in determining the range of the r_ρ reducing γ_r is progressed using adaptive splines. The required output as per the spline model for deciding $\left(\frac{t_r}{p_t}\times m\right)$ (i.e) $\left(Y_{r_\rho, t_r, p_t}\right)$ is given by equation (3a)

$$\left.\begin{aligned} Y_{r_\rho,\frac{t_r}{p_r}} &= (1-\Delta) + \sum_{i=1}^{r_\rho}(1-\Delta_i)B_i(r_\rho) \\ &where, \\ \Delta &= \left(r_\rho - \frac{\rho}{r_\rho}\right) and \\ B_i(r_\rho) &= \left(t_{r_1}\times r_{\rho_1}\right) + \frac{t_{r_2}}{2}\times r_{\rho_2} + \cdots + \frac{t_{r_{p_t}}}{p_t}\times r_{\rho_{p_t}} \end{aligned}\right\} \quad (3b)$$

Here, $B_i(r_\rho)$ is the basis function for the requests processed through p_t. In equation (3 a) and (3 b) the differential unit Δ is used for validating all the process of the independent offloading. For the estimation of $\left(\frac{t_r}{p_t}\times m\right)$ and the achievement of γ_r, the possibility of γ_r in $\emptyset_r$ is to be estimated with the consideration of Δ and p_c of the service provider. In this case, the tradeoff is computed as the different between time-lapse for the r_ρ received and service generated. This is represented as

$$\gamma_r = \begin{cases} m\left(\frac{t_r}{p_t} - \frac{p_t}{t_t}\right), if\ \Delta > 0 \\ 0, if\ \Delta < 0 \end{cases} \quad (4)$$

Substituting for γ_r from equation (2) and solving for 0 the RHS of equation (4)

$$\Delta \times \frac{\left(t_s - \frac{t_s}{r_\rho}m\right)}{\left(p_t - \frac{t_s}{r_\rho}m\right)} = \frac{(\Delta-1)\left(p_c - \frac{t_r}{p_t}m\right)}{\left(p_t - \frac{t_s}{r_\rho}m\right)} \quad (5\ a)$$

In equation (5 a), the p_t is substituted for $(t_r - t_s)$ time in m and considering $t_w = 0$. Similarly, $p_c \gg \frac{t_r}{p_t}m$, it can be neglected such that

$$\Delta\left(t_s - \frac{t_s}{r_\rho}m\right) = (1-\Delta)\frac{t_r}{p_t}m \quad (5\ b)$$

$$\Delta\left(t_s - \frac{t_s}{r_\rho}\right) = m\left(\frac{t_r}{p_t} + \frac{\Delta t_s}{r_\rho}\right) \quad (5c)$$

In the final value of γ_r in $\emptyset_r$ at a time span of $(t_t - t_r)$. The value is the trade off in equation (2) is reduced to the one represented in equation (5 c). If the t_s and p_t are assumed to be constant, then $\Delta\ \alpha m$. Now, the validation of range of $\left(\frac{t_r}{p_t}\times m\right)$ in $\emptyset_r$ for $Y_{r_\rho,\frac{t_r}{t_p}}$ is performed. From equation (3 b), the basis function is modeled with consideration of the limit as $\left(1, \frac{t_r}{p_t}m\right)$ such that,

$$B_i\left(\frac{t_r}{p_t}m\right) = \sum_{i=1}^{\frac{t_r}{p_t}m}\frac{t_{r_i}}{p_{t_i}}.(1-\Delta_i).r_{\rho_i} \quad (6\ a)$$

And

$$Y_{r_\rho,\frac{t_r}{p_t}} = (1-\Delta)\sum_{i=1}^{\frac{t_r}{p_t}m}(1-\Delta_i)\frac{t_{r_i}}{p_{t_i}}.r_{\rho_i} \quad (6\ b)$$

As the initial $\Delta = 0$ (before specifying the range of r_ρ),

$$Y_{r_\rho,\frac{t_r}{t_p}} = \sum_{i=1}^{\frac{t_r}{p_t}m}(1-\Delta_i)\frac{t_{r_i}}{p_{t_i}}.r_{\rho_i} \quad (6\ c)$$

