A multi-user tasks offloading scheme for integrated edge-fog-cloud computing environments

Samuel D. Okegbile, *Member, IEEE,* Bodhaswar T. Maharaj, *Senior Member, IEEE,* and Attahiru S. Alfa

***Abstract*—This paper presents a multi-user, multi-class and multi-layer edge computing-based framework for effective task offloading and computation processes. Important system requirements that were not captured in the existing multi-layer solutions such as offloading, computations and deadline requirements were captured in the system modeling, while both wireless communications and task computation constraints were considered. We considered three layers system, where each device offloads its generated tasks in each time slot to any selected layer for computation. On its arrival at such a selected layer, the task is only accepted if the queue size is below the pre-defined threshold, otherwise, such a task is offloaded to the next layer. Tasks were classified into class 1 and class 2 tasks following tasks' quality of service requirements. We adopted stochastic geometry, parallel computing and queueing theory techniques to model the performance of the considered integrated edge-fog-cloud computing environment and obtained analysis for various performance metrics of interest. The obtained analyses demonstrate the importance of multi-layer and multi-class edge computing systems towards improving the experience of both delay-sensitive and mission-critical applications in any task offloading environment.**

***Index Terms*—Latency, mean throughput, mobile computing, parallel computing, queueing theory.**

I. Introduction

THE phenomenal increase of data traffic spurred by the continuous evolution of the internet of things (IoT) and 5G applications continues to pose unprecedented challenges especially when the system is evaluated with a focus on latency, capacity, scalability and reliability. This is a result of the limited computing and storage capacities of the connected devices as well as the less reliable communication channels among the heterogeneous IoT devices. These challenges are further aggravated by the increasing interactions among network nodes, random topology changes and network heterogeneity [1]. To reduce the effects of these challenges on the performance of the system, more powerful computing and storage capacities are needed to enhance task processing and storage as well as caching capabilities, such that both delay-sensitive and mission-critical applications can be efficiently supported.

S.D. Okegbile was with Department of Electrical, Electronic and Computer Engineering, University of Pretoria, South Africa and is currently with Department of Electrical and Computer Engineering, Concordia University, Canada. E-mail: samokegbile@gmail.com.

B.T. Maharaj is with Department of Electrical, Electronic and Computer Engineering, University of Pretoria, South Africa. E-mail: sunil.maharaj@up.ac.za.

A.S. Alfa is with Department of Electrical and Computer Engineering, University of Manitoba, Canada and Department of Electrical, Electronic and Computer Engineering, University of Pretoria, South Africa. E-mail: attahiru.alfa@up.ac.za.

Cloud computing is a promising technique that can compensate for the storage and computing resource limitations of the connected IoT devices through the deployment of cloud servers on the core network layer [2]. These cloud servers are known to be powerful in terms of computing and storage capacities and have been receiving a lot of attention in the last decades owing to their promising network architecture that has been demonstrated to be useful in facing the upcoming mobile data traffic deluge [3]. With the integration of the cloud computing framework, IoT devices can offload computation-intensive tasks to the cloud server via wireless networks. Task offloading to the cloud servers – which are generally known to be spatially far from devices – can however results in high transmission latency, data leakage and other related issues. As a result, these cloud-centric solutions may be unsuitable for various delay-sensitive and mission-critical applications. The resulting challenges with the cloud-centric solutions necessitate the need for other novel solutions through the deployment of computational nodes at the network edges to reduce the communication latency, while also offering an efficient computation resource capability.

Edge and fog computing models are considered as expansions of cloud computing model which allow improved processing and storage capabilities for IoT devices. Edge solutions (i.e. edge/fog computing) provide ubiquitous and low latency access to computing resources in IoT-enabled applications and systems. When deployed, IoT devices can offload their tasks to the nearby edge nodes/servers, which provide lower communication latency when compared to cloud computing. Unlike cloud computing, however, the computation capabilities of edge servers are rather limited. To address the limitations of cloud, edge/fog computing environments, recent efforts have considered the integration of these approaches. The evolving solution is capable of producing a massive improvement on the users' quality of service (QoS) as well as the overall system performance, thereby introducing higher flexibility for rapid computations as well as high mobility patterns [4].

An integrated edge-fog-cloud computing environment can facilitate a multi-layer offloading process, where multiple edge layers are created with different capabilities and locations from the devices – located at the users' layer. When a task computation request cannot be executed at any selected layer, such a task is forwarded to another layer with a suitable computation capability for better and faster computation, subject to its assigned deadline. Tasks computation requests that cannot be executed at the edges due to limited computation capability are forwarded to the cloud layer. Due to the cloud

layer's relatively unlimited computation capacity, each arriving task computation request is always accepted for possible computation. With this, task expirations at the cloud layer due to deadlines can be captured using queueing theory with reneging technique. The integrated edge-fog-cloud computing model is very useful as it can prevent computation overloading at the edge servers through an efficient task offloading process, while also supporting the mobility of nodes.

Unlike the traditional radio access network, modeling of multi-layer edge computing-based networks is more difficult owing to a more complicated task offloading and computation processes resulting from the existence of interference in the system [5]. This is further complicated by possible server failures and vacations at different layers. Since the goal of any offloading scheme is to reduce the network response latency, the performance of the integrated edge-fog-cloud computing network[1] can be evaluated by investigating the network's communication and computation latency. These metrics have been well investigated in single-layer point-to-point mobile edge computing (MEC) systems using the tool of queueing theory. The existence of interference in large-scale systems, however, means investigating such metrics in large-scale edge computing networks is much more challenging. As a result, a large-scale edge computing network with infinite nodes has not been well explored [5] and remains the motivation of the work presented in this paper. A multi-layer framework will improve task offloading efficiency in large-scale networks and ensure that delay-sensitive applications are computed before the expiration of their deadlines.

In large-scale multi-layer edge computing-based networks, task offloading and computation decisions may depend on different underlying parameters such as transmission costs, computation powers and energy consumption requirements. Another important parameter when deciding an appropriate point of execution[2](PoE) for any generated task is the task execution deadline. The execution/computation of any task should be completed before the expiration of its deadline [6]. Deadline, therefore, becomes a useful parameter when deciding appropriate PoE in a multi-layer system, although understanding its analysis as in a real-time system is very difficult. When the deadline is far enough in the future, a delay can be tolerated to grant delay-sensitive tasks faster access to computation resources. An accurate decision on appropriate PoE selection can improve the computation offloading performance in edge-fog-cloud computing environments.

In this paper, we considered a multi-user, multi-class and multi-layer edge computing-based environment where task offloading and computation decisions are achieved based on the tasks transmissions, computations and deadline requirements, while also considering both wireless communications and task computation constraints in the system modeling. The PoE of any arbitrarily generated task can either be at the user, edge, fog, or cloud layer depending on the computation requirements. The user's layer is selected as the PoE if the offloading power requirement cannot be satisfied by the originating device. Otherwise, such a task can be offloaded to either the edge or fog layer. A task that cannot be computed at the edge and fog layers due to either limited queueing capacity or the required computation capacity is offloaded to the cloud layer for faster computation access before the likely expiration of its deadline. We considered important factors such as priority and server failure in the analysis. To the best of our knowledge, such a multi-user, multi-class, and multi-layer edge computing-based problem has not been considered before. The contributions of this paper are thus summarised as follows:

- We investigated a novel approach to improve task offloading processes in large-scale systems by proposing a multi-class and multi-layer offloading solution. To capture such a multi-layer system, we proposed an integrated edge-fog-cloud computing system. This integrated edge-fog-cloud computing system was modeled using the tool of stochastic geometry (SG) and queueing theory, where spatial relationships between nodes were captured through point processes. Each layer of this multi-layer system was further considered to have multiple virtual machines (VMs) or servers.
- We then obtained a spatiotemporal analysis for the proposed framework and presented an analysis for connectivity probability – a metric that captures the fractions of devices with reliable communication links to base stations (BSs). Following the analysis of the connectivity probability, we modeled the proposed communication model and obtained tractable analysis for communication latency – a metric that captures the total time from the task generation period to the task arrival period at the selected PoE.
- Next, the task computation at each layer was characterized using queueing theory and parallel computing techniques to ensure simultaneous multiple task execution among VMs at each layer. The effect of the resulting input/output (I/O) interference was carefully captured in the system modeling.
- We explored task generation and arrival rates, task departure rates, task priority, and server failure to improve the offloading experience of delay-sensitive tasks, and obtained analysis for computation latency. We capture the expiration of the deadline using the queueing system with reneging technique and considered each server breakdown to be independent of other servers in the same layer. The classification of task priorities was achieved based on the expected task execution deadline.
- Finally, we carried out numerical simulations to evaluate and demonstrate the performance of the presented multi-layer scheme.

The remainder of this paper is organized as follows: Section II highlights important related literature. In Section III, the details of the proposed multi-layer edge computing-based network model were presented, while analyses for the communication model were presented in Section IV. Section V

[1]We used the terms integrated edge-fog-cloud computing network and multi-layer edge computing-based network interchangeably throughout this paper.

[2]In the considered multi-layer system, PoE is defined as the layer in which any selected task is executed.

presents the analysis of the computation model, while the outcomes of the numerical simulations were presented in Section VI. Section VII concludes this paper.

II. Related Works

Computation offloading is an efficient technique towards an effective reduction of the response latency in wireless applications and systems. It is therefore unsurprising that the area continues to receive a lot of attention recently. Single-user computation offloading problem was formulated in [7] as a branch and bound algorithm, while multi-user computation offloading problem was also formulated as a mixed-integer linear programming problem – a problem reported to be an NP-hard problem. Offloading processes were achieved via bandwidth, divided into multiple channels, where each channel is allocated to only one user. Each user was also considered to be orthogonal to others. Unlike single-user offloading problems, multi-user offloading problems do not only depend on the overhead it saves but also on the interference generated due to the activities of other users. This makes its analysis often difficult to obtain. To avoid complicated analysis, existing works often assume users' independent offloading schemes. Such an assumption however failed to capture the intrinsic interaction among users owing to the need among users to compete for computing resources. Independent offloading scheme is therefore insufficient.

A multiuser computation offloading problem was studied in [8] using the game theory technique in a multi-channel environment with interference. The authors in [9] similarly formulated a joint MEC offloading and D2D computation offloading process as a sequential game in a multi-user system. Effective computation offloading in multiuser computation offloading environment requires reliable radio resources. In orthogonal radio communication channels, devices located within the same cell transmit over orthogonal channels, while operations of devices located within different cells may interfere with each other [10]. Interference control is also important in non-orthogonal multiple access [3]. Hence, offloading decisions must consider the effects of interference, especially in large-scale MEC systems. This interference can be effectively characterized using various SG tools [11], [12]. In [13], task offloading decision was made by each device at each time slot. With such an approach, any device can either offload tasks to the edge server or execute such a task at its local CPU. Offloading decision was assumed to be made by the cloud manager in [10] for simplicity.

Modeling of large-scale MEC networks using SG was first attempted in [5]. The work adopted the concepts of SG, queueing theory and parallel computing to study the performance of large-scale single layer MEC networks. The data obtained by various IoT sensors were processed by the edge servers, which in return provide intelligent real-time solutions for various applications and systems. In a similar work, the analysis for outage probability for heterogeneous mobile cloud computing systems using SG was obtained in [14]. A spatial and temporal decisions algorithm was also formulated for edge-computing cloud-enabled heterogeneous networks in [6]. Through the formulated Markov decision process model, the appropriate computation site and time are decided. The performance of centralized cloud and edge computing applications was investigated in [15].

