Cite As:
R. Yadav, W. Zhang, O. Kaiwartya, H. Song and S. Yu, "Energy-Latency Tradeoff for Dynamic Computation Offloading in Vehicular Fog Computing," in IEEE Transactions on Vehicular Technology, vol. 69, no. 12, pp. 14198-14211, Dec. 2020

Energy-Latency Tradeoff for Dynamic Computation Offloading in Vehicular Fog Computing

Rahul Yadav, *Member, IEEE*, Weizhe Zhang, *Senior Member, IEEE*, Omprakash Kaiwartya, *Senior Member, IEEE*, Houbing Song, *Fellow, IEEE*, and Shui Yu, *Senior Member, IEEE*

***Abstract*—Vehicular Fog Computing (VFC) provides solutions to relieves overload cloudlet nodes, reduces service latency during peak times, and saves energy for battery-powered cloudlet nodes by offloading user tasks to a vehicle (vehicular node) by exploiting the under-utilized computation resources of nearby vehicular node. However, the wide deployment of VFC still confronts several critical challenges: lack of energy-latency tradeoff and efficient resource allocation mechanisms. In this paper, we address the challenges and provide an Energy-efficient dynamic Computation Offloading and resources allocation Scheme (*ECOS*) to minimize energy consumption and service latency. We first formulate the *ECOS* problem as a joint energy and latency cost minimization problem while satisfying vehicular node mobility and end-to-end latency deadline constraints. We then propose an *ECOS* scheme with three phases. In the first phase, we propose an overload cloudlet node detection policy based on resource utilization. In the second phase, we propose a computational offloading selection policy to select a task from an overloaded cloudlet node for offloading, which minimizes offloading cost and the risk of overload. Next, we propose a heuristic approach to solve the resource allocation problem between the vehicular node and selected user tasks for energy-latency tradeoff. Extensive simulations have been conducted under realistic highway and synthetic scenarios to examine the ECOS scheme's performance. In comparison, our proposed scheme outperforms the existing schemes in terms of energy-saving, service latency, and joint energy-latency cost.**

***Index Terms*—Vehicular fog computing, computation offloading, energy-efficiency, green computing, efficient-latency, and vehicle mobility**

I. Introduction

With the expansion of Internet-of-Things (IoT) and communication technologies, there arises a critical issue that both the computational demand and data rate grows exponentially [1]. For example, emerging 5G applications such as infotainment application, interactive gaming, fleet tracking, blockchain, and natural language processing. Such complex applications require advance computation, data communication, storage, and energy consumption techniques to handle the complicated storage and data processing operations [2]. This poses a new challenge to the conventional cloud computing paradigm. It is difficult to guarantee the stringent quality of experience (QoE) and quality of service (QoS) requirements due to the long distance between remote data centers and user equipments (UEs) [3], [4]. To resolve this issue, a new paradigm has been proposed called fog computing, which extends cloud-based utilities for UEs at edge networks [5]. Zhou *et al.* [6] proposed a contract-learning approach for computational offloading and resource sharing in a fog computing environment to optimally shared the computation and communication resources, which reduced service latency. However, to cover a vast geographic area, deployment of a large number of servers required, which increases maintenance cost and energy consumption. Furthermore, considering the dynamically time-varying demands, these servers will waste vast amounts of resources during the off-peak time. Therefore, how to employ a server to process ever-increasing demand in communication and computation with moderate costs via a demand-adaptation approach remains an open problem. The term demand-adaptation can be described as the growing demand of the computation resources is fulfill by using exiting under-utilized computation resources without deploying any new servers or Cloudlet Nodes (CNs). The small-size data centers (such as gateway routers, switches, set-top boxes, roadside units (RUs), etc.) are known as CNs.

To solve this problem, an alternative choice is to leverage the under-utilized computation resources of nearby vehicles (Vehicular Node (VN)). Smart vehicles with on-board dedicated short-range communication, high-speed computer, and Long Term Evolution (LTE) communication devices are known as VN. It is predicted that more than 380 million connected vehicles will be on the road by 2021 [7], and these vehicles owners earn extra money by providing their under-utilized vehicle resources to UE (similar to Uber taxi service). Hence, a large group of nearby vehicles can provide an enormous amount of computational resources during peak time without deploying any additional computing nodes. Moreover, the computation demand of UEs can be offloaded to VN, whenever the CN detected overloaded to minimize energy consumption and service latency. This new computing paradigm is an integration of vehicular computing and fog computing, known as vehicular fog computing (VFC) [8]. However, despite the advantages of VFC mentioned above, the deployment of large-scale VFC still faces several critical challenges, which are summarized as follows.

Firstly, the idea of offloading the UEs task has been explored in some previous studies [9]–[11]. Most of the previous studies have assumed that unconditionally selection of computational

R. Yadav and W. Zhang is with the Harbin Institute of Technology, and with Cyberspace Security Research Center, Peng Cheng Laboratory, Shenzhen 518055, China. (e-mails: rahul@stu.hit.edu.cn, wzzhang@hit.edu.cn).