The output of equation (6 c) denotes the limit of $\left(\frac{t_r}{p_t}m\right)$ in $\emptyset_r$ such that $(1-\Delta_i)$ is the range determining factor. The process of offloading $\left(\frac{t_r}{p_t}m\right)$ range of r_p is represented in Figure 2 (a) and (b)

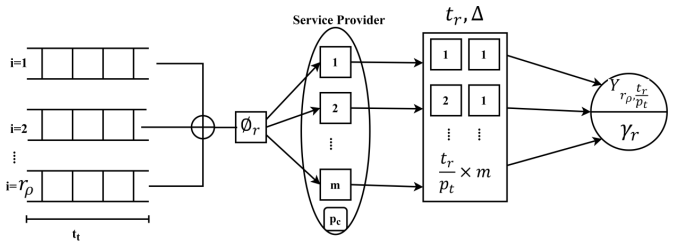

Figure 2 (a) Offloading for the range $\left[\mathbf{1}, \frac{t_r}{p_t}\times \boldsymbol{m}\right]$

The process in Figure 2(a) is a linear case, where the output of the spline is a multivariate, as $Y_{r_\rho,\frac{t_r}{p_t}}$ and γ_r. This multivariate needs to be approximated using Δ, in order to allocate maximum r_ρ to the available m in $(t_t - t_r)$ time. Therefore, the offloading range of r_ρ is computed as

$$offloading\ Range = \sum_{i=1}^{\frac{t_r}{p_t}m}\frac{\left(Y_{r_\rho,\frac{t_r}{p_t}}\right)_i^{-(\gamma r)_i}}{(\emptyset_r)i}x|(1-\Delta_i^2| \quad (7)$$

In equation (7), the offloading range for r_ρ is determined and therefore, the actual range is [offloading range (as in equation (7)), r_ρ]. As the independent offloading follows the range from first request to the range in equation (7), the r_ρ from $\emptyset_r$ that belongs to [1, offloading range] is the requests processed by m in $(t_t - t_r)$, $t_r > t_s$. The above range is estimated for $\emptyset_r$ in t_t provided γ_r is less, if the $(\frac{t_r}{p_t}xm)$is reduced to offloading range. This process is represented in figure 2(b).

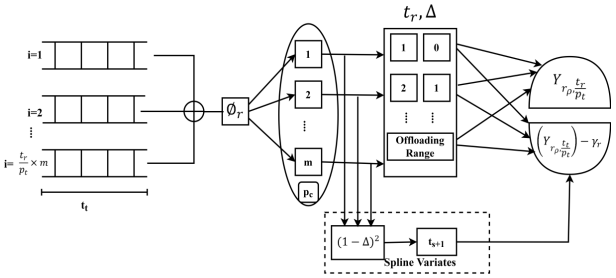

Figure 2 (b) Offloading for [1, offloading Range]

The representation in Figure 2(b) is non-linear due to the cross validation of multivariate spline factors $(1-\Delta^2$ and t_{s+1}(as Δ=0, in the first t_s) using the basis function. This offloading ensures $(t_t - t_r) \leq t_v$ or $t_r \leq t_v$ for the new offloading range. Hence, the processing of $[\emptyset_r x(1-\Delta^2) x \gamma_r]$ following equation (5c) is said to be high.