To improve the tasks offloading and computation rates, parallel computing technique is often adopted in wireless networks as in [5], [16]–[18]. Parallel computing can improve the runtime experience of tasks [18] while facilitating an energy-efficient design [19]. Service delay minimization method in the edge-cloud system was carried out in [20], while VMs migration in scalable MEC system was proposed in [21] to reduce delay. Traditional performance network models often ignore channel failures and recovery in network modeling and are generally known to overestimate network capacity and performances. Hence, the work in [16] considered network failures and recovery in cognitive radio networks. The analyses presented in [16] were later adopted in [17] to model server failures and repair in MEC-enabled wireless systems. The problem of feasible and dependable task execution, which often accounts for the joint limitation of network-wide mutual interference and parallel task computing by failure-prone VMs was also addressed. The work modeled both the failure and repair rates as independent Poisson processes, although offloading decisions and other important underlying parameters were not considered.

The continuous evolution of IoT and 5G applications toward the deployment of the smart environment means integration of centralized cloud and edge computing solutions is essential to meet the demands of the next-generation devices. Hence, the performance evaluation of a three layers cloud-fog-edge computing infrastructure through the adoption of queueing theory techniques was carried out in [4]. Task computation requests were assumed to arrive at edge nodes following independent and identically distributed (iid) Poisson processes, while task computation times at the edge, fog and cloud computing nodes were assumed to be distributed exponentially with a mean value. The associated deadline of each task computation request was further assumed to be exponentially distributed with a mean value. Similarly, holistic integration of cloud, fog and edge computing was introduced in [22], where fog computing paradigm meets the demanding requirements of various applications via multiple cloudlets collaborations, while edge computing provides edge-server-oriented service to achieve efficient processing with low latency.

While the works in [4], [22] introduced a novel approach of integrating cloud, fog and edge computing frameworks, important parameters that were not considered in such works include tasks priority, server failures, offloading and computation interference, etc. These parameters are central to a proper understanding of the multi-layer edge computing system. One useful approach towards achieving task priority is to categorize tasks based on their required deadlines. A similar approach was considered in [23], where users with lower latency requirements were considered to have non-preemptive priority over users with higher latency requirements. Multi-layer MEC scheme was also proposed in [24]–[26] and multi-service MEC scheme in [27], [28], where it was demonstrated that multi-layer and multi-service MEC scheme can reduce

the overall system latency while increasing the processing rate compared to the traditional edge networks.

In contrast to the existing solutions which failed to properly capture the important system requirements such as priority, server failure, offloading, and computation interference, in this paper, we concentrate on providing a multi-class and multi-layer solution towards improving the offloading scheme in large-scale environments. Tasks with low latency requirements (i.e., delay-sensitive tasks) are considered to possess non-preemptive priority over delay-tolerant tasks. Except otherwise defined, the definitions of common notations used throughout this paper are listed in Table I.

III. Network model

We considered a discrete-time multi-user, multi-class and multi-layer tasks offloading system where the time axis is segmented into unit intervals called slots t, $(\forall t = 1, 2, ...)$, while offloading attempts are synchronized. IoT devices can take advantage of the multi-layer environment by offloading their tasks to higher layers for efficient and faster computation opportunities, provided that the required total uplink and downlink transmissions power is less than the required computation power for local execution. The proposed environment is presented in Fig. 1, where the computation capacity $c = \{c_L, c_E, c_F, c_{Cl}\}$ satisfied the constraint $c_L < c_E < c_F < c_{Cl}$ given that parameters c_L, c_E, c_F and c_{Cl} represent the computation capacity of the local device, edge, fog and cloud servers respectively. A task computation request that cannot be executed at the local CPU of each device is offloaded for execution at the edge, fog, or cloud layer depending on the required computation capacity and subject to offloading power and task deadline requirements, since a lower computation delay can be achieved at the higher layer at the expense of higher transmission cost. The details of the proposed multi-layer edge computing system are discussed in the following subsections.

A. Spatial distributions model

The distributions of devices located at the users' layer are considered to follow homogeneous PPP (HPPP) Ψ_L of intensity λ_L, while similar to [4], [17], BSs within which edge servers were equipped at the edge layer are also assumed to be distributed following an independent HPPP Ψ_E of intensity λ_E. Similarly, the distributions of BSs located at the fog layer follow independent HPPP Ψ_F of intensity λ_F. The signal power attenuates at the rate of $r^{-\alpha}$ following unbounded path-loss propagation model, where r and $\alpha \geq 2$ represent the distance and path-loss exponent respectively. We also assumed that channel links between the users' layer and any selected higher layer (edge, fog, or cloud) are subject to Rayleigh fading such that the desired signal power gains h_d and interference signal h_I are exponentially distributed with unit mean, i.e. $h_d, h_I \approx \exp(1)$, while each of B uplink links is randomly selected by each device. We further adopted channel inversion power control, which allows each device to adjust its transmit power to ensure that the received uplink power level at any BS is equal to a predetermined power threshold ρ. We assumed that communications between edge, fog and cloud layers are achieved through dedicated error-free channels with negligible interference, while there is no loss of packets on such channels to avoid complicated analysis.

We modeled the service zones of BSs located at the edge layer using the bipolar model [29], where the service zone of each BS $x_i^E \in \Psi_E$ is considered to depend on its transmission signal power P_E. The radius of any selected BS $x_k^E \in \Psi_E$ service zone D_E can be obtained as $D_E = r_E(\frac{\Delta\theta_E P_M}{P_E})^{\frac{1}{\alpha}}$, where r_E is the distance between any tagged device $x_k^L \in \Psi_L$ and its corresponding BS $x_k^E \in \Psi_E$ located at the edge layer, θ_E is a predefined threshold for successful task reception at the edge layer, P_M is the transmit power of any device and Δ is the design factor [29]. The radius of any selected BS $x_i^F \in \Psi_F$ service zone in fog layer, given as D_F can also be obtained following the same process. Similar to [4], we assumed that the service zone of cloud services is relatively unlimited. Each device $x_i^L \in \Psi_L$ is located within the service zone of at least one edge node $x_i^E \in \Psi_E$ as shown in Fig. 1. Similarly, each edge node $x_i^E \in \Psi_E$ is located within the service area of at least one fog node $x_i^F \in \Psi_F$, while each fog node $x_i^F \in \Psi_F$ is located within the service area of cloud server. The service zone of any BS is defined to be the same as its coverage zone.

In a large-scale network, an IoT device may be located within the service zone of more than one BSs belonging to the same layer. In such a case, a typical device will be connected to its nearest BS for the task offloading process as in [11], [17]. Consider a typical BS located at the origin, the number of devices located within the service zone $\Omega(0, D_E)$ of such a typical BS $x_k^E \in \Psi_E$ in the edge layer can be captured through the contact distribution function [30]. The probability that there is at least one device located within $\Omega(0, D_E)$ is given as

$$P(\Omega, D_E) = 1 - \exp(\lambda_L \pi D_E^2), \forall D_E \geq 0. \tag{1}$$

Similarly, the probability that there is at least one device located within $\Omega(0, D_F)$ is obtained as

$$P(\Omega, D_F) = 1 - \exp(\lambda_L \pi D_F^2), \forall D_F \geq 0. \tag{2}$$

B. Task generation model

The task generation model for the proposed multi-layer edge computing system is modeled as a discrete-time queueing model[3]. Hence, task generation at each device is considered to follow an independent Bernoulli process with rate g tasks per unit slot. Similarly, task inter-arrival times were considered to follow geometric distributions. We assumed that there is no synchronization between devices. From devices' perspectives, task generations are assumed to occur at the beginning of any time slot t, while the offloading of a single task occupies a random number of slots $\mathbb{N} \geq 1$.

Some tasks are known to be delay-sensitive tasks in real-life applications, while some are delay-tolerant. It is therefore

[3]Discrete-time queueing models are better suited than continuous-time queueing models when analysing and designing digital transmitting and telecommunication systems [31].

TABLE I
COMMON NOTATIONS USED

Notation	Definition
ς	Any selected layer. $\varsigma = \{E, F, Cl\}$ for edge, fog and cloud layer respectively
$c; c_\varsigma$	Computation capacity; computation capacity of ς
$\Psi; \lambda$	Distribution of nodes; the intensity of nodes
$r; \alpha$	Distance between any two communicating nodes; path-loss exponent
$h_d; h_I$	Desired signal power; interference signal power
ρ	Uplink power threshold
$P_M; P_\varsigma$	Offloading power of devices; Offloading power of BS located at layer ς
$\theta_\varsigma; \Delta$	SIR threshold; design factor
$D_\varsigma; \Omega(0, D_\varsigma)$	Service zone radius of any BS; service zone of a BS in layer ς
$g; g_n$	Task generation rate; class n task generation rate
$\tau; \eta$	Length of deadline period; the occurring rate of τ
$H_\varsigma; l_\varsigma$	Queue capacity at layer ς; queue size at layer ς
$K_\varsigma; Pr_\varsigma$	Number of VMs in layer ς; the probability of selecting layer ς
$m_\varsigma; d_\varsigma$	Queueing system efficiency threshold; server degradation factor at layer ς
$\phi; v_\varsigma$	Reliability threshold; the weight of task acceptance at layer ς
$\rho_\varsigma; \rho_{n,\varsigma}$	System load at layer ς; system load of class n task at layer ς

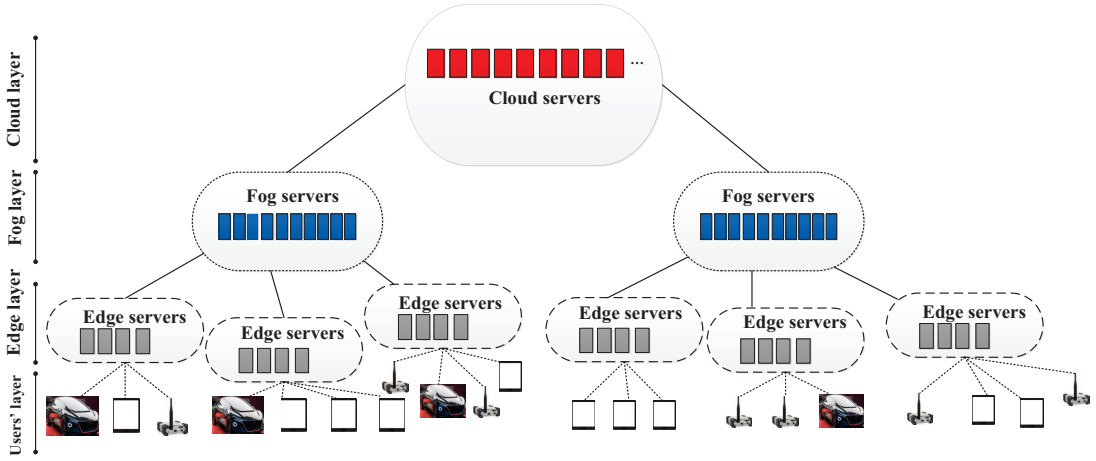

Fig. 1. Multi-layer edge computing environment.

important to classify tasks based on their delay requirements to ensure that the QoS of each task is satisfied [32]. With task classifications, different priority levels could be associated with different task requests [10]. Preferably, any generated task should be computed before the expiration of its deadline (if there is any). Hence the priority level can be assigned to tasks based on the latency requirements. As in [23], tasks with shorter deadlines (i.e., lower latency requirements) were classified as delay-sensitive tasks, while tasks with longer deadlines were classified as delay-tolerant tasks. Delay-sensitive tasks (referred to as class 1 tasks) were thus considered to have non-preemptive priority over delay-tolerant tasks (referred to as class 2 tasks). The joint task generation rate at each device can then be captured as $g = \{g_1 \cup g_2\}$, where g_1 and g_2 represent independent task generation rates of class 1 and class 2 tasks respectively[4], given that $g_1 \ll g_2$. With $g_1 \ll g_2$, the QoS requirements of class 2 tasks can be satisfied.

