Deadline-Constrained RSU-to-Vehicle Task Offloading Scheme for Vehicular Fog Networks

Maurice Khabbaz

***Abstract*—In Vehicular Fog Computing (VFC), the RSU-to-Vehicle (R2V) task offloading process is highly affected by undesirable yet sometimes inevitable events (e.g., buffer exhaustion, task HoL blocking and deadline expiry), the occurrence of which will notably alter the RSU's performance in terms of crucial Quality-of-Service (QoS) metrics such as the average system response time, the blocking and deadline expiry probabilities. This article proposes a novel R2V Deadline-Constrained Task Offload (R2V-DCTO) scheduling scheme with the objective of improving the RSU's performance in terms of the above-mentioned metrics; hence, filling an important literature gap. For this purpose, a stochastic modelling framework is established to capture the RSU's functional dynamics and assess its performance as it operates under R2V-DCTO. Extensive simulations are conducted to confirm the model's validity and accuracy and then compare R2V-DCTO's performance to that of the often adopted FIFO scheme. Results indicate that, on average, R2V-DCTO outperforms FIFO by 33.4% in terms of the probability of deadline mismatch and by 83.3% in terms of mean system response time; those being critical QoS performance metrics.**

***Index Terms*—Task offloading, mobile edge computing, fog, deadline-constrained, modelling, performance analysis.**

I. Introduction

A. Preliminaries

The Internet of Things (IoT) outlines a connectivity archetype promoting a next-level blend of all-time-anywhere intercommunication of users and things through any available network in support to a wide variety of services. Indeed, with the advancement of wireless technology and IoT's exponential proliferation, urban life is witnessing continuous revolutionary upgrades catalyzed by the foundation of custom-built services intended to improve the living standards of humans. At this level, one cannot overlook the sacred marriage between IoT and smart cities aiming at accentuating metropolitan instrumentation, connectivity and intelligence. Yet, a brutal opposition to the evolution of smart cities lies in the protuberance of IoT-generated massive volumes of data, part of which is deadline-constrained and requires real-time processing. At first thought, cloud mega-centers appear reasonably capable of working around this challenge. However, owing to the excessive delays emanating from data transmission over backhaul links, processing tasks in the clouds contributes excessive delays and, hence, is ill-suited for erupting outbursts of applications characterized by hard deadlines. Luckily though, Mobile Edge Computing (MEC) has been renowned as a momentous and efficient alternative aiming at provisioning on-the-fly data processing and analysis services at networks' edges.

Manuscript received 3 December 2022; revised 2 April 2023; accepted 9 June 2023. Date of publication 13 June 2023; date of current version 14 November 2023. The review of this article was coordinated by Prof. Heejo Lee.

The author is with the Computer Science Department, American University of Beirut, Beirut 11-0236, Lebanon (e-mail: maurice.khabbaz@aub.edu.lb).

Digital Object Identifier 10.1109/TVT.2023.3285255

B. Motivation

A narrower MEC sector of particular interest herein is Vehicular Fog Computing (VFC), which enables the opportunistic exploitation of idle computing resources of mobile/stationary vehicles as well as edge-server-augmented stationary RoadSide Units (RSUs) for task processing purposes. Indeed, today's vehicles are being equipped with ultra-modern server-grade computerized modules and wireless communication devices rendering them capable of exchanging, computing and analyzing data in real-time. RSUs, in turn, may be also privileged with backhaul connectivity allowing them to tap, on-demand (e.g., under overload conditions), into the computing resources of hierarchically organized tiers of cloudlets to palm off some computationally-heavy tasks; thus, reducing their load.

In the above context, task offload scheduling and resource allocation has gained significant attention over the recent past as numerous bidirectional (i.e., Vehicle-to-RSU (V2R) and/or RSU-to-Vehicle (R2V)) task offloading problem variants have been addressed. However, a thorough literature survey reveals that the majority of existing work in this regard aimed at: *a*) minimizing IoT devices' energy consumption [1], *b*) optimizing resource allocation and utilization [2], *c*) minimizing latency [3], *d*) contract-/incentive-based resource sharing [4], *e*) optimal scheduling of task batches subject to latency, resource availability and task priority constraints [5], [6], *f*) accounting for service vehicle availability and vehicular mobility limitations [7], *g*) exploitation of computational resource pooling [8], [9], among other publications revolving around derivatives of these problem variants resolved either through the development of complex optimization formulations eased by meta-heuristics or using the overly trending model free Deep Reinforcement Learning (DRL) frameworks. Task denial-of-admission, reneging (i.e., premature task drop due to deadline expiry), Head-of-Line (HoL) blocking and deadline-based task prioritization, are all equally important aspects as their above-enumerated counterparts but have been totally overlooked in the literature especially in the context of R2V task offloading scenarios. Filling this gap is, hence, this work's primary objective.

C. Novel Contributions

The distinguishing features of this present article and its novel contributions are:

1) The development of a R2V Deadline-Constrained Task Offloading (R2V-DCTO) scheme that prioritizes incoming tasks according to their deadlines (i.e., serve tasks with the shortest deadlines first). R2V-DCTO has the objective of jointly reducing the average System Response Time (SRT) and the task deadline mismatch probability.
2) A Work-Conserving Earliest-Deadline-First (WC-EDF) stochastic queueing model is developed to represent an RSU that equipped with a finite task buffer operating under R2V-DCTO. This model also captures the system dynamics and theoretically quantifies important Quality-of-Service (QoS) metrics such as the task blocking probability (due task buffer exhaustion), the probability of task deadline mismatch, the probability of matched service completion, and the mean SRT.
3) Establishing an extensive numerical analysis and simulation framework to: *i*) assert the correctness and validity of the proposed model and *ii*) evaluate the RSU's performance under R2V-DCTO and gauge its merit compared to other existing schemes used for benchmarking.