Shared Offloading

The other range of $\emptyset_r$ (i.e) [offloading range, r_ρ] is handled through this process. In this case, the objective is to reduce r_ρ, which means not all the r_ρ in time t_t is offloaded. This helps to reduce the unnecessary wait time and response delay of r_ρ. The available m is not computationally sufficient (in time p_t) for handling the new range of offloaded request traffic. Therefore, there are two concurrent processes in this offloading namely optimal reduction of the offloading limit and satisfying $(t_t - t_r + t_w)$ or $(p_t - t_w) \leq t_v$. This demands a multi level decision making that is facilitated by CPS knowledge base. The knowledge base and event log elements of CPS is exploited for handling the offloaded $\emptyset_r$. The previous state (independent offloading) r_ρ and γ_r are stored in the knowledge base and the event log held's the remaining $\emptyset_r$(i.e) $\left(Y_{r_\rho, \frac{t_r}{p_t}} - \gamma_r\right)$ request traffic to be handled. Until an offloading range variation is experienced, shared offloading is not facilitated. The first come first serve function of the service provider is modified in a dependent and discrete manner through the local decisions of CPS. The aim of such computation is to maximize the ratio of offloaded $\emptyset_r$ processing without increasing the p_t. The idle time of the m is identified using p_t and t_s to assign the segregated $\emptyset_r$from the independent offloading process. As identified in the previous case,) $\left(Y_{r_\rho, \frac{t_r}{p_t}} - \gamma_r\right)$is the offloaded traffic where in the $\emptyset_r$ experiences t_w until the appropriate m is assigned. The decision making is performed on the basis of differential adaptive spline factors (p_c, p_t) for satisfying the objective as in equation (1 b). the case of p_c being the same for all the available 'm' is assumed in this offloading process such that $p_c > 0 \in (t_r - t_s)$. Similar to the previous case, the γ_r minimization is considered for the remaining $\emptyset_r$. The basis function for (p_c, p_t) variates is estimated as

$$B_i(\partial) = \sum_{i=q}^{r_\rho} \frac{p_{t_i} xi}{p_c}\left(1 - \frac{i}{\emptyset_r}\right) \quad \text{(8a)}$$

For the above basis function, the constraint of $(p_t + t_w) \leq t_v$ is to be satisfied dialing which results in extended delay and response time. Therefore, the assignment (offloading) of the remaining $\emptyset_r$ follows the extended case of γ_r as given below: Considering that m service providers have different t_r, the estimated of γ_r (not 0) is:

$$\left.\begin{aligned} \gamma_r(1) &= m \times \frac{p_{t_1}}{p_c} + \sum_{i=1}\left(\frac{q_i}{\partial_i}\right) + \Delta \\ \gamma_r(2) &= m \times \frac{p_{t_2}}{p_c} + \sum_{i=2}\left(\frac{q_i}{\partial_i}\right) + \Delta_1 \\ &\vdots \\ \gamma_r(q) &= m \times \frac{p_{t_\partial}}{p_c} + \sum_{i=q}\left(\frac{q_\partial}{\partial_i}\right) + \Delta_{q-1} \end{aligned}\right\} \quad (9)$$

The estimation of γ_r for the ∂ follows p_c and p_t to improve the offloading rate, where in Δ_q is estimated as$\left(r_\rho - \frac{\rho}{r_\rho}\right) > 0$, provided $(t_w + p_t > t_v$. Therefore, $(\Delta_q \times \emptyset_r)$ is the rate of failed requests. The conventional offloading in presented in Figure 3, for linear ∂ management

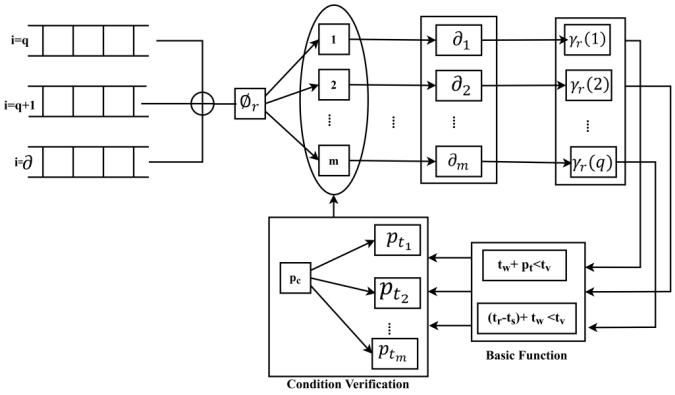

Figure 3 Linear ∂ Offloading

In Figure 3(a), the normalized and linear offloading of ∂ is represented. In this model, the condition $(t_w + p_t) < t_v$ or $[(t_r - t_s) + t_w] < t_v$ is verified at the each processing level of ∂, from offloading range to q. On the hand, this condition cannot be retained for a prolonged time as the t_r of the queued $\emptyset_r$ is less than the consecutive t_s. In this case, the increasing order of t_s and p_c of the m are considered to improve the swiftness in offloading. Based on the information from the knowledge base and event logs, the offloading in shared manner occurs. The event log is updated with q and ∂ whereas knowledge base holds $\gamma_r(q)$ and verifies the time based conditions for the offloaded range of $\emptyset_r$. The knowledge base helps to vary the offloading (different from first come first serve) process depending on t_v of the $r_\rho$and Δ of the previous $\gamma_r(q)$. In this method, the output for the remaining $\emptyset_r(i.e.)$ Y_∂ is estimated as