After being generated, a task has until the expiration of its deadline, a certain length of time slot τ for its execution. At the expiration of τ, an unprocessed task leaves the system (a process known as task reneging). Generally, the deadline τ can be represented as a random variable in the range $[0, \infty)$ and its probability density function can be represented as

$$f(t) = \eta \exp(-\eta t), \forall \eta \geq 0, t \geq 0, \tag{3}$$

where η is the occurring rate of deadline τ, while $\frac{1}{\eta}$ gives the average length of τ. A task is classified as class 1 if $\tau \leq d_t$ and as class 2 if otherwise. To enhance fairness in real practical systems, the threshold d_t should be constantly defined by an

[4]It is often assumed that delay-sensitive tasks are rare in real-life compared to delay-tolerant tasks, hence the reason for the condition $g_1 \ll g_2$. The model, however, remains valid for condition $g_2 > g_1$ with little modifications.

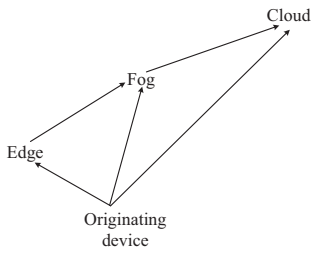

Fig. 2. Tasks offloading and transferring processes.

independent external application. Such a threshold can depend on the total required computation time of tasks waiting in service and is beyond the scope of this paper.

C. Tasks offloading and acceptance modeling

A newly generated task is offloaded to any one of the edge, fog and cloud layers at the beginning of any time slot t if the offloading requirement is satisfied at the originating device (i.e., the originating device can produce the required offloading power necessary for a successful reception at the selected PoE). The offloading process in terms of layer selection is shown in Fig. 2. Such a task is offloaded to an edge layer if the required computation capacity $c \leq c_E$ and to a fog layer if $c_E < c \leq c_F$. The task is offloaded to a cloud layer if $c > c_F$. Given that a typical device selects an appropriate layer based on the required computation capacity c, the probability of selecting each layer as a potential PoE is given as

$$Pr_{\varsigma} = \begin{cases} Pr_E = P(c \leq c_E), & \text{if selecting edge layer} \\ Pr_F = P(c_E < c \leq c_F), & \text{if selecting fog layer} \\ Pr_{Cl} = P(c > c_F), & \text{if selecting cloud layer,} \end{cases} \tag{4}$$

where Pr_E, Pr_F, and Pr_{Cl} are the probabilities that a typical device will select edge, fog and cloud layer respectively for its computation, with $Pr_E + Pr_F + Pr_{Cl} = 1$.

The queues at the edge and fog layers are considered to be of finite capacities of sizes H_E and H_F respectively, where task execution within each class follows a first come first serve (FCFS) scheduling policy. Owing to its unlimited capacity and faster computation process, the queue at the cloud layer is considered to be of infinite capacity. When the offloading requirement cannot be satisfied by the originating devices, any generated tasks can only be computed locally. In this paper, we only focus on situations where the offloading requirements are satisfied. Hence the analysis presented focuses on edge, fog and cloud layers. We also assumed that all devices are always aware of the computation capacities of the edge, fog and cloud servers.

A task is accepted for computation at the edge layer if the queue size l_E satisfies the condition $l_E < m_E \leq H_E$. Such a task joins the queue at the edge layer and waits for computation opportunity. Otherwise, an unaccepted task is transferred to the fog server by the edge server for computation. Such a task is accepted at the fog layer if the queue size l_F at the fog layer satisfies the condition $l_F < m_F \leq H_F$, otherwise, the task is transferred to the cloud layer. The queue capacities H_E and H_F are selected to ensure all accepted tasks are computed before the expiration of their deadlines[5]. We assumed that edge, fog and cloud layers are made up of K_E, K_F and K_{Cl} number of virtual machines (VMs) respectively. Owing to the non-preemptive priority possessed by class 1 tasks, any class 1 task, upon arrival at any layer, is immediately moved to the head of the queue for immediate computation. We assumed that each accepted task has different sizes; hence each task requires a random number of slots for computation.

One metric of interest, in this case, is therefore the acceptance probability – defined as the probability that an offloaded task is accepted at the selected layer. Generally, an acceptance probability at the edge layer is 1 when the queue size l_E is less than the predefined threshold $m_E \leq H_E$. Similarly, the acceptance probability at the fog layer is 1 when the queue size l_F is less than the predefined threshold $m_F \leq H_F$. This acceptance probability is known to decrease with an increase in the queue size [33]. Due to unlimited capacity at the cloud layer, the acceptance probability is always 1 at the cloud layer. For $\varsigma \in \{E, F\}$, the acceptance probability can be summarized as

$$\varphi_{l_{\varsigma}} = \begin{cases} 1, & l_{\varsigma} \leq m_{\varsigma} \\ 1 + (\ln\{\frac{H_{\varsigma}}{m_{\varsigma}}\})^{-1} \ln\{\frac{m_{\varsigma}}{l_{\varsigma}}\}, & m_{\varsigma} < l_{\varsigma} \leq H_{\varsigma}. \end{cases} \tag{5}$$

At $l_{\varsigma} = H_{\varsigma}$, $\varphi_{l_{\varsigma}} = 0$. The proposed task offloading scheme is summarized in Algorithm 1.

D. Tasks arrival rate

A careful observation of the proposed multi-layer systems shows that the task arrival rate at each layer – which is also considered to follow the Bernoulli process – not only depends on the task generation rate and acceptance probability but also the offloading success probability. This offloading success probability at any arbitrary layer $\varsigma \in \{E, F, Cl\}$ can be obtained in terms of signal-to-interference ratio (SIR) as $O_{\varsigma} = P(SIR_{\varsigma} > \theta_{\varsigma})$. Since offloading success probabilities in different slots are independent, there is no need for time index t. Generally, the task arrival rate of class n tasks at the edge layer when φ_{l_E} is neglected is given as

$$a_{n,E} = Pr_E O_E g_n, \forall n = \{1, 2\}. \tag{6}$$

Similarly, given that a task is received at the fog layer from the edge layer owing to full queue capacity at the edge layer (i.e., $l_E \geq H_E$ with probability $\pi_{H_E} = \sum_{k=1}^{\infty} \pi_{H_E}(k)$, the task arrival rate of class n tasks at the fog layer when φ_{l_F} is neglected is given as

$$a_{n,F} = Pr_F O_F g_n + a_{n,E} \pi_{H_E}, \forall n = \{1, 2\}, H_E \geq 1. \tag{7}$$

Finally, the task arrival rate of class n tasks at the cloud layer given that such a task is received at the cloud layer owing to

[5]Note that the parameters that matter are m_E and m_F. The parameters H_E and H_F are, however, necessary to represent the finite capacities of the queues. This is useful to show that, the system becomes unstable when $m_{\varsigma} > H_{\varsigma}$, for $\varsigma \in \{E, F\}$, due to the number of waiting tasks.

Algorithm 1: Task offloading scheme

Initialization: φ_{l_ς}, m_E, m_F, H_E, H_F
End Initialization
Input: $x_k^L \in \Psi_L$ and c
while *task computation is required* **do**
  if $c < c_E$ **then**
    $x_k^L \in \Psi_L$ offloads to layer $\varsigma = \{E\}$;
  end
  else if $c_E < c < c_F$ **then**
    $x_k^L \in \Psi_L$ offloads to layer $\varsigma = \{F\}$
  end
  else
    $x_k^L \in \Psi_L$ offloads to layer $\varsigma = \{Cl\}$
  end
  For any packet at $\varsigma = \{E, F\}$:
  if $l_\varsigma < m_\varsigma < H_\varsigma$ **then**
    $\varphi_{l_\varsigma} = 1$;
  end
  else if $m_\varsigma < l_\varsigma < H_\varsigma$ **then**
    $\varphi_{l_\varsigma} = 1 + (\ln\{\frac{H_\varsigma}{m_\varsigma}\})^{-1} \ln\{\frac{m_\varsigma}{l_\varsigma}\}$
  end
  else
    Forward the packet to the next layer
  end
end

full queue capacity at the fog layer with probability $\pi_{H_F} = \sum_{k=1}^{\infty} \pi_{H_F}(k)$ is obtained as

$$a_{n,Cl} = Pr_{Cl} O_{Cl} g_n + a_{n,F} \pi_{H_F}, \forall n = \{1, 2\}, H_F \geq 1. \quad (8)$$

We adopted an early arrival system policy where departures occur immediately before the slot boundaries in the interval $(t-, t)$, while arrivals occur immediately after the slot boundaries in the interval $(t, t+)$. From (5)–(8), we know that the overall joint tasks arrival rate a_ς at each layer ς, for $\varsigma \in \{E, F\}$, considering the effect of φ_{l_ς} is given as

$$a_\varsigma = \varphi_{l_\varsigma} \sum_{n=1}^{2} a_{n,\varsigma} =$$

$$\begin{cases} \sum_{n=1}^{2} a_{n,\varsigma}, & l_\varsigma \leq m_\varsigma \\ \sum_{n=1}^{2} a_{n,\varsigma} (1 + (\ln\{\frac{H_\varsigma}{m_\varsigma}\})^{-1} \ln\{\frac{m_\varsigma}{l_\varsigma}\}), & m_\varsigma < l_\varsigma \leq H_\varsigma. \end{cases} \quad (9)$$

For $\varsigma \in \{Cl\}$, the joint arrival rate $a_\varsigma = a_{Cl}$ is simplified as

$$a_{Cl} = \sum_{n=1}^{2} a_{n,Cl}. \quad (10)$$

E. Tasks reneging rate

It is also important to investigate the reneging rate of any arbitrarily accepted task due to the expiration of its deadline. From (3), we know that arrivals and departures of tasks due to deadline expiration are independent. Therefore, it is safe to capture the reneging rate as a function of the queue size. An unprocessed task is said to have reneged if such a task leaves the system after a certain period τ. When the queue size $l_\varsigma \leq m_\varsigma$ for $\varsigma \in \{E, F, Cl\}$, we assumed that no task renege as the expected total computation time is expected to be less than τ. However, when $l_\varsigma > m_\varsigma$, the reneging behavior can be said to be an exponentially distributed random variable with parameter γ_ς. The reneging rate can be obtained as

$$\gamma_{l_\varsigma} = \begin{cases} 0, & l_\varsigma \leq m_\varsigma \\ \gamma_\varsigma (l_\varsigma - m_\varsigma), & m_\varsigma < l_\varsigma \leq H_\varsigma. \end{cases} \quad (11)$$

The probability that any arriving task will renege after acceptance is provided in Section V.

F. Tasks computation rate

We considered the mean computation time at each layer to follow independent general distributions, where the service times at each VM are iid random variables. Each of these VM is subject to failure or breakdown following the Bernoulli process. Hence, the probability that a typical VM is available during any time slot is given as $p \in (0, 1]$. With the adoption of parallel computing at each layer, each VM suffers from I/O interference from other VMs that must be captured in the system modeling. Let the task execution rate of a single available VM in layer ς be given as $\mu_{0,\varsigma}$ tasks per unit slot, the task execution rate of such a VM when multiple VMs are involved, given that pK_ς VMs are available at any selected layer can be obtained as

$$\mu_\varsigma^1 = \frac{\mu_{0,\varsigma}}{(1 + d_\varsigma)^{pK_\varsigma - 1}}, \quad (12)$$

where d_ς depicts the computation degradation factor owing to I/O interference among pK_ς VMs. Degradation factor $(d_\varsigma \geq 0)$ is defined as the percentage increase in the expected service time of a task, experienced by a VM when multiple VMs are multiplexed. From (12), the mean execution rate at any selected layer can be approximated as

$$\mu_\varsigma = p\mu_\varsigma^1. \quad (13)$$

IV. Analysis of Communication Model

In this section, we analyze the communication model to investigate the performance of the proposed multi-layer edge computing system. We first present the analysis for the connectivity probability followed by an analysis for communication latency.