II. Scenario Description and System Model

Along the road network of a smart city, are deployed RSUs that have been empowered with a multitude of new roles that go beyond typically serving as stationary access points connecting in-range users and vehicles to backhaul networks and/or the Internet as well as provisioning them with specific services these RSUs host.

Factually, today's RSUs constitute integral active infrastructure components. They support/catalyze the emergence/proliferation of new revolutionary edge computing paradigms. These latter are expected to handle massive amounts of continuously generated data and requested computationally exhaustive tasks by IoT devices including but not limited to those pertaining to: *a*) commuting vehicles (e.g. Augmented-reality-enhanced autonomous driving through sensing surroundings and creating virtual environments [10], lane condition scanning [11], etc), *b*) in-range surveillance UAVs (e.g., image dehazing, object detection, 3D rendering, multi-sensor data fusion, pattern recognition, video processing, feature extraction and many others, [12], [13], [14], [15], [16], [17], [18]), *c*) pedestrians (e.g., AI-related tasks, object recognition, road status, [5]), among others. Truly, this brings upfront significant challenges for both bandwidth and computational resources especially that, often, data requires to be immediately communicated and various tasks need to be processed in real-time. Owing to: *i*) the impracticality of offloading tasks to higher-tier cloudlets, which is induced by considerable bandwidth consumption and lengthy response delays, as well as, *ii*) under high offered loads, the over-occupancy of edge server computing resources extirpating these latter reduced-delay virtue, VFC presents itself as an efficient alternative offering deadline-constrained and computationally intensive applications the luxury of opportunistically exploiting

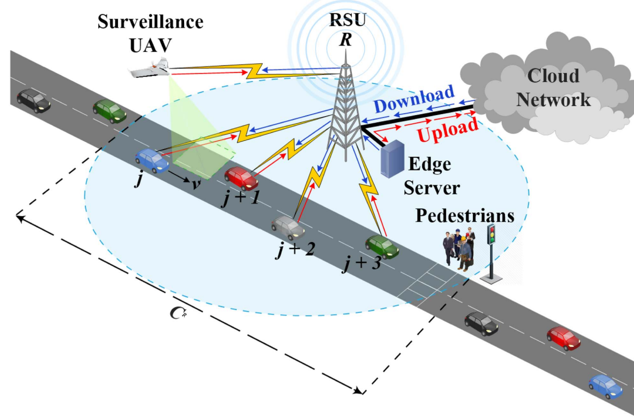

Fig. 1. R2V task offloading scenario.

the idle computational resources of vehicles navigating through the communication ranges of deployed RSUs.

In the above-described context, which is also illustrated in Fig. 1, the RSU, R, will follow the rules dictated by the R2V-DCTO scheme proposed hereafter and play the critical role of task offloading coordinator that will receive newly incoming deadline-constrained tasks from different users/devices, collectively store them in its finite buffer and adequately schedule and dispatch/assign them to in-range vehicles (e.g., vehicles j through $j+3$) that, in turn, may possess enough computational capacity suitable for processing these tasks within their respective deadlines.

Tasks are prioritized based on the ascending order of their deadlines; meaning that tasks with the shorter deadlines are queued ahead of those with larger deadlines and service is offered for tasks with the shortest deadlines first. In addition, for simplicity, it is assumed that all vehicles are cooperative; hence, vehicles will announce their presence and put their available resources at the RSU's disposal. It follows that, given a tagged task, $\mathcal{T}$, occupying the RSU's buffer's front position, the RSU shall be responsible for determining and possibly releasing $\mathcal{T}$ to an available in-range vehicle that is deemed suitable for processing it. Upon releasing $\mathcal{T}$, the RSU will then move to serving the next task in line. However, observe here that two critical cases may arise, namely: *i*) there exist no vehicles within the RSU's range, and, *ii*) given the presence of in-range vehicles, none of them is suitable for processing $\mathcal{T}$ (i.e., none of the vehicles is capable of completing $\mathcal{T}$ before it departs from the RSU's range). Under such circumstances, $\mathcal{T}$ will be forcefully further retained at the buffer's HoL position until a suitable vehicle arrives; hence, obstructing the way out for all other subsequently queueing tasks. Under medium-to-high task arrival rates, tasks will start accumulating within the RSU's buffer and experience larger response times (i.e., queueing delays). It will actually not take too long before task deadline expiries start to occur. Also, eventually, the RSU's buffer may get completely exhausted, an event upon the occurrence of which, any newly incoming task will be denied admission and immediately discarded.

Bottom line, induced by the RSU's clearance restriction of an HoL task such as $\mathcal{T}$ is an euphoric turn of possible events, the occurrence of which being highly correlated to various offloading feasibility influential factors consisting of: *i*) the mobility of vehicles and, hence, their opportunistic residency within the RSU's range, *ii*) vehicular heterogeneity (i.e. the per-vehicle amount of available on-board computing resources), and, *iii*) task heterogeneity (i.e. task type and size in terms of the number of instructions). The tangible quantification of such factors is outside this article's scope and is left for future work. However, the RSU's HoL task clearance/dispatch inability is definitely accounted for herein through the introduction of an original parameter γ defined as the probability of task release. In other words, an HoL task is released with a probability of γ. Otherwise, with a probability of $1-\gamma$ the task is blocked at the buffer's front position.