$$Y_\partial = \sum_{i=q}^{r_\rho} \frac{p_{t_i}}{p_c} \times \left[\frac{1-\gamma_r(q)_i}{q_i}\right] \quad \text{(10 a)}$$

and if $B_i(\partial) = B_i(q)$, *then*

$$Y_\partial = \sum_{i=q}^{r_\rho} \frac{p_{t_i}}{p_c} \frac{\gamma(q)}{q_i} \times (1 - \Delta_i) \quad \text{(10 b)}$$

The change in equation (10 b) from (10 a) occurs if $q = \partial$ provided t_s Y_∂ is less that the $(p_c \times p_t)$ of the m. Therefore,

the wait time of $\emptyset_r$ from q to ∂ be $\left[\frac{Y_\partial}{\emptyset_r}(1-\Delta^2)\times t_s - t_r\right]$. The traditional wait time of $(p_t - t_s)$ is reduced due to concurrent and shared offloading of ∂ as well under controlled wait time.

IV. Performance Analysis

The performance of the proposed RTOS is validated using simulations carried out in Contiki Cooja simulator. In coherence to the IoT-cloud-CPS architecture, a large-scale IoT environment comprising of 800 users is used in the simulation. The cloud is modeled with 8 servers each of 2 TB storage and 2×4GB physical memory. In this environment, the communication layer consists of 32 access points for connecting cloud, CPS and IoT user devices. In Table 1 (a) and 1 (b), the physical configuration of devices, cloud and CPS and simulation parameters and its values are discussed.

Table 1 (a) Physical Configuration

Configuration	Device	Cloud	CPS
Number	800	8	6
Physical Memory (Gb)	2-3	8	8
Storage Size (Gb)	16-64	2048	80
CPU (GHz)	1.3-1.8	3.1	2.8

Table 1 (b) Simulation Parameters and Values

Simulation Parameters	Value
Requests/ s	200-1000
Request Validity Time (ms)	210
Processing Capacity of m (requests/s)	500
Computation Memory of CPS (Mb)	3
Event Log Memory (Mb)	24
Bandwidth (Mb/s)	10

The above configurations are used for performing both independent and shared offloading process for the varying traffic (request) flows. The metrics that are computed in this experimental setup are: Processing time, processing ratio, average delay, and response time. The metrics are estimated for varying traffic flows, user density. For a consistency verification, the proposed RTOS is compared with the existing DTOS-LBBD [25], APS-QL[21], and ATO [26].