A. Connectivity model

The connectivity model captures the coverage of radio access networks in the proposed multi-layer edge computing system at the network level. This model follows from the analysis of offloading success probability and is useful in measuring the fraction of devices with reliable links/connections to BSs. When any tagged device $x_k^L \in \Psi_L$ offloads its tasks to any selected BS $x_k^\varsigma \in \Psi_\varsigma$, the probability that such a task is successfully received depends on the received SIR at the paired BS, considering an interference-limited channel. Let $\vartheta^t_{x_k^L,\varsigma} \in \{0, 1\}$ represents the user-server association indicator in time slot t, such that $\vartheta^t_{x_k^L,\varsigma} = 1$ indicates that a device x_k^L

offload its task to layer ς in time slot t and $\vartheta^t_{x^L_k,\varsigma} = 0$ indicates otherwise. We assume that each device can only offload to one layer (for computation at a single VM) at a time. The user-server association rule can be obtained as

$$\kappa = \begin{cases} \vartheta^t_{x^L_k,\varsigma} \in \{0,1\}, & \forall x^L_k \in \Psi_L, x^\varsigma_k \in \Psi_\varsigma \\ \sum_{x^\varsigma_k \in \Psi_\varsigma} \vartheta^t_{x^L_k,\varsigma} = 1, & \forall x^L_k \in \Psi_L. \end{cases} \quad (14)$$

The received SIR at any tagged BS $x^\varsigma_k \in \Psi_\varsigma$ located at layer $\varsigma = \{E, F, CL\}$ is obtained as

$$SIR_\varsigma = \frac{\rho h_d}{\sum_{x^L_i \in \Psi_L \setminus x^L_k} \vartheta^t_{x^L_i,\varsigma} P_M h_I ||z||^{-\alpha}}, \quad (15)$$

where $||z||$ is the Euclidean distance between an active device $x^L_i \in \Psi_L$ except the tagged device $x^L_k \in \Psi_L$ and the tagged BS. From (15), the connectivity probability is obtained as the probability that the received SIR_ς is greater than the pre-defined signal threshold for uplink transmission. This is obtained as

$$P^\varsigma_{conn} = O_\varsigma = P\Big(\frac{\rho h_d}{\sum_{x^L_i \in \Psi_L \setminus x^L_k} \vartheta^t_{x^L_i,\varsigma} P_M h_I ||z||^{-\alpha}} > \theta_\varsigma | \Psi_L, \Psi_\varsigma\Big). \quad (16)$$

We make two key assumptions next.

Assumption 1: Stationary point process is assumed such that the location of each device within each observation period remains unchanged.

Remark: This assumption has been widely adopted when modeling wireless communications to avoid complicated analysis and has been verified through simulations. Hence, the network topology remains unchanged in each time slot and independent of the other time slots [34]–[37].

Assumption 2: The transmission powers of active devices are independent.

Remark 2: Note that the transmission powers of active devices are dependent owing to the adopted channel inversion power control. However, to ensure tractable analysis, we neglect such dependence. This assumption has been validated in [38], [39]. Following Assumption 1 and Assumption 2, the connectivity probability at any selected layer is given in Lemma 1.

Lemma 1: Given that any tagged device located at the users' layer offloads its generated tasks to any selected layer $\varsigma = \{E, F, Cl\}$ and that $\frac{\lambda_L}{\lambda_\varsigma}$ is the average number of devices connected to each BS $x^\varsigma_i \in \Psi_\varsigma$, the connectivity probability at the selected layer is straightforward from (16) and can be obtained as

$$P^\varsigma_{conn} = \frac{\exp\Big\{\frac{-2\theta_\varsigma g P r_\varsigma \frac{\lambda_L}{\lambda_\varsigma B}}{\alpha - 2} {}_2F_1(1, 1 - \frac{2}{\alpha}, 2 - \frac{2}{\alpha}, -\theta_\varsigma)\Big\}}{(1 + \frac{\theta_\varsigma g P r_\varsigma \frac{\lambda_L}{\lambda_\varsigma B}}{(1+\theta_\varsigma)u})^u}, \quad (17)$$

where ${}_2F_1(.)$ represents the Gaussian hypergeometric function, while $u = 3.575$ captures the approximation of the probability distribution function of the PPP Voronoi cell area in Euclidean space R^2. At $\alpha = 4$, (17) is simplified as

$$P^\varsigma_{conn} = \frac{\exp\{-g P r_\varsigma \frac{\lambda_L}{\lambda_\varsigma B} \sqrt{\theta_\varsigma} \arctan(\sqrt{\theta_\varsigma})\}}{(1 + \frac{\theta_\varsigma g P r_\varsigma \frac{\lambda_L}{\lambda_\varsigma B}}{(1+\theta_\varsigma)u})^u}. \quad (18)$$

Proof: Since tasks offloading from devices are unscheduled, inter-cell and intra-cell interference will affect the offloading process of the tagged device. This analysis follows from approximating the locations of interfering devices with a PPP of the same intensity, hence ignoring the spatial correlations among devices. The proof of Lemma 1 is straightforward from [38], [39] and is summarised in Appendix A for completeness.

B. Connectivity-reliability

It is also interesting to investigate the reliability of the connectivity model. For the connectivity process to be reliable, the portion of the connected devices must not be less than ϕ, with $0 < \phi \ll 1$. Connectivity-reliability captures the portion of devices in each realization of Ψ_L that achieves a SIR of θ_ς with a probability of at least ϕ. The goal is to find the complementary cumulative distribution function of P^ς_{conn} given as

$$F_{P^\varsigma_{conn}}(\phi) = P^!\{P^\varsigma_{conn}(\theta_\varsigma) > \phi\}, \forall 0 < \phi \ll 1, \theta_\varsigma \in \mathbb{R}^+, \quad (19)$$

where $P^!$ represents the reduced Palm measure of the PPP.

Lemma 2: The Connectivity-reliability of an uplink offloading of tasks can be obtained from the analytical Meta distribution of the SIR following the Gil-Pelaez theorem [40] as

$$F_{P^\varsigma_{conn}}(\phi) = \frac{1}{2} + \frac{1}{\pi} \int_0^\infty \frac{\Im e^{-jw \ln \phi} M_{jw}}{w} dw, \quad (20)$$

where $\Im(z)$ represents the imaginary parts of complex number z and M_{jw} is obtained through the moments of P^ς_{conn}. The b-th moment of P^ς_{conn} presented in (16) can be approximated as

$$M_{b,\varsigma}(\theta_\varsigma) \triangleq E^0(P^\varsigma_{conn}(\theta_\varsigma)^b), b \in \mathbb{C}. \quad (21)$$

Note that the random variable $P^\varsigma_{conn}(\theta_\varsigma) \in [0, 1]$, then

$$M_{b,\varsigma}(\theta_\varsigma) = \int_0^1 b\phi^{b-1} F_{P^\varsigma_{conn}}(\phi) d\phi. \quad (22)$$

From [41], we know that $P^\varsigma_{conn}(\theta_\varsigma) \equiv M_1(\theta_\varsigma)$ is the first moment of the connectivity probability (i.e., offloading success probability). The approximate solution using beta distribution for the analysis presented in Lemma 2 is straightforward from [41], [42]. The pdf of any beta distributed random variable X with mean ξ can be obtained from the general analysis of the beta function as

$$f_X(x) = \frac{x^{\frac{\xi(\beta+1)-1}{1-\xi}} (1-x)^{\beta-1}}{B(\beta\xi/(1-\xi), \beta)}, \quad (23)$$

where $B(:,:)$ is the beta function. The parameter β is obtained as

$$\beta = \frac{(1-\xi)(\xi - M_2)}{M_2 - \xi^2}. \quad (24)$$

Through the analysis of the moments with $\xi = M_1$, the Meta distribution can be obtained for each value of ϕ and θ_ς. This provides insights into the connectivity-reliability of the network.

C. Communication latency

The communication model captures the communication efficiency during tasks offloading in the radio access network between any tagged device and its corresponding BS in the selected layer. The efficiency of any communication channel can be measure through the joint uplink and downlink communication latency. While the uplink transmission captures the tasks offloading between an active device and its corresponding BS in the selected layer, downlink transmission represents tasks transmission from the PoE to the users' layer after execution. The communication latency is the mean number of time slots required to transmit a task until a successful reception at the intending receiver (i.e., BS at any selected layer for uplink transmission and the tagged device for the downlink transmission). For uplink transmission, the communication latency is obtained as

$$T_{com,up} \triangleq E^0(\min\{t \in \mathbb{N} : 1_t(m \to \varsigma)\}), \tag{25}$$

where $1_t(m \to \varsigma) = 1$ if $SIR_\varsigma > \theta_\varsigma$ in time slot t and 0 if otherwise. For downlink transmission, the communication latency is obtained as

$$T_{com,dow} \triangleq E^0(\min\{t \in \mathbb{N} : 1_t(\varsigma \to m)\}). \tag{26}$$

Because the size of the processed data after computation is much smaller than the size of the input data, while the transmission power at the higher layer is also much larger than the transmission power of devices, the downlink transmission time is usually ignored. Hence, we similarly assumed that the downlink transmission (achieved via dedicated error-free channels) time is negligible, while the location of the tagged device during downlink transmission is known at the selected layer. As a result, we focus only on the uplink communication latency.

Proposition 1: The communication latency experienced during task offloading between a tagged device and its corresponding BS located at the selected layer is given as

$$T_{com,\varsigma} = E^0_\Psi\Big(\frac{1}{P^\varsigma_{conn}}\Big) \approx M_{-1}. \tag{27}$$

Proof: The expression is obtained from the analysis of P^ς_{conn} following Assumption 1. The analysis for $T_{com,\varsigma}$ is obtained as the -1-st moment of connectivity probability conditioned on Ψ_L and Ψ_ς. This is possible since generated tasks are not queued but are offloaded immediately they are generated.

Note that any tagged device does not know the present queueing condition at the higher layers, thus an appropriate layer is selected for the newly generated task based on the required computation capacity. A task that cannot be computed at the selected layer due to the queueing constraints are forwarded to the next higher layer by the selected layer via a dedicated error-free channel with negligible interference. This means additional time is consumed when tasks are forwarded to another layer due to queue constraints. This will be captured through the quality level parameter introduced in Section V.

D. Mean task throughput

The efficiency of the communication model can also be evaluated through the mean task throughput – a metric that captures the mean number of tasks that a typical device can offload successfully in a time slot. This metric is related to the communication latency – which captures the number of time slots required to successfully offload a task.

Definition 1: Given that the number of tasks generated by a tagged device $x^L_k \in \Psi_L$ within any time slot $[0, t]$ is $N_{x^L_k}(t)$, while the average communication latency of any i-th task is $T^i_{com,avg} = \frac{\sum_{\varsigma=1}^{n_l} T^i_{com,\varsigma}}{n_l}$, where $n_l = 3$ represents the number of layers in the system. The mean task throughput can be defined as

$$T_{avg} = \lim_{t\to\infty} \frac{N_{x^L_k}(t)}{\sum_{i=1}^{N_{x^L_k}(t)} T^i_{com,avg}}. \tag{28}$$

By adopting queueing theory technique, we can represent the communication model as a queueing system, where the task generation rate serves as the input to the system, while the offloading success rate serves as the output. With this, the mean task throughput can be obtained as shown in Proposition 2.