The remaining of this article shall be dedicated for theoretically characterizing the above elaborated system dynamics governed by R2V-DCTO and quantifying important QoS metrics (i.e., as enumerated in Section I.)

III. Modelling and Performance Analysis of an RSU's Queue Operating Under R2V-DCTO

The presented model hereafter was rooted in [20]. Its elemental assumptions are first presented below.

A. Elemental Assumptions

1) Time is subdivided into slots of normalized duration $\tau = 1$ (*s*). Incoming tasks are assumed to arrive at the end of a slot.
2) Each task's service time is constant and equal to one slot.
3) An arriving task has an initial deadline $T_\Delta = t$. Initial task deadlines are i.i.d. integer r.v.s distributed over a range $[0; T_\Delta^{\max}]$ with a cumulative distribution function (c.d.f.) $F_{T_\Delta}(\cdot)$ such that $F_{T_\Delta}(0) = 0$ and $F_{T_\Delta}(t)\big|_{t \geq T_\Delta^{\max}} = 1$. A task's residual deadline is a decreasing function of time. At the beginning of a time slot, a task is *licit* if it has a strictly positive residual deadline. To derive the probability of task deadline mismatch, once $T_\Delta = 0$, it keeps holding this value.
4) The RSU's queue is finite with queueing positions conveniently numbered 1 through $Q-1$.
5) Task arrivals to the RSU follow a Poisson process with parameter λ_T. Relative to a queued task, say $\mathcal{T}$, having, at the start of a slot, a residual deadline of t, any incoming tasks within that slot having strictly smaller initial deadlines than t shall be inserted in front of $\mathcal{T}$. Clearly, during time slot l, the arrivals of α_{t_l} tasks having shorter initial deadlines than t follow a Poisson process with parameter $\lambda_t = F_{T_\Delta}(t-1)\lambda_T$ with $\lambda_1 = 0$. Consequently, tasks with unity residual deadlines are guaranteed to hold their respective positions.
6) An incoming task's initial position is known. Once queued, a task either moves forward until it gets served, or, it gets pushed backwards by incoming higher-priority tasks until it gets dropped out of the queue.
7) The number, $N_S^{(l)}$ of tasks served during an arbitrary slot l has a geometric distribution with parameter γ.

B. Modelling of the R2V-DCTO Governed RSU Buffer

Denote by $s_l = (q_l, t_l)$ the state of a target task $\mathcal{T}$ at an arbitrary slot l where $q_l \in [0; Q]$ denotes $\mathcal{T}$'s position (equivalently, the number of tasks in front of $\mathcal{T}$ and will be served before this latter) and $t_l \in [0; T_\Delta^{\max}]$ denotes its residual deadline. Here, note that if $q_l = 0$, then $\mathcal{T}$ is right away admitted into service. In contrast, if $q_l = Q$, this means that $\mathcal{T}$ is promptly dropped. Obviously, regardless of t_l's value, as soon as $\mathcal{T}$ reaches any of the states $(0, t_l)$ or (Q, t_l), is gets stuck there forever. Accordingly, these two states are absorbing states. Keep in mind that a value of $t_l = 0$ points at the expiry of $\mathcal{T}$'s deadline. The set $\{s_l\}_{l \geq 0}$ composed of a finite number of states (i.e., the state space Λ is finite) constitutes a Markov Chain (MC), the evolution of which is governed by:

$$q_{l+1} = \mathcal{G}\left(0, q_l - N_S^l + \alpha_{t_l}, Q\right) \tag{1}$$

$$t_{l+1} = \max\left(0, t_k - 1\right) \tag{2}$$

where $\mathcal{G}(\beta, \omega, \phi) = \beta$ if $\phi \leq \beta$, $\mathcal{G}(\beta, \omega, \phi) = \omega$ if $\beta < \omega < \phi$, and, $\mathcal{G}(\beta, \omega, \phi) = \phi$ if $\omega \geq \phi$. The r.v.s α_{t_l} and $N_S^{(l)}$ were defined in assumptions 5 and 7 of Section IV-A. Also note that, given a random variable (r.v.), $M = m$, representing the number of suitable vehicles residing within the RSU's range, the RSU may attempt to offload at most m tasks at the start of each time slot where $m \in [0; M_{\max}]$ with $M_{\max}$ being the maximum number of vehicles that may be concurrently found within the RSU's range. Note that M has a probability mass function (p.m.f.) denoted by $f_M(m)$ that has been derived in [19]. Consequently, $N_S^{(l)} \in [0; \min(q_l, m)]$.

At this point, it becomes clear that the earlier-mentioned MC comprises a total of $(T_\Delta^{\max}+1)(Q+1)$ states each being identified by an integer index from the range $\mathcal{I} = [0; (T_\Delta^{\max}+1)(Q+1)-1]$.