4.1 Analysis of Processing Time

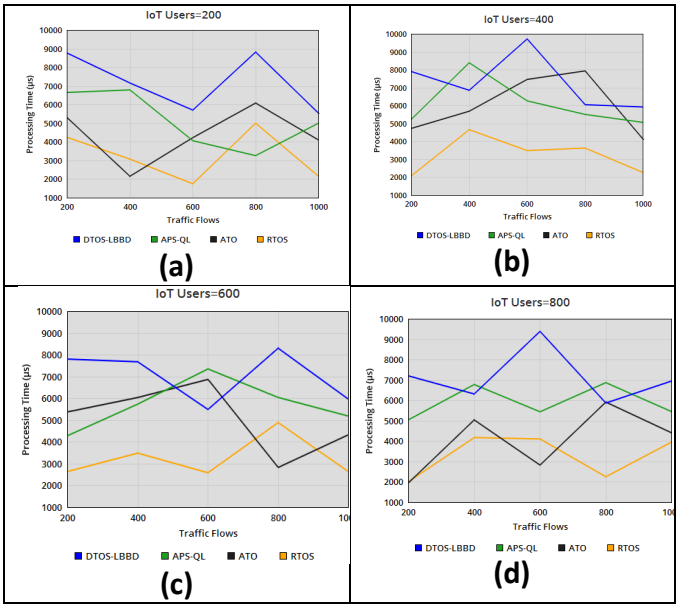

Figure 4 Processing Time

In Figure 4 (a)-4 (d), the comparative analysis of processing time with respect to varying traffic flows and users is presented. The joint $\emptyset_r$ processing time as in independent and shared offloading is estimated in the above presentation. The time required for processing r_ρ in $\emptyset_r$ in the independent offloading is $(t_s - t_r)$ and in the shared offloading case, it is $(t_w + p_t)$ as the offloading is linear. This is replaced with the new multivariate factors (p_c, p_t). The basis function confines the processing time to be not extended beyond t_v such that $(t_s - t_r)$ and $(t_w + p_t)$ in independent and shared offloading satisfies the condition of $\leq t_v$. Therefore, the processing time of shared offloading of ∂ is congruent with the $Y_{r_\rho, \frac{t_r}{p_t}}$ in $\emptyset_r$. Thus, the classified and approximated range of $\emptyset_r$ in both the offloading achieves less processing time. The processing time is balanced by altering the spline multivariate depending on ∂ and p_c for reducing additional time for Y_∂ .

4.2 Analysis of Processing %

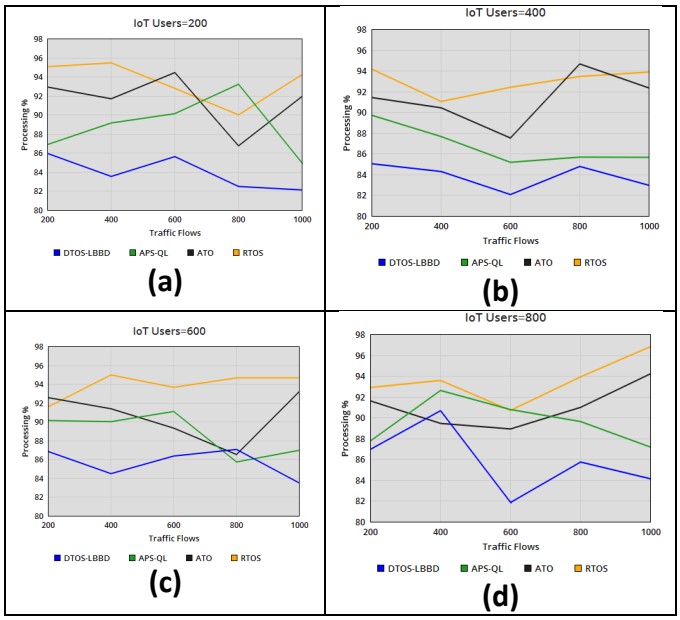

Figure 5 Processing Ratio

There are two factors that augment the ratio of processing requests namely the spline multivariate determination and offloading based on CPS knowledge base. In the former factor, the adaptive feature of the regression model is exploited by varying the multivariate $\left(\gamma_r, \frac{t_r}{p_t} m\right)$to (p_c, p_t) depending on the rate of $\emptyset_r$ processed. The classification and detection of $\emptyset_r$range through $B_i(r_\rho)$ helps to reduce the failure in processing $\emptyset_r$. The classification helps to maximize the chances of processing r_ρ in $\emptyset_r$ by defining the ranges. Therefore, it is common that processing ratio is better in the first offloading case. In Table 2, the comparative analysis of processing ratio with respect to the wait time (8ms to 16ms) is presented.

Table 2 Processing Ratio with respect to Varying Wait Time

Wait Time (ms)	DTOS-LBBD	APS-QL	ATO	RTOS
8	47.31	61.25	96.15	83.86
10	47.62	61.46	69.31	83.98
12	47.89	61.76	69.52	84.02
14	48.03	62.04	69.89	84.16
16	48.1	62.82	71.5	86.48

In case of shared offloading the objective in equation (1b) is satisfied by altering the first come first serve r_ρ processing to knowledge base and event log based recommendations. This recommendation considers p_c and $(t_t + p_w) < t_v$ constraint to assign all the $\emptyset_r$ from q to ∂ to the available m. Therefore, the rate of processing $\emptyset_r$ is high in the shared offloading process [Refer Figure 5 (a) - 5 (d)].