Proposition 2: Given that the tasks generation rate follows Bernoulli process with rate g, while the average offloading success probability follows the iid general distribution of rate $P^{avg}_{com} = \frac{\sum_{\varsigma=1}^{n_l} P^\varsigma_{conn}}{n_l}$, the mean task throughput of any tagged device $x^L_k \in \Psi_L$ is given as

$$T_{avg} = \Big\{\frac{P^{avg}_{conn} - g}{1 - g}\Big\}^+, \tag{29}$$

where $\{.\}^+$ represents $\max\{., 0\}$.

Proof: The proof follows from the analysis of queueing system with geometric inter-arrival and general service times [32]. It is omitted for brevity.

V. Analysis of Computation model

Like the communication model, the computation model captures the computation efficiency in the proposed multi-layer edge computing system. This is evaluated through the computation latency – a metric that captures the total time slots a typical task spends waiting for computation opportunity on the queue and the total time slots used for the actual computation. Recall that in Section III, we classified tasks as class 1 and class 2 tasks, with class 1 tasks possessing non-preemptive priority over class 2 tasks. Hence, class 1 tasks may experience lower computation latency compare to class 2 tasks. Next, we obtain the queue dynamics which was followed by the computation details of class 1 and class 2 tasks.

A. Queueing dynamics and service rate

At the beginning of each time slot t, the system contents can be represented using the tuple $Q(t)_\varsigma = (q^\varsigma_1, q^\varsigma_2)$, where q^ς_1 and q^ς_2 are the respective number of class 1 and class 2 tasks in layer ς including those currently being computed. Clearly, $Q(t)_\varsigma$ forms a Markov chain. The queue process $\{Q(t)_\varsigma\}^\infty_{t=1}$ evolves following

$$Q(t+1)_\varsigma = a_\varsigma(t) + \{Q(t)_\varsigma - \mu_\varsigma(t) - \gamma_\varsigma(t)\}^+, \tag{30}$$

where $a_\varsigma = (a_{1,\varsigma}, a_{2,\varsigma})$ represents the number of arriving tasks during the considered time slot, $\mu_\varsigma = (\mu_{1,\varsigma}, \mu_{2,\varsigma})$

represents the number of successfully computed tasks and $\gamma_\varsigma = (\gamma_{1,\varsigma}, \gamma_{2,\varsigma})$ represents the number of reneged tasks.

We assumed a single queue at each layer with K_ς identical VMs where each VM computes a single task at a time. Once a task joins a queue, it cannot leave except due to the expiration of deadline or end of its successful computation. Note that the availability of each VM during any slot is independent of its availability during the other slot and independent of the availability of other VMs. A computed task leaves the system immediately after its successful computation. Owing to the adopted discrete-time queueing system, the computation of any arriving task during a time slot begins at the beginning of the next slot subject to available computation resources. For simplicity and to reduce the impact of I/O interference on the computation of class 1 tasks, we assumed that once a class 1 task is admitted for computation at beginning of any time slot t, the entire layer is unavailable for class 2 tasks. Without loss of generality, we considered VMs to be numbered from 1 through K_ς such that the task with the lowest index is computed by the available VM with the lowest index and task with the second-lowest index is computed by the next available VM, and so on.

Since a maximum of K_ς VMs may be available at any time slot, the average service rate at each layer depends on the number of available VM at such a slot. Hence, the system task execution degree can be summarized as

$$\Lambda_{l_\varsigma} = \begin{cases} \frac{l_\varsigma}{pK_\varsigma}, & l_\varsigma \leq pK_\varsigma \\ 1, & pK_\varsigma < l_\varsigma \leq m_\varsigma \\ 1 + b\ln\left\{\frac{l_\varsigma}{m_\varsigma}\right\}, & m_\varsigma < l_\varsigma \leq H_\varsigma, \end{cases} \tag{31}$$

where b is a constant value. Let $\mu_{\max}$ be the maximum average service rate possible, the dynamic iid average task execution rate can be obtained as

$$\mu_{l,\varsigma} = \Lambda_{l_\varsigma} pK_\varsigma \mu_\varsigma = \begin{cases} l_\varsigma \mu_\varsigma, & l_\varsigma \leq pK_\varsigma \\ pK_\varsigma \mu_\varsigma, & pK_\varsigma < l_\varsigma \leq m_\varsigma \\ pK_\varsigma \mu_\varsigma \Big(1 + b\ln\left\{\frac{l_\varsigma}{m_\varsigma}\right\}\Big), & m_\varsigma < l_\varsigma \leq H_\varsigma \end{cases} \tag{32}$$

where $b = \frac{(\frac{\mu_{\max}}{pK_\varsigma \mu_\varsigma} - 1)}{\ln(\frac{H_\varsigma}{m_\varsigma})}$. At $l_\varsigma = H_\varsigma$,

$$\mu_{H,\varsigma} = pK_\varsigma \mu_\varsigma \Big(1 + b\ln\{\frac{H_\varsigma}{m_\varsigma}\}\Big) = \mu_{\max}. \tag{33}$$

Recall that a task can be redirected from the edge layer to the fog layer or from the fog layer to the cloud layer via a federated computing process [43] if such a task arrives when the queue capacity $l_\varsigma = H_\varsigma$. Such a federated computing system can be characterized by a quality level $q_{f,\varsigma} (0 < q_{f,\varsigma} \leq 1)$ that captures the task's QoS in terms of the expected computation time. In such a case, the average task execution time can be captured as

$$\frac{1}{\mu_{f,\varsigma}} = \frac{1}{q_{f,\varsigma}\mu_{l,\varsigma}} > \frac{1}{\mu_{l,\varsigma}}. \tag{34}$$

From (32) and (34), we can obtain the overall average task execution rate at any selected layer ς as

$$\mu_{ov,\varsigma} = v_\varsigma \mu_{l,\varsigma} + v_f \mu_{f,\varsigma}, \forall v_\varsigma + v_f = 1, \tag{35}$$

where v_ς and v_f represent weights of acceptance at layer ς from the originating device and layer $\varsigma - 1$ respectively. At the edge layer, $v_f = 0$, hence $\mu_{ov,E} = \mu_{l,\varsigma}$. Since a task once admitted at any typical layer can only depart due to expiration of its deadline or end of its successful computation, the average task departure rate is generally given as

$$\omega_\varsigma = \begin{cases} \mu_{ov,\varsigma}, & l_\varsigma \leq m_\varsigma \\ \mu_{ov,\varsigma} + \gamma_{l_\varsigma}, & m_\varsigma < l_\varsigma \leq H_\varsigma. \end{cases} \tag{36}$$

To ensure necessary and sufficient conditions for stability at each layer, it is important to assume that the demand for task execution at each layer does not overload the task execution mechanism. Hence, the average arrival rate at each layer is considered to be less than the average departure rate, i.e., $\rho_\varsigma = \frac{a_\varsigma}{\omega_\varsigma} < 1$. The average joint departure rate $\omega_\varsigma = \omega_{1,\varsigma} + \omega_{2,\varsigma}$, where $\omega_{1,\varsigma}$ and $\omega_{2,\varsigma}$ represent class 1 and class 2 tasks departure rates respectively.

B. Computation latency for class 1 tasks

At any layer, class 1 tasks are computed at the beginning of any slot as long as such tasks are present in the system. Hence class 1 tasks experience lower computation latency compared to class 2 tasks. To obtain the computation latency, one must first understand the system content at the beginning of any arbitrary slot. Hence, we obtain the system content of class 1 tasks in Proposition 3.

Definition 2: The probability that the system in any selected layer is empty depends on the joint arrival rate and departure rate of tasks in such a layer. This is given as $\pi_{0,\varsigma} = 1 - \rho_\varsigma$.

Proposition 3: The mean value of the system contents of class 1 tasks in any layer ς at the beginning of any typical slot is given as

$$U_1^\varsigma = \frac{2\rho_{1,\varsigma}\rho_{\bar{1},\varsigma} + T_{1,\varsigma}V_{a_{1,\varsigma}} + a_{1,\varsigma}^2 V_{T_{1,\varsigma}} + a_{1,\varsigma}a_{2,\varsigma}V_{T_{2,\varsigma}}}{2\rho_{\bar{1},\varsigma}}, \tag{37}$$

where $\bar{X} = 1 - X$, $T_{n,\varsigma} = \frac{1}{\omega_{n,\varsigma}}$ and $V_{[.]}$ is the variance of $[.]$.

Proof: The proof is straightforward from [44] and is presented in Appendix B for completion. From (37), the expression for the mean computation latency can be obtained for any typical class 1 task.

Proposition 4: The mean computation latency of any tagged class 1 task is given as

$$T_{comp}^{1,\varsigma} = \frac{\rho_{1,\varsigma}\rho_{\bar{1},\varsigma} + T_{1,\varsigma}V_{a_{1,\varsigma}} + a_{1,\varsigma}^2 V_{T_{1,\varsigma}} + a_{1,\varsigma}a_{2,\varsigma}L}{2a_{1,\varsigma}\rho_{\bar{1},\varsigma}}, \tag{38}$$

where $L = (V_{T_{2,\varsigma}} + T_{2,\varsigma}^2 - T_{2,\varsigma})$.

Proof: The proof follows from Proposition 3 and is summarized in Appendix C for completeness.

C. Computation latency for class 2 tasks

Similar to class 1 tasks, the computation of class 2 tasks at any selected layer is computed following FCFS and provided that no class 1 task arrived at the beginning of the considered time slot. Hence the computation latency of class 2 tasks is affected by the arrival and departure rates of class 1 tasks. The system content of class 2 tasks in any selected layer is given

in the next Proposition.

Proposition 5: The mean value of the system contents of class 2 tasks in any layer ς at the beginning of any typical slot is given as

$$U_2^{\varsigma} = \frac{2\rho_{2,\varsigma}\bar{\rho}_{\varsigma}\bar{\rho_{1,\varsigma}} + T_{1,\varsigma}^2 a_{2,\varsigma} V_{a_{1,\varsigma}} + \bar{\rho_{1,\varsigma}}[2T_{1,\varsigma}V_{a_{1,\varsigma}} + T_{2,\varsigma}V_{a_{2,\varsigma}}]}{2\bar{\rho}_{\varsigma}\bar{\rho_{1,\varsigma}}} + \frac{a_{2,\varsigma}(a_{1,\varsigma}V_{a_{1,\varsigma}} + a_{2,\varsigma}V_{a_{2,\varsigma}})}{2\bar{\rho}_{\varsigma}\bar{\rho_{1,\varsigma}}}. \quad (39)$$

Proof: The proof follows from Appendix C. From Proposition 5, the expression for mean computation latency can be obtained for any typical class 2 task.

Proposition 6: The mean computation latency of any tagged class 2 task is given as

$$T_{comp}^{2,\varsigma} = \frac{1}{2}\Big\{T_{2,\varsigma} + \frac{V_{a_{2,\varsigma}}^2 T_{2,\varsigma} + 2T_{1,\varsigma}V_{a_{1,2,\varsigma}}}{a_{2,\varsigma}\bar{\rho}_{\varsigma}}\Big\} + \frac{1}{2\bar{\rho_{1,\varsigma}}} \Big\{-\bar{\rho_{1,\varsigma}}(T_{2,\varsigma}-1) + \frac{a_{2,\varsigma}V_{T_{2,\varsigma}} + V_{a_{1,\varsigma}^2 T_{1,\varsigma}^2} + a_{1,\varsigma}V_{a_{1,\varsigma}}^2}{\bar{\rho}_{\varsigma}}\Big\}, \quad (40)$$

where $V_{a_{1,2,\varsigma}}$ is the covariance of $a_{1,\varsigma}$ and $a_{2,\varsigma}$.