Now, without loss of generality (w.l.o.g.) assume that $\mathcal{T}$'s state at the start of slot l is $x \in \mathcal{I}$. $\mathcal{T}$'s next state, $y \in \mathcal{I}$, may be stochastically characterized based on: *a*) the current state x, and, *b*) the respective distributions of α_{t_l} and $N_S^{(l)}$. Denote by $\rho_{xy} = \Pr[s_{l+1} = y \big| s_l = x]$ the MC's transition probabilities. Given the fact that the distributions of α_{t_l} and $N_S^{(l)}$ are independent of l, the probabilities $\{\rho_{xy}\}$ are also independent of l. As such, the MC, $S = \{s_l\}_{l \in \mathbb{N}}$, is both absorbing and homogeneous. S's finite state space Λ comprises a subset of absorbing states Θ that, in turn, are sub-dividable into two subsets as follow: *a*) $\Phi \triangleq$ the subset of states where $\mathcal{T}$ shall receive service, and, *b*) $\Delta \triangleq$ the subset of states where $\mathcal{T}$ shall be dropped. Denote by $\Gamma = \Lambda - \Theta$ the subset of the remaining transient states. To this end, (Φ, Δ, Γ) forms a partition of Λ. Temporal homogeneity promulgates that S's states may be featured by (q, t) where $q \in [0; Q]$ and $t \in [0; T_\Delta^{\max}]$.

C. Performance Metrics Derivation

At the heart of the derivation of all of the earlier-mentioned QoS metrics in Section I lies ρ_{xy} with $(x, y) \in \mathcal{I}^2$. Being the one-step transition probabilities, ρ_{xy}, indicate the possibility of S to transition in one stop from state x (i.e., (q_x, t_x)) to state y (i.e. (q_y, t_y)). It is easy to show that these transition probabilities can be expressed as in (3) shown at the bottom of this page. The terms ρ_{xy} can, therefore, be used to build the transition matrix $\mathbf{T}$ of S. Let $\mathcal{C}_\Lambda = \mathrm{Card}(\Lambda)$ denote the cardinality of Λ (the state space). $\mathbf{T}$ is, thus, a $(\mathcal{C}_\Lambda \times \mathcal{C}_\Lambda)$ square matrix. Let $\sigma \triangleq \mathrm{Card}(\Gamma)$ represent the number of transient states and $\kappa \triangleq \mathrm{Card}(\Theta) = \mathrm{Card}(\Phi) + \mathrm{Card}(\Delta)$ represent the number of absorbing states. It can be easily shown that, canonically, $\mathbf{T} = [\mathbf{A}, \mathbf{B}; \mathbf{0}, \mathbf{I}]$ where $\mathbf{A}$ is a $(\sigma \times \sigma)$ matrix of one-step transition probabilities among non-absorbing states, $\mathbf{B}$ is a $(\sigma \times \kappa)$ matrix of one-step transition probabilities from transient states to their absorbing counterparts, $\mathbf{0}$ is a $(\kappa \times \sigma)$ matrix of zeros and $\mathbf{I}$ is a $(\kappa \times \kappa)$ identity matrix. With S being an absorbing chain, $\mathbf{I} - \mathbf{A}$ is invertible and $\mathbf{T}_\mathcal{F} = [\mathbf{I} - \mathbf{A}]^{-1}$ is S's $(\sigma \times \sigma)$ fundamental matrix. Denote by t_{xy} an element of $\mathbf{T}_\mathcal{F}$ located at row x and column y. It is easily proven that $t_{xy} = \rho_{xy}^{(0)} + \rho_{xy}^{(1)} + \rho_{xy}^{(2)} + \cdots$, where ρ_{xy}^k is the k-step transition probability from state x to state y with $(x, y) \in [1; \sigma]^2$. To this end, the Probability of Absorption (PoA) may be derived as follows. Let ϵ_x and ζ_x represent the events where task $\mathcal{T}$ (with initial position $x \Rightarrow s_0 = x$) gets finally absorbed by either one of Φ's or Δ's states respectively. It can be shown that:

$$e_x = \Pr[\epsilon_x] = \begin{cases} 0 & , x \in \mathcal{I}_\Delta \\ \sum_{\forall y \in \mathcal{I}_\Phi} \sum_{\forall w \in \mathcal{I}_\Gamma} t_{xw} \cdot \rho_{wy} & , x \in \mathcal{I}_\Gamma \\ 1 & , x \in \mathcal{I}_\Phi \end{cases} \quad (4)$$

$$z_x = \Pr[\zeta_x] \begin{cases} 0 & , x \in \mathcal{I}_\Phi \\ \sum_{\forall y \in \mathcal{I}_\Delta} \sum_{\forall w \in \mathcal{I}_\Gamma} t_{xw} \cdot \rho_{wy} & , x \in \mathcal{I}_\Gamma \\ 1 & , x \in \mathcal{I}_\Delta \end{cases} \quad (5)$$

where $\mathcal{I}_\Delta = [\sigma + \mathrm{Card}(\Phi) + 1; \sigma + \kappa]$, $\mathcal{I}_\Phi = [\sigma + 1; \sigma + \mathrm{Card}(\Phi)]$ and $\mathcal{I}_\Gamma = [1; \sigma]$. Now, using the above-computed PoAs, the QoS metrics of interest are:

1) *Probability of Matched Service Completion (PMSC), $\mathcal{P}_{msc}$:* is the probability that $\mathcal{T}$ was served before expiry (i.e. $t > 0$) computed using (4) $\forall y \in \{\mathcal{I}_\Phi - \{\sigma + 2\}\}$ corresponding to states $(0, t_y)$ where $t_y \in [1; T_\Delta^{\max}]$.
2) *Probability of Deadline Mismatch (PDM), $\mathcal{P}_{dm}$:* is the probability that $\mathcal{T}$ is served while having $t = 0$. It is computed using (4) with $y = \sigma + 1$ corresponding to state $(0, 0)$.
3) *Probability of Blocking (PB), $\mathcal{P}_b$:* is the probability that $\mathcal{T}$ is dropped before it expired (i.e., $t > 0$). It is computed using (5) with $y \in \{\mathcal{I}_\Delta - \{\sigma + \mathrm{Card}(\Phi) + 1\}\}$ corresponding to states (Q, t_y) where $t_y \in [1; T_\Delta^{\max}]$.
4) *Average System Response Time (SRT), $\overline{\mathcal{R}_x}$:* representing the expected number of waiting time slots that $\mathcal{T}$ experiences until absorption by one of Φ's states if $\mathcal{T}$'s initial state is x:

$$\overline{\mathcal{R}_x} = \sum_{y \in \mathcal{I}_\Gamma} t_{xy} \cdot e_x \quad , x \in \mathcal{I}_\Gamma \quad (6)$$

IV. Numerical Analysis and Simulations

A custom-built JAVA-based discrete-event simulator is developed at this stage to: *i*) assert the validity and accuracy of the above-proposed theoretical derivations as well as *ii*) emulate the RSU's buffer's dynamics as it operates under R2V-DCTO and assess its performance in terms of $\mathcal{P}_{\text{msc}}$, $\mathcal{P}_{\text{dm}}$, $\mathcal{P}_{\text{b}}$ and $\overline{\mathcal{R}_x}$. Each of these metrics will be evaluated for a number of 10^6 tasks and averaged out over multiple simulator runs with the objective of realizing at least 95% confidence interval. The simulator's input parameter values vary in the following ranges: *a*) $\gamma \in [0; 0.5]$, *b*) $T_\Delta^{\max} = 6$ (*s*), *c*) $\lambda_t \in [0.1; 1]$ (*tasks/s*), *d*) task service rate, $\mu_t = 0.5$ (*tasks/s*).

A. Model Verification and Discussion of Results

To validate the accuracy of the presented model in Section III, the RSU is assumed to operate under normal conditions (i.e., $\gamma = 0$ implying that the RSU always finds a suitable in-range vehicles to dispatch its queueing tasks). Also note that, for the purpose of driving the generated simulation results closer to reality, some of the model's restrictive assumptions (e.g. constant per-task service interval of 1 time slot) are relaxed throughout the simulations. To this end, Fig. 2(a) and (b) plot concurrently distinct pairs of simulated curves together with their theoretical counterparts corresponding to each of the performance metrics listed in Section IV-A. These figures constitute tangible proofs of the validity of the proposed model and its accuracy as the simulation results almost perfectly coincide with their theoretical counterparts. Observe in these figures that, for a fixed task service rate, the number of queued tasks in the RSU's buffer increases as a function of λ_t; hence, leading an increase (here almost quasi linear with a weak slope) in $\mathcal{P}_{\text{dm}}$ as well as a more significant increase in the blocking probability. This is especially true since, following Little's Theorem, the larger the buffer size becomes, the larger the average SRT $\overline{\mathcal{R}_x}$ will become (Fig. 2(b)); hence, inducing a higher likelihood of task deadline expiry as well as a buffer overflow. This will also lead to an equivalent drop in $\mathcal{P}_{\text{msc}}$.

B. Further R2V-DCTO Performance Evaluation

First, the performance of R2V-DCTO will be evaluated in terms of more impactful parameters such as the task release probability (equivalently the resource availability probability), namely, γ. Recall that, a task shall be forcefully retained at

$$\rho_{xy} = \begin{cases} 0 & , \forall (q_x, t_x) \in \Theta \text{ and } \forall (q_y, t_y) \in \Gamma \\ \sum_{m=1}^{M_{\max}} \left[\sum_{n=0}^{\min(q_x, m)} \Pr\left[N_S^{(x)} = n\right] \cdot \Pr\left[\alpha_{t_x} = q_y - (q_x - n)\right] \right] \cdot f_M(m) & , \forall (q_x, t_x) \in \Gamma \text{ and } \forall (q_y, t_y) \in \Lambda \end{cases} \quad (3)$$

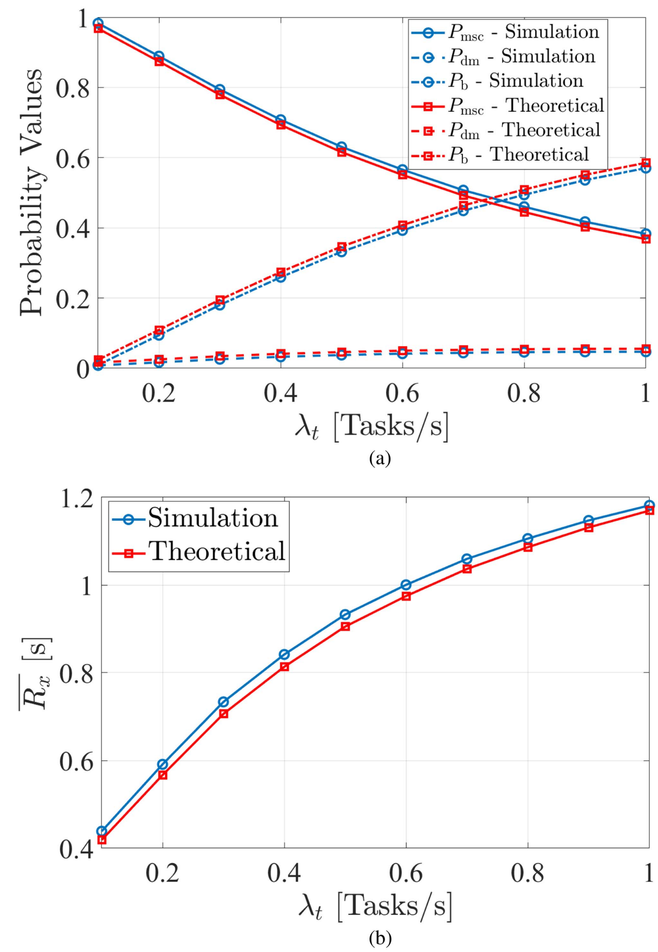

Fig. 2. Model correctness and accuracy verification.