4.3 Analysis of Delay

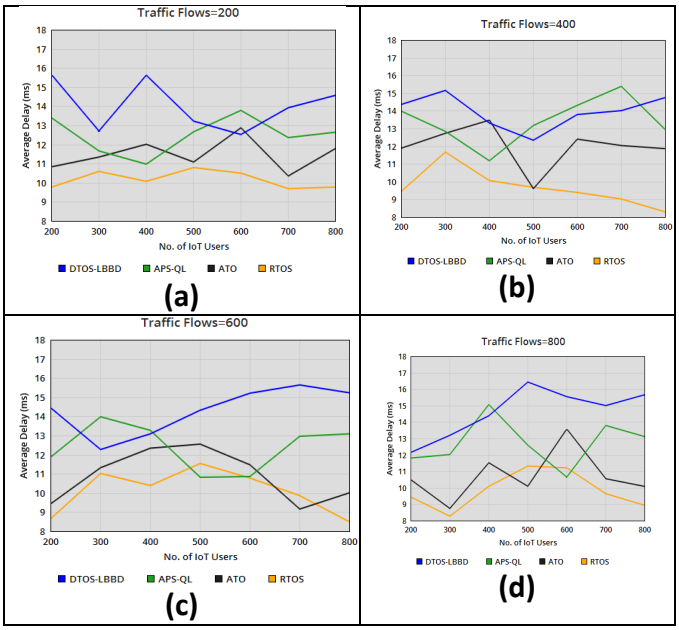

Figure 6 Average Delay

The time difference between t_t and t_r is computed as delay that includes t_w and p_t irrespective of $\emptyset_r$. The classification and range determination helps to reduce the number of failures in processing requests. Therefore, unnecessary wait time or carry forward of $\emptyset_r$ is not required in RTOS. For the shared offloading, the processing is performed by identifying $\gamma_r(q)$ and in independent offloading γ_r is identified for assigning p_c effective m such that objecticve (1b) is satisfied. The basis functions $B_i(\partial)$ and $B_i(r_\partial)$ is prominent in determining the $Y_{r_\partial, \frac{t_r}{p_t}}$ and Y_∂ as classified using the multivariate spline. Therefore, $\emptyset_r$ is operated using normal splines as in Figure 2 (a) and 2 (b) [based on range] and modified splines $p_c, p_t)$ as per Δ identification recommended by the knowledge base of the CPS. This differential processing using CPS and cloud resources retains the processing and response time of the $\emptyset_r$, reducing the delay [Refer Figure 6 (a)- 6 (d)].

4.4 Analysis of Response Time

Table 3 Response Time with respect to the Varying Traffic Flows

Traffic Flows	DTOS-LBBD	APS-QL	ATO	RTOS
200	8.85	8.02	6.07	5.12
400	8.96	8.14	6.19	5.11
600	9.05	8.22	7.81	6.11
800	9.15	8.59	6.91	6.19
1000	9.49	8.78	7.33	6.9

In the proposed RTOS, the response time is confined within t_v by classifying the range for $\emptyset_r$. The r_ρ is differentiated for independent and shared offloading on the basis of the splines and its multivariate. In the independent offloading process, γ_r is used to determine the time –lapse between t_r and t_s such that the processing time is confined. On the other hand, in the shared offloading process, the actual time of t_r is confined within the t_v by identifying $\gamma_r(q)$ and based on the recommendations of the CPS. The $\emptyset_r$ that is excess is offloaded to the m in the increasing order of t_s. Therefore, the time-laspse between two or more concurrent r_ρ in either of the offloading process is restricted with t_v, reducing the response time. Identifying Δ in both the cases adds to the minimization of t_r. The comparative tabulation of response time is presented in Table 3. Similarly, in Table 4 (a) and 4 (b), the comparative analysis of processing time and Processing ratio, and average delay is presented.