Proof: The proof is similar to the proof of Proposition 3 and is omitted for brevity.

D. Reneging due to expiration of deadlines

We capture the probability that a task expires before the completion of its computation using the reneging technique. Let π_{ς} depicts the probability that at least one VM is working in any selected layer, then the probability that any newly arriving class 1 task at such a layer will renege (i.e., experience deadline expiration before computation opportunity) can be approximated as

$$P_{re,1}^{\varsigma} = \frac{a_{1,\varsigma} - \omega_{1,\varsigma}(\pi_{\varsigma} - \pi_{0,\varsigma})}{a_{1,\varsigma}}. \quad (41)$$

Similarly, the probability that any newly arriving class 2 task at such a layer will renege

$$P_{re,2}^{\varsigma} = \frac{a_{\varsigma} - \omega_{\varsigma}(\pi_{\varsigma} - \pi_{0,\varsigma})}{a_{\varsigma}}. \quad (42)$$

From (41) and (42), we can obtain the analysis for offloading efficiency as $E_{off,\varsigma} = 1 - P_{re,n}^{\varsigma}$.

VI. Numerical Simulations and Results

We now numerically analyze the performance of the multi-layer edge computing system investigated in this paper and validate the analyses using independent Monte Carlo simulations averaged over $50,000$ channel realizations. Except otherwise stated, the following simulation parameters were used similar to [11], [12], [45]: $\alpha = 4$, $\lambda_L = 0.4$, $\lambda_E = \lambda_F = \lambda_{CL} = 0.3$.

The connectivity probability gives the fraction of devices with a reliable connection to BSs located in all the considered layers $\varsigma = \{E, F, Cl\}$. As expected, the number of devices with reliable connections reduces as θ_{ς} increases. Owing to the adopted power control technique, we considered the parameter θ_{ς} to be the same for all layers. This allows us to find the

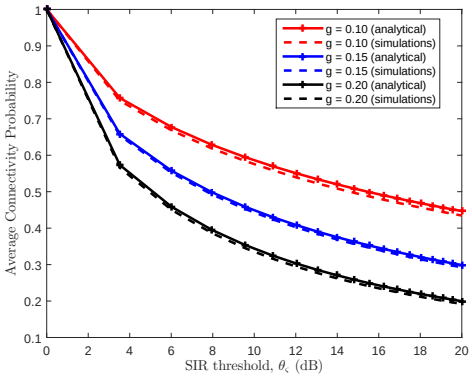

Fig. 3. Average connectivity probability.

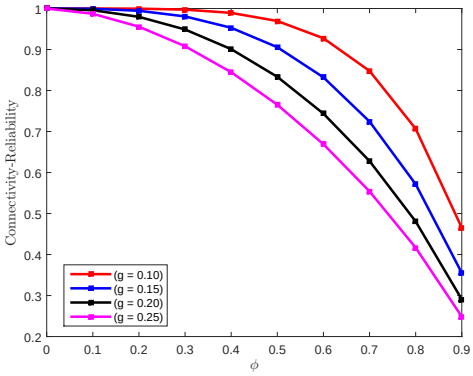

Fig. 4. Connectivity-Reliability achieved, $\theta_{\varsigma} = 9.54$ dB.

average connectivity probability using the fraction of reliable connections to all layers at any observation period as shown in Fig. 3. Similarly, the average connectivity probability was also shown to reduce as the task generation rates increase at the users' level. This is because the number of offloaded tasks is often proportional to the task generation rate, which can increase interference in the system.

It is also desirable to investigate the fraction of connections in each network realization that can achieve θ_{ς} with the reliability of at least ϕ. This was investigated through the Meta distribution – a method that has been properly investigated in the literature. As shown in Fig. 4, the connectivity-reliability reduces as the reliability parameter ϕ increases. This implies that the system performance level decreases when the parameter ϕ is set to a very high value. The connectivity-reliability was also observed to depend on the tasks generation rate and was shown to reduce as task generation rates increases due to the impact of interference.

The communication latency as a function of θ_{ς} is presented in Fig. 5. This metric largely depends on the number of available BSs in any selected layer. We set $\lambda_E = 0.2$, $\lambda_F = 0.3$ and $\lambda_{CL} = 0.5$ to investigate the experienced communication latency when edge, fog and cloud layers are respectively selected. As expected, the communication latency increases with the SIR threshold θ_{ς} regardless of the selected layer. Note that $Pr_E + Pr_F + Pr_{Cl} = 1$, hence $Pr_F + Pr_{Cl} = 0$ when

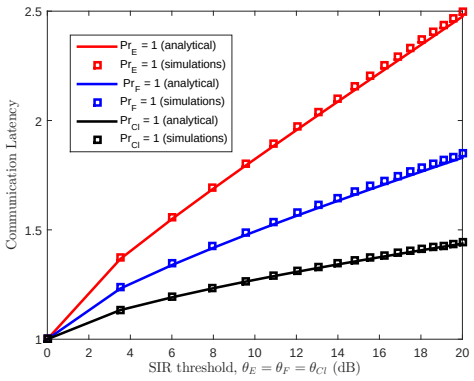

Fig. 5. Average communication latency, $g = 0.1$.

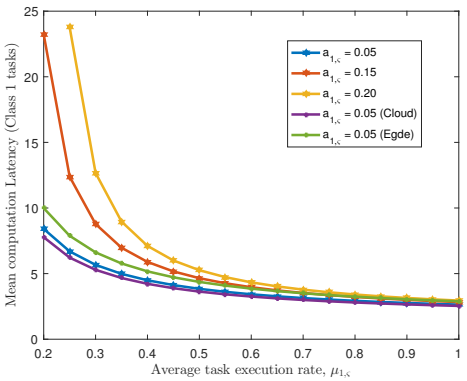

Fig. 7. Computation latency of class 1 task.

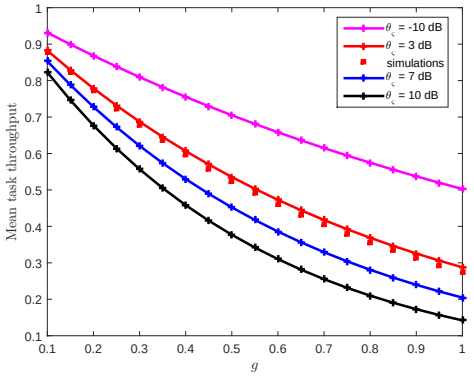

Fig. 6. Obtained mean task throughput vs g, $\lambda_\varsigma = 0.3$.

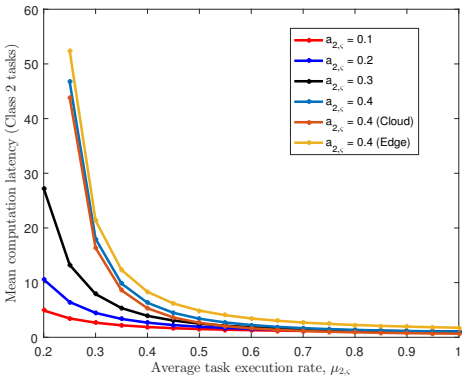

Fig. 8. Effects of average service rate of class 2 tasks on computation latency of class 2 tasks, $a_{1,\varsigma} = 0.1$, $\mu_{1,\varsigma} = 0.3$.

$Pr_E = 1$. A similar condition is applicable for $Pr_F = 1$ and $Pr_{Cl} = 1$. In Fig. 6, the performance of the system in terms of the mean task throughput was investigated. The system mean task throughput was shown to reduce as task generation rates increases. This is also expected as the probability of task re-offloading is expected to increase with g. The metric was similarly shown to depend on θ_ς.

The computation latency is another important metric that is useful in investigating the performance of the system. The mean computation latency of any tagged class 1 task is presented in Fig. 7 under the assumption that the considered system is stable. Owing to the non-preemptive priority enjoys by class 1 tasks, its computation at any layer depends on the class 1 tasks arrival and departure rates, as well as the number of class 2 tasks that were already under computation when such a tagged class 1 task arrived. When the average task execution rate increases (for instance due to an increased number of available VMs), the computation latency reduces. This latency is expected to increase with the arrival rate since more tasks will require computations at the servers. As obtained, the proposed multi-class framework can reduce the computation latency of class 1 tasks (when compared with the traditional single layer approach) without sacrificing offloading efficiency.

Similar to the case of class 1 tasks, the computation latency of class 2 tasks is affected by many factors including the arrival and departure rates of class 1 tasks. Fig. 8 shows the mean computation latency of class 2 tasks as a function of class 2 arrival and mean task execution rates. The mean computation latency of class 2 tasks was shown to increase with the joint arrival rates of tasks in the system. Similarly, Fig. 9 shows the effect of mean task execution rates of class 1 and class 2 tasks on the mean computation latency of class 2 tasks. Class 2 tasks experience improved performance when class 1 and class 2 tasks are executed at relatively high rates, which is possible with multiple VMs deployed at multiple layers.

Generally, the proposed multi-layer edge computing scheme achieved a lower latency and thus higher computation rate when compared with the conventional MEC scheme – a scheme that suffers from limited computing capacity, thereby preventing and delay-sensitive and mission-critical applications to be processed on time. The scheme also achieved a better performance when compare with the traditional cloud computing-based scheme from communication perspective.

Finally, to further demonstrate the ability of the proposed multi-layer edge computing scheme to meet the various tasks' QoS requirements, we evaluate the performance of the multi-layer scheme using task acceptance rate, task execution rate, and average task reneging rate following the analyses presented in the previous sections. As presented in Fig. 10, the multi-layer scheme can achieve a reliable acceptance rate at

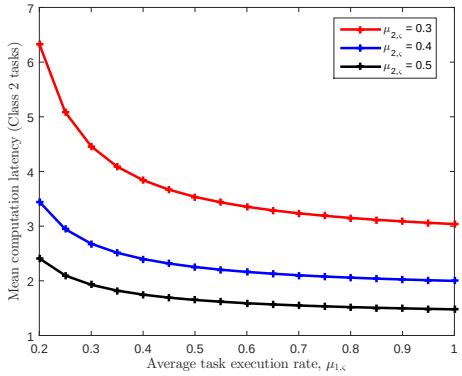

Fig. 9. Effects of average service rate of class 1 tasks on computation latency of class 2 tasks, $a_{1,\varsigma} = 0.1$, $a_{2,\varsigma} = 0.2$.

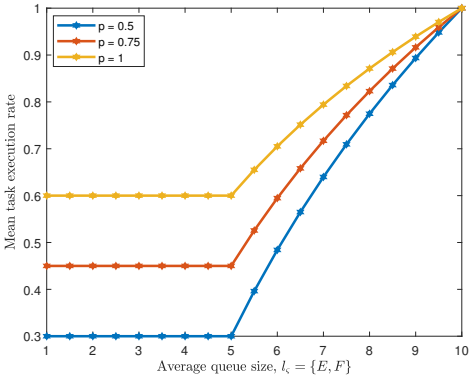

Fig. 11. Task execution probability under stability condition.

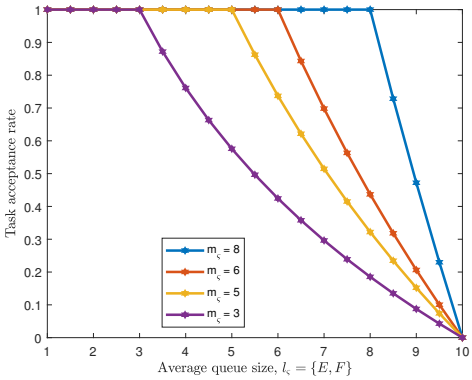

Fig. 10. Task acceptance probability under stability condition.