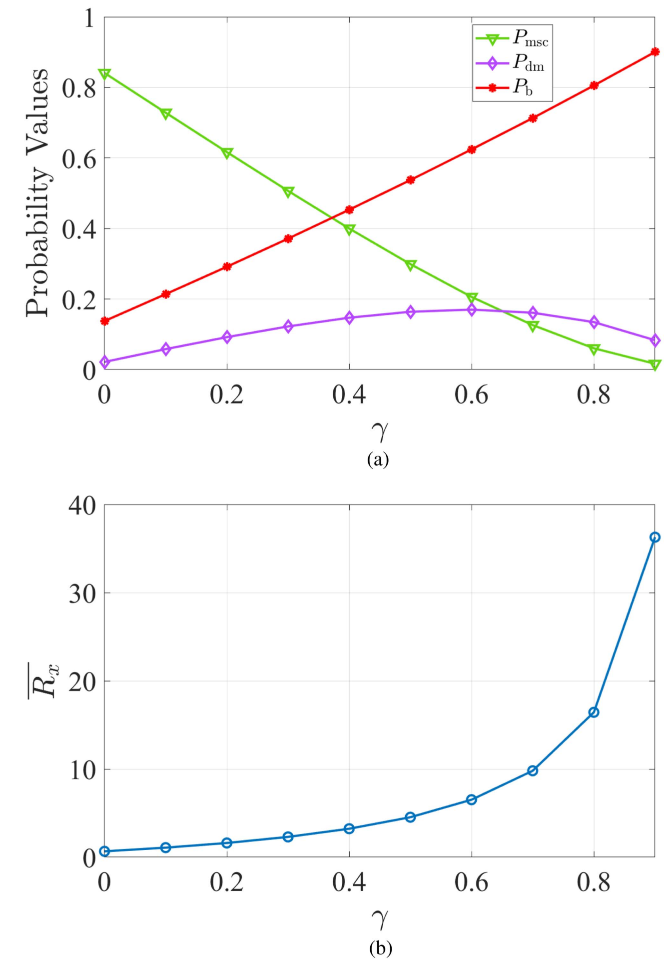

Fig. 3. Effect of γ on RSU's performance under R2V-DCTO.

the HoL position of the RSU's buffer if no suitable in-range vehicles. Regardless of the fact that the theoretical computation of γ is currently outside the scope of this article, the developed simulator herein does account for vehicular mobility as it is fed with realistic mobility traces generated by the well renowned Simulation for Urban MObility (SUMO), [21]. A carefully chosen combination of traffic parameter values as per the guidelines of [19] with per-vehicle available computing capacity values from the adopted range in [5] allows for varying $\gamma \in [0; 0.9]$. Here, to ensure that the RSU's buffer is stable, a value of $\lambda_t = 0.25$ is taken. This will truely contribute to highlighting the unique effect of γ on the system's performance as reported in Fig. 3. Here, obviously, a larger value of γ indicates the difficulty that the RSU experiences in clearing out an HoL task; this latter blocking the way out in front of all other queued tasks behind and, hence, causing the queue length to increase. Under such circumstances, one shall expect that queued tasks shall experience higher SRT, which is indeed the case as shown in Fig. 3(b) and as such, task deadline expiries will become more likely to occur and, more importantly, the RSU's buffer shall rapidly overflow (as shown in the increasing $\mathcal{P}_{\rm dm}$ and $\mathcal{P}_{\rm b}$ as a function of γ). Nonetheless, an important aspect of $\mathcal{P}_{\rm dm}$'s trend is that, beyond a threshold value of γ (here $\gamma = 0.5$), $\mathcal{P}_{\rm dm}$ surprisingly starts to decrease. In fact, this arises because the RSU's buffer overflows whereby it becomes more likely that newly lower-deadline incoming tasks will casue the RSU to forcefully push out tail tasks that have not yet expired. Discarding such tasks though manifests itself as a more pronounced increase in $\mathcal{P}_{\rm b}$ (due to premature blocking) that overshadows $\mathcal{P}_{\rm dm}$'s decrease. The cumulative effect is reflected by an overall damped decrease in $\mathcal{P}_{\rm msc}$.