Table 4 (a) Processing Time for Varying Users

Metric	Users	DTOS-LBBD	APS-QL	ATO	RTOS
Processing Time (ms)	200	5535.162	4998.978	4097.115	2152.443
	400	5917.684	5058.718	4125.357	2264.468
	600	5959.517	5182.12	4322.801	2635.795
	800	6946.169	5445.171	4404.422	3955.88
Processing %	200	82.11	84.93	91.93	94.22
	400	82.95	85.65	92.33	93.87
	600	83.49	86.96	93.19	94.65
	800	84.12	87.16	94.21	96.82

Table 4 (b) Average Delay for varying Traffic Flows

Traffic Flows	DTOS-LBBD	APS-QL	ATO	RTOS
200	8.85	8.02	6.07	5.12
400	8.96	8.14	6.19	5.11
600	9.05	8.22	7.81	6.11
800	9.15	8.59	6.91	6.19
1000	9.49	8.78	7.33	6.9

V. CONCLUSION

This manuscript discusses response-aware traffic offloading scheme for improving the request processing rate of user-centric IoT paradigm. The proposed scheme initially classifies the traffic for independent and shared offloading to reduce the processing failures. Offloading process is supported by multivariate spline regression learning model for identifying the time-lapse to assign appropriate service providers to confine additional processing time and delay. Both the offloading process is monitored and supported using the computational recommendations of the CPS coupled with the IoT-Cloud architecture. This helps to improve the processing ratio if the requests and reduce processing time and delay for varying user density and traffic flows.

REFERENCES

[1] W.Yu, F. Liang, X. He, W. G. Hatcher, C. Lu, C. Lin, J and X.Yang, " A survey on the edge computing for the Internet of Things," IEEE access, vol.6,pp. 6900-6919, 2018.

[2] X. Min, X. Xu, Z. Liu, D. Chu, and Z. Wang, "An Approach to Resource and QoS-Aware Services Optimal Composition in the Big Service and Internet of Things," IEEE Access, vol. 6, pp. 39895–39906, 2018.

[3] C. Jian, M. Li, and X. Kuang, "Edge cloud computing service composition based on modified bird swarm optimization in the internet of things," Cluster Computing, 2018.

[4] J. Kang and D.-S. Eom, "Offloading and Transmission Strategies for IoT Edge Devices and Networks," Sensors, vol. 19, no. 4, p. 835, 2019.

[5] P. Gazori, D. Rahbari, and M. Nickray, "Saving time and cost on the scheduling of fog-based IoT applications using deep reinforcement learning approach," Future Generation Computer Systems, 2019.

[6] M. Jia, Z. Yin, D. Li, Q. Guo, and X. Gu, "Toward Improved Offloading Efficiency of Data Transmission in the IoT-Cloud by Leveraging Secure Truncating OFDM," IEEE Internet of Things Journal, vol. 6, no. 3, pp. 4252–4261, 2019.

[7] L. Lei, H. Xu, X. Xiong, K. Zheng, and W. Xiang, "Joint Computation Offloading and Multiuser Scheduling Using Approximate Dynamic Programming in NB-IoT Edge Computing System," IEEE Internet of Things Journal, vol. 6, no. 3, pp. 5345–5362, 2019.

[8] Y. Wen, Z. Wang, Y. Zhang, J. Liu, B. Cao, and J. Chen, "Energy and cost aware scheduling with batch processing for instance-intensive IoT workflows in clouds," Future Generation Computer Systems, vol. 101, pp. 39–50, 2019.

[9] J. Xie, Y. Jia, Z. Chen, Z. Nan, and L. Liang, "D2D Computation Offloading Optimization for Precedence-Constrained Tasks in Information-Centric IoT," IEEE Access, vol. 7, pp. 94888–94898, 2019.

[10] J. Ren, H. Wang, T. Hou, S. Zheng, and C. Tang, "Federated Learning-Based Computation Offloading Optimization in Edge Computing-Supported Internet of Things," IEEE Access, vol. 7, pp. 69194–69201, 2019.

[11] P. C. Santos, J. P. C. D. Lima, R. F. D. Moura, H. Ahmed, M. A. Alves, A. C. Beck, and L. Carro, "A Technologically Agnostic Framework for Cyber-Physical and IoT Processing-in-Memory-based Systems Simulation," Microprocessors and Microsystems, vol. 69, pp. 101–111, 2019.

[12] L. Qi, Y. Chen, Y. Yuan, S. Fu, X. Zhang, and X. Xu, "A QoS-aware virtual machine scheduling method for energy conservation in cloud-based cyber-physical systems," World Wide Web, 2019.