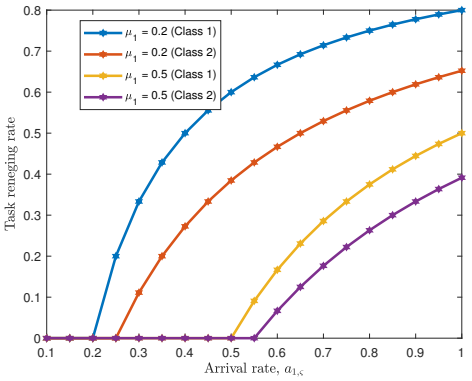

Fig. 12. Impact of class 1 tasks arrival on task completion rate.

$H_\varsigma = 10$. When the queue length $l_\varsigma \leq m_\varsigma$, the acceptance rate is maximal. As the queue length increase beyond the m_ς, the task acceptance rate begins to decrease until the acceptance rate becomes 0 for $l_\varsigma \leq H_\varsigma$. Given a constant task execution rate, the average task acceptance rate increases with m_ς.

Furthermore, the mean task execution rate depends on the queue size and is observed to remain constant for $l_\varsigma \leq m_\varsigma$ at $H_\varsigma = 10$, $\mu_\varsigma = 0.6$ and $K_\varsigma = 1$. As the queue size increases beyond m_ς, the task execution rate increases as shown in Fig. 11. The execution is also observed to decrease with p (i.e., the availability rate of VMs). The task execution rate can be further increased as the number of VMs increases. This will ensure more tasks are computed simultaneously.

To investigate the impact of high arrivals of class 1 tasks on class 2 tasks, we investigated the tasks reneging rate of class 1 and class 2 tasks as $a_{1,\varsigma}$ increases in Fig. 12. Under stable condition, (i.e., $a_{1,\varsigma} < \mu_{1,\varsigma}$), with $a_{2,\varsigma} = 0.15$ and $\mu_{2,\varsigma} = 0.2$ the task reneging rate for both classes of tasks remain zero. This implies that all tasks will be served before the expiration of their deadlines. As the $a_{1,\varsigma} > \mu_{1,\varsigma}$, the system becomes unstable. Thus, the tasks reneging rate increases drastically for both classes of tasks. Interestingly, the unstable behaviour is rare in practical systems as stability condition is often assume.

VII. Conclusion

The need to improve task processing and storage as well as caching capabilities of task offloading systems continue to be important requirements towards a successful deployment of the next-generation system. This will ensure that both delay-sensitive and mission-critical applications are efficiently supported. To meet these requirements, we proposed a multi-layer edge computing system and obtained various analyses to investigate its performances. We adopted the tools of SG, queueing theory and parallel computing.

While the capacity of available VMs in each layer was only captured in the analyses using the probability that such VMs are available in such a layer, this work can be improved by incorporating the computation capacities available at each layer through a more involved queueing dynamic. This will ensure a better understanding of the computation performance at each layer. Notwithstanding this, we believe that the analyses presented in this paper offer important information to understand the importance of the multi-class and multi-layer offloading approach towards improving users' experience in any offloading system.

APPENDIX A
PROOF OF LEMMA 1

The connectivity probability at any selected layer can be obtained as

$$P_{conn}^{\varsigma} = P(SIR_{\varsigma} > \theta_{\varsigma}). \tag{43}$$

Based on the adopted power control scheme by various devices and the nearest BS association, it has been established in [38], [39] that the intra-cell interference from any interfering device is ρ, while the inter-cell interference from any interfering device is less than ρ. Following Assumption 2, (43) gives

$$P_{conn}^{\varsigma} = \mathcal{L}_{I_{out}}\Big(\frac{\theta_{\varsigma}}{\rho}\Big)\mathcal{L}_{I_{in}}\Big(\frac{\theta_{\varsigma}}{\rho}\Big). \tag{44}$$

We know that the inter-cell interference in (44) can be obtained as

$$I_{out} = \sum_{x_i^L \in \Psi_L \setminus x_k^L} 1_{P_i||z_i||^{-\alpha} < \rho} P_i h_i ||z_i||^{-\alpha}. \tag{45}$$

At $\alpha = 4$,

$$\mathcal{L}_{I_{out}}(s) = \exp\Big\{ -2\pi g \lambda_L s^{\frac{1}{2}} E[P^{\frac{1}{2}}] \int_{(s\rho)^{-\frac{1}{4}}}^{\infty} \frac{y}{y^4+1} dy \Big\}, \tag{46}$$

From [38],

$$E[P^{\Theta}] = \frac{\rho^{\Theta}\Gamma\Big\{2\Theta+1, \pi\lambda_{\varsigma}(\frac{P_M}{\rho})^{\frac{1}{2}}\Big\}}{(\pi\lambda_{\varsigma})^{2\Theta}\Big\{1-\exp(-\pi\lambda_{\varsigma}(\frac{P_M}{\rho})^{\frac{1}{2}})\Big\}},$$

where $\Gamma(a,b) = \int_0^b t^{a-1}\exp(-t)dt$ is the lower incomplete Gamma function, while Θ is a positive real number. For the intra-cell interference in (44), we also have

$$I_{in|i}(s) = \sum_i \rho h_i.$$

$$\mathcal{L}_{I_{in}}(s) = E[\exp(-sI_{in})] = \frac{1}{(1+s\rho)^i}. \tag{47}$$

By substituting (46) and (47) into (44), the expression presented in Lemma 1 is obtained.

APPENDIX B
PROOF OF PROPOSITION 3

At the beginning of any slot t in any selected layer, the system can be idle, executing class 1 tasks or executing class 2 tasks. The probability generating function (PGF) of the general system content is given as

$$\begin{aligned} U_{\varsigma}\{z_1(t), z_2(t)\} &= E[Z_1^{u_1(t)} Z_2^{u_2(t)}\{\text{no task}\}] + E[Z_1^{u_1(t)} \\ &Z_2^{u_2(t)}\{\text{executing class 1 tasks}\}] + E[Z_1^{u_1(t)} Z_2^{u_2(t)} \\ &\{\text{executing class 2 tasks}\}] \overset{(a)}{=} U_{\varsigma}(z_1, z_2) \\ &\overset{(b)}{=} U_{\varsigma}(0,0) + (1 - U_{\varsigma}(0,0))\Big\{\frac{\rho_{1,\varsigma}}{\rho_{\varsigma}} E[Z_1^{u_1(t)} Z_2^{u_2(t)} | q_1^{\varsigma} > 0] + \\ &\frac{\rho_{2,\varsigma}}{\rho_{\varsigma}} E[Z_1^{u_1(t)} Z_2^{u_2(t)} | q_1^{\varsigma} = 0, q_2^{\varsigma} > 0]\Big\}. \end{aligned} \tag{48}$$

(a) is obtained through the steady-state equation of $U_{\varsigma}\{z_1(t), z_2(t)\}$ given as $\lim_{t\to\infty} U_{\varsigma}\{z_1(t), z_2(t)\}$, while (b) is obtained knowing that class 1 tasks will be served in slot t if at least one class 1 task arrived at the beginning of slot t. $U_{\varsigma}(0,0)$ captures the case where no task is present (see Definition 2) and is given as

$$U_{\varsigma}(0,0) = Pr[q_1^{\varsigma} = q_2^{\varsigma} = 0] = 1 - \rho_{\varsigma},$$

where $\rho_{\varsigma} = \rho_{1,\varsigma} + \rho_{2,\varsigma}$. Through some algebraic manipulations, the expression in Proposition 3 is obtained by taking the first derivatives of the obtained PGF for $z = 1$.

APPENDIX C
PROOF OF PROPOSITION 4

Note that computation latency is the total slots spent in the system for computation which is the number of slots between the end of the slot in which any tagged task arrived and the end of the slot during which such a task departs from the system due to service completion. A class 1 task will be computed before class 2 tasks following the FCFS scheduling policy. The computation latency of any tagged class 1 task arriving during slot t can be obtained as

$$\begin{aligned} T_{comp}^{1,\varsigma} = \max(T_{\varsigma} - T_{\varsigma}^{el} - 1, 0) + \sum_{m=1}^{U_{1,\varsigma}^{(l)}-1} T_{1,\varsigma}(m) + \sum_{i=1}^{T_{\varsigma}^{el}} \sum_{m=1}^{a_{1,\varsigma}^{(t-i)}} \\ T_{1,\varsigma}^{(t-i)}(m) + \sum_{m=1}^{a_{1,\varsigma}^{*(t)}} T_{1,\varsigma}(m) + T_{1,\varsigma}, \end{aligned} \tag{49}$$

where T_{ς} and T_{ς}^{el} are the execution time and elapsed execution time of the tasks in service during the arriving slot of the tagged task respectively, $U_{1,\varsigma}^{(l)}$ is the system content at the start slot l, and $a_{1,\varsigma}^{*(t)}$ is the number of class 1 tasks that arrive during slot t but are computed before the tagged tasks. From this, the analysis in Proposition 4 is straightforward. Interested readers are referred to [44] for more details.

REFERENCES

[1] M. Bennis, M. Debbah, and H. V. Poor, "Ultrareliable and low-latency wireless communication: Tail, risk, and scale," Proceedings of the IEEE, vol. 106, no. 10, pp. 1834–1853, Sept. 2018.

[2] M. Huang, W. Liu, T. Wang, A. Liu, and S. Zhang, "A cloud–MEC collaborative task offloading scheme with service orchestration," IEEE Internet of Things Journal, vol. 7, no. 7, pp. 5792–5805, Nov. 2019.

[3] M. Kaneko, I. Randrianantenaina, H. Dahrouj, H. ElSawy, and M. S. Alouini, "On the opportunities and challenges of NOMA-based fog radio access networks: An Overview," IEEE Access, Nov. 2020,doi: 10.1109/ACCESS.2020.3037183.

[4] R. Fantacci, and B. Picano, "Performance Analysis of a Delay Constrained Data Offloading Scheme in an Integrated Cloud-Fog-Edge Computing System," IEEE Transactions on Vehicular Technology, vol. 69, no. 10, pp. 12004–12014, July 2020.

[5] S. W. Ko, K. Han, and K. Huang, "Wireless networks for mobile edge computing: Spatial modeling and latency analysis," IEEE Transactions on Wireless Communications, vol. 17, no. 8, pp. 5225–5240, June 2018.

[6] H. Ko, J. Lee, and S. Pack, "Spatial and temporal computation offloading decision algorithm in edge cloud-enabled heterogeneous networks," IEEE Access, vol. 6, pp. 18920–18932, Mar. 2018.

[7] Z. Ning, P. Dong, X. Kong, and F. Xia, "A cooperative partial computation offloading scheme for mobile edge computing enabled Internet of Things," IEEE Internet of Things Journal, vol. 6, no. 3, pp. 4804–4814, Sept. 2018.

[8] X. Chen, L. Jiao, W. Li, and X. Fu, "Efficient multi-user computation offloading for mobile-edge cloud computing," IEEE/ACM Transactions on Networking, vol. 24, no. 5, pp. 2795–2808, Oct. 2015.