Next, the performance of R2V-DCTO is compared to that of the First-In-First-Out (FIFO) that is observed to be typically adopted in the literature (e.g., [4], [5], [7], [9]) for arranging tasks arriving to and queueing within the RSU's buffer. For the purpose of gauging the merit of R2V-DCTO over FIFO, simulations are conducted with elimination of the external resource outage effect (i.e., $\gamma = 0$). Fig. 4 concurrently plots the achieved metrics' values by both R2V-DCTO and FIFO as a function of λ_t. While the performance trends pertaining to R2V-DCTO have already been investigated above in terms of all the metrics of interest, those of FIFO, opposite to all expectations, appear to be atypical especially in terms of $\mathcal{P}_{\rm dm}$ and $\mathcal{P}_{\rm b}$. As Fig. 4(a) shows, FIFO's $\mathcal{P}_{\rm dm}$ exhibits a quasi-exponential growth as a function of λ_t only leading to a spike at $\lambda_t = 0.5$ and start slowly decreasing thereafter. In explaining this behavior, a closer observation of the RSU's FIFO-based queue reveals the queueing of a relatively considerable proportion of tasks with large deadlines ahead of

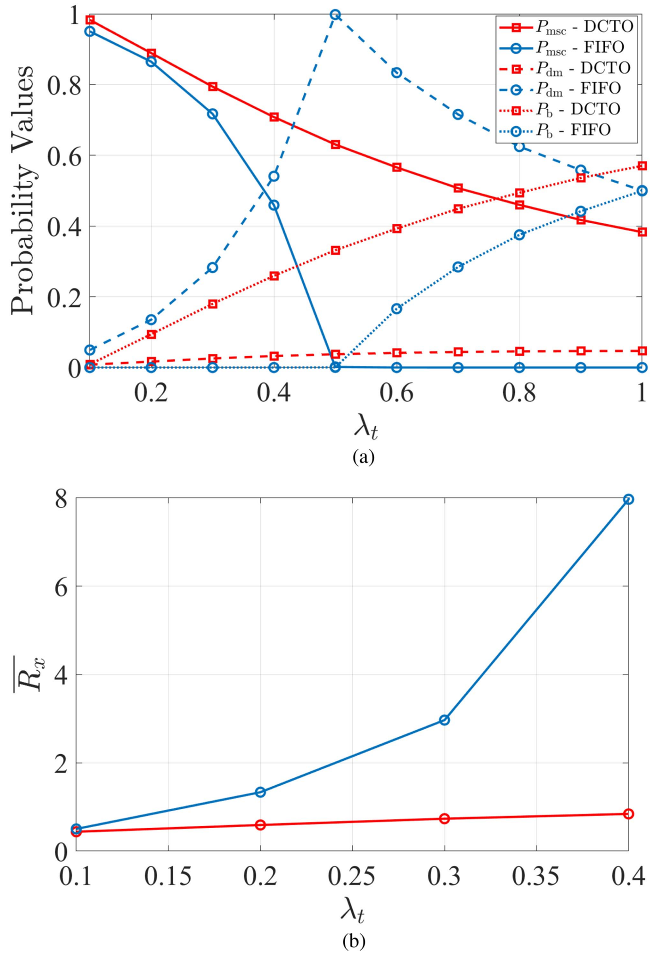

Fig. 4. Performance comparison of R2V-DCTO versus FIFO.

short-lived ones; hence, inducing numerous task deadline expiries. Such expiries truly hinder the buffer's overflow especially that these expiries, up to $\lambda_t = 0.5$, occur at a much higher rate than that of task arrivals. This is why FIFO's $\mathcal{P}_{\rm b} = 0$ as no blocking occurs. Beyond that λ_t takes over causing blocking events to occur, a notable portion of which are premature (i.e. discarding non-expired tail tasks to replace them with newer ones with shorter deadlines). This explains the decay of FIFO's $\mathcal{P}_{\rm dm}$ at the same slope of FIFO's $\mathcal{P}_{\rm b}$'s increase. In fact, beyond $\lambda_t = 0.5$ clearly, extremely little tasks are released by the RSU while the majority end up either blocked or expired obviously going in parallel with excessive response times as indicated in Fig. 4(b) that does not show results beyond $\lambda_t = 0.4$ since, thereafter, FIFO's $\overline{\mathcal{R}_x}$ will overshoot; hence, marginalizing the RSU's utility in task offloading to in-range vehicles as FIFO's $\mathcal{P}_{\rm msc} \approx 0$ beyond that point.

On the overall, R2V-DCTO remarkably outperforms FIFO by an average of 33.4% in terms of $\mathcal{P}_{\rm msc}$ and 83.3% in terms of $\overline{\mathcal{R}_x}$; those being the essential performance indicators.

V. Conclusion

This article proposes an RSU-to-Vehicle Deadline-Constrained Task Offloading (R2V-DCTO) scheme to exploit computational resources of in-range vehicles in processing exhaustive yet deadline and latency sensitive tasks in the context of a Vehicular Fog Computing scenario. R2V-DCTO prioritizes short-deadlined tasks over ones with larger deadline intervals. A theoretical work conserving EDF-based queueing model is established herein to capture the RSU's buffer dynamics as it operates under R2V-DCTO and assess its performance in terms of essential Quality-of-Service (QoS) metrics such as the deadline mismatch, blocking and matched service completion probabilities as well as the average system response time. Extensive numerical analyses and simulations contributed to verifying the correctness and accuracy of the formulated model and gauge R2V-DCTO's merit against the typically adopted FIFO scheme. Results indicate clearly that R2V-DCTO remarkably outperforms FIFO in terms of the above-listed crucial QoS metrics.

References

[1] H. Wu, Y. Sun, and K. Wolter, "Energy-efficient decision making for mobile cloud offloading," *IEEE Trans. Cloud Comput.*, vol. 8, no. 2, pp. 570–584, Apr.–Jun. 2020.