[13] K. Thramboulidis, I. Kontou, and D. C. Vachtsevanou, "Towards an IoT-based Framework for Evolvable Assembly Systems," IFAC-PapersOnLine, vol. 51, no. 11, pp. 182–187, 2018.

[14] R. Seiger, S. Huber, and T. Schlegel, "Toward an execution system for self-healing workflows in cyber-physical systems," Software & Systems Modeling, vol. 17, no. 2, pp. 551–572, Mar. 2016.

[15] H.-W. Kim, J. H. Park, and Y.-S. Jeong, "Adaptive job allocation scheduler based on usage pattern for computing offloading of IoT," Future Generation Computer Systems, vol. 98, pp. 18–24, 2019.

[16] A. Jaddoa, G. Sakellari, E. Panaousis, G. Loukas, and P. G. Sarigiannidis, "Dynamic decision support for resource offloading in heterogeneous Internet of Things environments," Simulation Modelling Practice and Theory, p. 102019, 2019.

[17] G. Zhang, F. Shen, Z. Liu, Y. Yang, K. Wang, and M.-T. Zhou, "FEMTO: Fair and Energy-Minimized Task Offloading for Fog-Enabled IoT Networks," IEEE Internet of Things Journal, vol. 6, no. 3, pp. 4388–4400, 2019.

[18] H. Sun, H. Yu, G. Fan, and L. Chen, "Energy and time efficient task offloading and resource allocation on the generic IoT-fog-cloud architecture," Peer-to-Peer Networking and Applications, 2019.

[19] S. Misra and N. Saha, "Detour: Dynamic Task Offloading in Software-Defined Fog for IoT Applications," IEEE Journal on Selected Areas in Communications, vol. 37, no. 5, pp. 1159–1166, 2019.

[20] G. L. Stavrinides and H. D. Karatza, "A hybrid approach to scheduling real-time IoT workflows in fog and cloud environments," Multimedia Tools and Applications, vol. 78, no. 17, pp. 24639–24655, 2018.

[21] D. Kim, T. Lee, S. Kim, B. Lee, and H. Y. Youn, "Adaptive Packet Scheduling in IoT Environment Based on Q-learning," Procedia Computer Science, vol. 141, pp. 247–254, 2018.

[22] X. Ma, H. Gao, H. Xu, and M. Bian, “An IoT-based task scheduling optimization scheme considering the deadline and cost-aware scientific workflow for cloud computing,” EURASIP Journal on Wireless Communications and Networking, vol. 2019, no. 1, Aug. 2019.

[23] C. Zhang, G. Zeng, H. Wang, and X. Tu, “Hierarchical resource scheduling method using improved cuckoo search algorithm for internet of things,” Peer-to-Peer Networking and Applications, vol. 12, no. 6, pp. 1606–1614, 2019.

[24] H. R. Boveiri, R. Khayami, M. Elhoseny, and M. Gunasekaran, “An efficient Swarm-Intelligence approach for task scheduling in cloud-based internet of things applications,” Journal of Ambient Intelligence and Humanized Computing, vol. 10, no. 9, pp. 3469–3479, Apr. 2018.

[25] H. A. Alameddine, S. Sharafeddine, S. Sebbah, S. Ayoubi, and C. Assi, “Dynamic Task Offloading and Scheduling for Low-Latency IoT Services in Multi-Access Edge Computing,” IEEE Journal on Selected Areas in Communications, vol. 37, no. 3, pp. 668–682, 2019.

[26] S. Luo, Y. Wen, W. Xu, and D. Puthal, “Adaptive Task Offloading Auction for Industrial CPS in Mobile Edge Computing,” IEEE Access, vol. 7, pp. 169055–169065, 2019.

[27] Y. Yang, Y. Ma, W. Xiang, X. Gu, and H. Zhao, “Joint Optimization of Energy Consumption and Packet Scheduling for Mobile Edge Computing in Cyber-Physical Networks,” IEEE Access, vol. 6, pp. 15576–15586, 2018.

[28] S. Yin, J. Bao, J. Li, and J. Zhang, “Real-time task processing method based on edge computing for spinning CPS,” Frontiers of Mechanical Engineering, vol. 14, no. 3, pp. 320–331, Jan. 2019.