[9] G. Hu, Y. Jia, and Z. Chen, "Multi-user computation offloading with D2D for mobile edge computing," in IEEE Global Communications Conference, Abu Dhabi, Dec. 2018, pp. 1–6.
[10] S. Sardellitti, G. Scutari, and S. Barbarossa, "Joint optimization of radio and computational resources for multicell mobile-edge computing," IEEE Transactions on Signal and Information Processing over Networks, vol. 1, no. 2, pp. 89–103, Jun. 2015.
[11] S. D. Okegbile, B. T. Maharaj, and A. S. Alfa, "Interference characterization in underlay cognitive networks with intra-network and inter-network dependence," IEEE Transactions on Mobile Computing, May 2020, doi: 10.1109/TMC.2020.2993408.
[12] S. D. Okegbile, B. T. Maharaj, and A. S. Alfa, "Stochastic geometry approach towards interference management and control in cognitive radio network: A survey," Computer Communications, vol. 166, pp. 174–195, Dec. 2020.
[13] J. Liu, Y. Mao, J. Zhang, and K. B. Letaief, "Delay-optimal computation task scheduling for mobile-edge computing systems," in IEEE International Symposium on Information Theory, Barcelona, Jul. 2016, pp. 1451–1455.
[14] H. S. Lee, and J. W. Lee, "Task offloading in heterogeneous mobile cloud computing: Modeling, analysis, and cloudlet deployment," IEEE Access, vol.6, 14908–14925, Mar. 2018.
[15] B. P. Rimal, D. P. Van, and M. Maier, "Mobile-edge computing vs. centralized cloud computing in fiber-wireless access networks," in IEEE Conference on Computer Communications Workshops, San Francisco, Apr. 2016, pp. 991–996.
[16] I. A. Balapuwaduge, F. Y., Li, and V. Pla, "Dynamic spectrum reservation for CR networks in the presence of channel failures: Channel allocation and reliability analysis," IEEE Transactions on Wireless Communications, vol. 17, no. 2, pp. 882–898, Nov. 2017.
[17] M. Emara, H. El Sawy, M. C. Filippou, and G. Bauch, "Spatiotemporal dependable task execution services in MEC-enabled wireless systems," IEEE Wireless Communications Letters, vol. 10, no. 2, pp. 211–215, Sept. 2020.
[18] V. Q. Rodriguez, and F. Guillemin, "Cloud-RAN modeling based on parallel processing," IEEE Journal on Selected Areas in Communications, vol. 36, no. 3, pp. 457–468, Mar. 2018.
[19] Y. Deng, Z. Chen, X. Yao, S. Hassan, and A. M. Ibrahim, "Parallel offloading in green and sustainable mobile edge computing for delay-constrained IoT system," IEEE Transactions on Vehicular Technology, vol. 68, no. 12, pp. 12202–12214, Oct. 2019.
[20] T. G. Rodrigues, K. Suto, H. Nishiyama, and N. Kato, "Hybrid method for minimizing service delay in edge cloud computing through VM migration and transmission power control," IEEE Transactions on Computers, vol. 66, no. 5, pp. 810–819, May 2016.
[21] T. G. Rodrigues, K. Suto, H. Nishiyama, N. Kato, and K. Temma, "Cloudlets activation scheme for scalable mobile edge computing with transmission power control and virtual machine migration," IEEE Transactions on Computers, vol. 67, no. 9, pp. 1287–1300, Sep. 2018.
[22] Z. Xu, Y. Zhang, H. Li, W. Yang, and Q. Qi, "Dynamic resource provisioning for cyber-physical systems in cloud-fog-edge computing," Journal of Cloud Computing, vol. 9, no. 1, pp. 1–16, Dec. 2020.
[23] S. Guo, D. Wu, H. Zhang, and D. Yuan, "Queueing network model and average delay analysis for mobile edge computing," in IEEE International Conference on Computing, Networking and Communications, Maui, Mar. 2018, pp. 172–176.
[24] Y. Zhang, B. Di, P. Wang, J. Lin, and L. Song, "HetMEC: Heterogeneous multi-layer mobile edge computing in the 6G era," IEEE Transactions on Vehicular Technology, vol. 69, no. 4, pp. 4388–4400, Apr. 2020.
[25] P. Wang, Z. Zheng, B. Di, and L. Song, "HetMEC: Latency-optimal task assignment and resource allocation for heterogeneous multi-layer mobile edge computing," IEEE Transactions on Wireless Communications, vol. 18, no. 10, pp. 4942–4956.
[26] Z. L. Chang, C. Y. Wang, and H. Y. Wei, "Flat-rate Pricing and Truthful Offloading Mechanism in Multi-Layer Edge Computing," IEEE Transactions on Wireless Communications, Apr. 2021, doi: 10.1109/TWC.2021.3071722.
[27] H. Feng, S. Guo, L. Yang, and Y. Yang,"Collaborative Data Caching and Computation Offloading for Multi-Service Mobile Edge Computing," IEEE Transactions on Vehicular Technology, July 2021, doi: 10.1109/TVT.2021.3099303.
[28] C. Sun, X. Wu, X. Li, Q. Fan, J. Wen, and V. C. Leung, "Cooperative Computation Offloading for Multi-Access Edge Computing in 6G Mobile Networks via Soft Actor Critic," IEEE Transactions on Network Science and Engineering, Apr. 2021, doi: 10.1109/TNSE.2021.3076795.
[29] C. H. Lee, and M. Haenggi, "Interference and outage in Poisson cognitive networks," IEEE Transactions on Wireless Communications, vol. 11, no. 4, pp. 1392–1401, Feb. 2012.
[30] D. Moltchanov, "Distance distributions in random networks," Ad Hoc Networks, vol. 10, no. 6, pp. 1146–1166, Aug. 2012.
[31] V. Goswami, "A Discrete-Time Queue with Balking, Reneging, and Working Vacations," International Journal of Stochastic Analysis, vol. 2014, pp. 1–8, Jan. 2014.
[32] S. D. Okegbile, B. T. Maharaj, and A. S. Alfa, "Spatiotemporal characterization of Users' experience in massive cognitive radio networks," IEEE Access, vol. 8, pp. 57114–57125, Mar. 2020.
[33] Q. Wang, and B. Zhang, "Analysis of a busy period queuing system with balking, reneging and motivating," Applied Mathematical Modelling, vol. 64, pp. 480–488, Dec. 2018.
[34] S. D. Okegbile, B. T. Maharaj, and A. S. Alfa, "Outage and throughput analysis of cognitive users in underlay cognitive radio networks with handover," IEEE Access, vol. 8, pp. 208045–208057, Nov. 2020.
[35] R. Arshad, H. ElSawy, S. Sorour, T. Y. Al-Naffouri, and M.-S. Alouini, "Handover management in dense cellular networks: A stochastic geometryapproach," in Proc. IEEE Int. Conf. Commun. (ICC), Kuala Lumpur, Malaysia, May 2016, pp. 1–7.
[36] R. Arshad, H. ElSawy, S. Sorour, T. Y. Al-Naffouri, and M.-S. Alouini, "Cooperative handover management in dense cellular networks," in Proc. IEEE Global Commun. Conf. (GLOBECOM), Washington, DC, USA, Dec. 2016, pp. 1–6.
[37] R. Arshad, H. ElSawy, S. Sorour, T. Y. Al-Naffouri, and M.-S. Alouini, "Velocity-aware handover management in two-tier cellular networks," IEEE Trans. Wireless Commun., vol. 16, no. 3, pp. 1851–1867, Mar. 2017.
[38] H. ElSawy, and E. Hossain, "On stochastic geometry modeling of cellular uplink transmission with truncated channel inversion power control," IEEE Transactions on Wireless Communications, vol. 13, no. 8, pp. 4454–4469, Apr. 2014.
[39] M. Gharbieh, H. ElSawy, A. Bader, and M.Alouini, "Spatiotemporal Stochastic Modeling of IoT Enabled Cellular Networks: Scalability and Stability Analysis," IEEE Transactions on Communications, vol. 65, no. 8, pp. 4454–4469, Aug. 2017.
[40] J. Gil-Pelaez, "Note on the inversion theorem," Biometrika, vol. 38, nos. 3–4, pp. 481–482, Jul. 1951.
[41] Y. Zhong, M. Haenggi, T. Q. Quek, and W. Zhang, "On the stability of static Poisson networks under random access," IEEE Transactions on Communications, vol. 64, no. 7, pp. 2985–2998, June 2016.
[42] X. Tang, X. Xu, and M. Haenggi, "Meta distribution of the SIR in moving networks," IEEE Transactions on Communications, vol. 68, no. 6, pp. 3614–3626, Feb. 2020.
[43] D. Bruneo, "A Stochastic Model to Investigate Data Center Performance and QoS in IaaS Cloud Computing Systems," IEEE Transactions on Parallel and Distributed Systems, vol. 25, no. 3, pp. 560–569, Mar. 2014.
[44] J. Walraevens, D. Fiems, and H. Bruneel, "Performance analysis of priority queueing systems in discrete time," in Network Performance Engineering, Berlin, Germany: Springer, 2011, pp. 203–232.
[45] S. D. Okegbile, and B. T. Maharaj, "Age of Information and Success Probability Analysis in Hybrid Spectrum Access-Based Massive Cognitive Radio Networks," Applied Sciences, vol. 11, no. 4, pp. 1940, Jan. 2021.

S.D. Okegbile received B.Tech (Hons.) degree in Computer Engineering (First class division) from Ladoke Akintola University of Technology, Ogbomoso, Nigeria, in 2011 and M.Sc. degree in Computer Science (Distinction) from Obafemi Awolowo University, Ile-Ife, Nigeria, in 2016. He also received Ph.D. degree in Computer Engineering from the University of Pretoria, South Africa in 2021. His research interests are in the area of pervasive and mobile computing which includes various interesting topics on cognitive radio networks, internet of things and wireless sensor networks.

B. T. Maharaj received the Ph.D. degree in wireless communications from the University of Pretoria, South Africa. He is currently a Professor and holds the position of Sentech Chair of Broadband Wireless Multimedia Communications with the Department of Electrical, Electronic, and Computer Engineering, University of Pretoria. He has over 30 years; industry, consulting, research, and academic experience, and his research interests are in cognitive radio networks, MIMO systems, sensor wireless networks resource allocation, and rural broadband communications systems. Dr. Maharaj has over 140 peer-reviewed journal articles and full conference papers in proceedings and 2 international patents. He has been the conference Chair for IEEE Wireless Africa 2019 conference and the General Conference Chair for the World Engineering Education (WEEF) and Global Engineering Deans Council (GEDC) 2020 Conference.

Attahiru S. Alfa is Professor Emeritus at the University of Manitoba, Department of Electrical and Computer Engineering and Extraordinary Professor at the University of Pretoria, Department of Electrical, Electronic and Computer Engineering. Dr. Alfa's most recent research focus covers age of information, wireless sensor networks, cognitive radio networks, network restoration tools for wireless sensor networks, and the role of 5G on IoT, with specific interest in the mathematical modeling of those systems. His general research covers, but not limited to, the following areas: queueing theory and applications, optimization, performance analysis and resource allocation in telecommunication systems, modeling of communication networks, analysis of cognitive radio networks, modeling and analysis of wireless sensor networks, and smart cities. Some of his previous works include developing efficient decoding algorithms for LDPC codes, channel modeling, traffic estimation for the Internet, and cross layer analysis. Dr. Alfa also works in the application of queueing theory to other areas such as transportation systems, manufacturing systems and healthcare systems. He has authored two books, "Queueing Theory for Telecommunications: Discrete Time Modelling of a Single Node System", published by Springer in 2010, and "Applied Discrete-Time Queue" published in 2015, also by Springer, as a second edition of the first book; and also co-authored (with Dr. Ibrahim) a book titled: "Optimization Methods for User Admissions and Radio Resource Allocation for Multicasting over High Altitude Platforms" published by River Publishers in 2019.