[2] S.-S. Lee and S. Lee, "Resource allocation for vehicular fog computing using reinforcement learning combined with heuristic information," *IEEE Internet Things J.*, vol. 7, no. 10, pp. 10450–10464, Oct. 2020.

[3] Q. Wu, H. Liu, R. Wang, P. Fan, Q. Fan, and Z. Li, "Delay-sensitive task offloading in the 802.11p-based vehicular fog computing systems," *IEEE Internet Things J.*, vol. 7, no. 1, pp. 773–785, Jan. 2020.

[4] Z. Zhou, P. Liu, J. Feng, Y. Zhang, S. Mumtaz, and J. Rodriguez, "Computation resource allocation and task assignment optimization in vehicular fog computing: A contract-matching approach," *IEEE Trans. Veh. Technol.*, vol. 68, no. 4, pp. 3113–3125, Apr. 2019.

[5] I. Sorkhoh, D. Ebrahimi, C. Assi, S. Sharafeddine, and M. Khabbaz, "An infrastructure-assisted workload scheduling for computational resources exploitation in the fog-enabled vehicular network," *IEEE Internet Things J.*, vol. 7, no. 6, pp. 5021–5032, Jun. 2020.

[6] J. Shi, J. Du, J. Wang, J. Wang, and J. Yuan, "Priority-aware task offloading in vehicular fog computing based on deep reinforcement learning," *IEEE Trans. Veh. Technol.*, vol. 69, no. 12, pp. 16067–16081, Dec. 2020.

[7] Q. Wu, H. Ge, H. Liu, Q. Fan, Z. Li, and Z. Wang, "A task offloading scheme in vehicular fog and cloud computing system," *IEEE Access*, Vol. 8, pp. 1173–1184, 2020.

[8] K. Sharma, B. Butler, and B. Jennings, "Graph-based heuristic solution for placing distributed video processing applications on moving vehicle clusters," *IEEE Trans. Netw. Serv. Manage.*, vol. 19, no. 3, pp. 3076–3089, Sep. 2022.

[9] C. Tang, S. Xia, Q. Li, W. Chen, and W. Fang, "Resource Pooling in Vehicular Fog Computing," *J. Cloud Comput.*, vol. 10, 2021, Art. no. 19.

[10] Y. Xiao and Chao Zhu, "Vehicular fog computing: Vision and challenges," in *Proc. IEEE Int. Conf. Pervasive Comput. Commun. Workshops*, 2017, pp. 6–9.

[11] C. Yu, B. Lin, P. Guo, W. Zhang, S. Li, and R. He, "Deployment and dimensioning of fog computing-based Internet of Vehicle infrastructure for autonomous driving," *IEEE Internet Things J.*, vol. 6, no. 1, pp. 149–160, Feb. 2019.

[12] A. Mehra, M. Mandal, P. Narang, and V. Chamola, "ReViewNet: A fast and resource optimized network for enabling safe autonomous driving in hazy weather conditions," *IEEE Trans. Intell. Transp. Syst.*, vol. 22, no. 7, pp. 4256–4266, Jul. 2021.

[13] Q. Liu, H. Liang, R. Luo, and Q. Liu, "Energy-efficiency computation offloading strategy in UAV aided V2X network with integrated sensing and communication," *IEEE Open J. Commun. Soc.*, vol. 3, pp. 1337–1346, 2022.

[14] M. Jiang, Y. Li, Q. Zhang, and J. Qin, "Joint position and time allocation optimization of UAV enabled time allocation optimization networks," *IEEE Trans. Commun.*, vol. 67, no. 5, pp. 3806–3816, May 2019.

[15] Z. Yang, C. Pan, K. Wang, and M. Shikh-Bahaei, "Energy efficient resource allocation in UAV-enabled mobile edge computing networks," *IEEE Trans. Wireless Commun.*, vol. 18, no. 9, pp. 4576–4589, Sep. 2019.

[16] X. Cao, J. Xu, and R. Zhang, "Mobile edge computing for cellular-connected UAV: Computation offloading and trajectory optimization," in *Proc. IEEE 19th Int. Workshop Signal Process. Adv. Wireless Commun.*, 2018, pp. 1–5.

[17] Q. Hu, Y. Cai, G. Yu, Z. Qin, M. Zhao, and G. Y. Li, "Joint Offloading and Trajectory Design for UAV-enabled Mobile Edge Computing Systems," *IEEE Internet Things J.*, vol. 6, no. 2, pp. 1879–1892, Apr. 2019.

[18] A. Alioua, H.-E. Djeghri, M. E. T. Cherif, S.-M. Senouci, and H. Sedjelmaci, "UAVs for traffic monitoring: A sequential game-based computation offloading/sharing approach," *Comput. Netw.*, Vol. 177, 2020, Art. no. 107273.

[19] R. F. Atallah, M. J. Khabbaz, and C. M. Assi, "Modeling and performance analysis of medium access control schemes for drive-thru internet access provisioning systems," *IEEE Trans. Intell. Transp. Syst.*, vol. 16, no. 6, pp. 3238–3248, Dec. 2015.

[20] R. A. E. Bardan, "EDF-based connection setup management in WDM optical networks," M.Sc.E. Thesis, Lebanese American Univ., Beirut, Lebanon, 2009.

[21] P. A. Lopez et al., "Microscopic traffic simulation using SUMO," in *Proc. IEEE 21st Int. Conf. Int. Transp. Syst.*, 2018, pp. 2575–2582.