O. Kaiwartya is with the Nottingham Trent University, U.K. (e-mail: omprakash.kaiwartya@ntu.ac.uk.)

H. Song is with the Embry-Riddle Aeronautical University, USA. (e-mail: songh4@erau.edu)

S. Yu is with the University of Technology Sydney and with Peng Cheng Laboratory, Shenzhen 518055, China. (e-mail: shui.yu@uts.edu.au)

tasks for offloading is too optimistic in practice. The cost incurred by task offloading which depends on the resources required by tasks. For example, the selected UE task required a higher bandwidth for offload to VN, which significantly increase offloading cost. Thus, the question arises, which UEs tasks should be offload to VN. Therefore, it is of vital importance to develop policy, which can effectively select UEs tasks to offload to VN and optimize offloading cost.

Secondly, there lacks a near-optimal resources allocation mechanism. This mechanism efficiently tradeoff energy consumption and service latency of CN. With the existence of different characteristics of tasks (such as CPU-intensive tasks and latency-sensitive tasks), a critical challenge is how to allocate VN computation resources to UEs tasks such that minimize energy consumption and service latency of different characteristics of the tasks without compromising QoS. Since each UE task's characteristic is different, it is highly possible that each UE task needs computing resources and service latency as per their characteristics. Additionally, this mechanism should take into account the computation task requirements and availability of computing resources at different VNs. Therefore, an optimal resource allocation mechanism needs to develop to balance energy consumption and service latency as per task characteristics.

These aforementioned challenges motivate us to develop an ECOS scheme for optimizing the energy consumption and service latency of the CN. The benefits brought by ECOS scheme are: (i) firstly, as a CN can be overloaded during peak times, one can release the burdens on that CN by directing some UEs task to offload to the neighboring VN, thus preventing the limited computational resources on each CN from becoming the bottleneck; (ii) secondly, selection of computational task from overloaded CN to offload to VN with more favorable UE task condition, thus reducing offloading cost; (iii) finally, coordination of resource allocation to offload UEs task across multiple VNs can help in computational resources contention among the UEs task and optimize energy-latency tradeoff when multiple CNs are overloaded and offloaded their tasks simultaneously. The main contributions of this work are summarized as follows:

- We investigate how to optimize energy and latency of overloaded CN during the peak time by exploring computation energy consumption, transmission energy consumption, task transmission latency, and task computation latency. An in-depth theoretical analysis of the functions describing the inherent mobility of vehicles is described to optimizing the offloading failures. Next, we formulate the joint optimal energy and latency task offloading problem to minimize the Energy and Latency Efficient Cost (ELEC) under the constraints of dynamically fluctuating limited resources at the VN, under the constraint of VN mobility, and hard end-to-end latency deadline constraint.
- To provide a tractable solution, we propose a three-phase ECOS scheme. In the first phase, evaluate CPU and memory demands of all UEs tasks that are executing on the CN and calculate aggregate CPU and memory demands for detecting an overload CN. In the second phase, the computation offloading selection policy introduced for selecting an optimal UEs task from overloaded CN to offload. This policy aims to determine which task of UEs is offloaded onto a VN to reduces offloading cost. In the third phase, a heuristic approach based joint energy and latency efficient resource allocation algorithm introduced, which search an efficient VN for processing offloading tasks. The benefits brought by the proposed scheme are: reduces the risk of overload, minimize the offloading failures, maximize the energy saving, minimize service latency, and reduces the time complexity.
- We present simulation results to illustrate the performance of the proposed ECOS scheme by using optimal parameter configuration found for resource allocation, efficient energy consumption, latency, and VN mobility.

The rest of the paper is organized as follows: Section II provides a related literature survey on latency and energy efficiency management in vehicular fog computing. The VFC system model, the detail of the energy & latency-aware algorithm formulation, the proposed model, and problem formulation are described in Section III. The ECOS scheme introduced in Section IV. Section V introduced the simulation setup for the proposed scheme, analysis and compared the simulation results with existing approaches. Finally, the conclusion is provided in Section VI of the paper.

II. Related Work

Since VFC is envisioned as a promising approach to enhance the overall capacity of fog or edge computing, several works have already studied VFC architecture [12], [13]. Lee *et al.* introduced a three-tier vehicular cloud network architecture, which combined the vehicular cloud and information-centric network [12]. Hou *et al.* first introduced an idea of VFC, which utilizes computational and communication resources of parked and moving vehicles. In this work, a quantitative analysis of the fog-enabled network capacity is carried out [13]. Zhu *et al.* introduced a joint optimization of service latency and quality loss for the task assignment problem and solved by exploiting mixed-integer linear programming [14].

Some works have already investigated the design problem in Fog/Edge computing. Fog or Edge provides computational resources to network edges for reducing incoming traffic toward clouds, which in turn reduces the response time of network requests. Su *et al.* also introduced a framework for utilizing the resources of parked vehicles as content caching nodes. Comparing with the traditional method of content-centric networking, which heavily relies on roadside-units (RUs), this proposal imposes less burden on RUs, which provides higher capacity and also allows vehicle-vehicle communication via decentralized communication [15]. Zanni *et al.* proposed a task selection algorithm for mobile edge computing environment. This algorithm can parse an Android application autonomously and classify all the methods based on their offloadability by

adopting a finegrained and multi-steps analyzer [16]. Neto *et al.* proposed a lightweight and efficient framework for mobile computation offloading. It is equipped with a decision engine that minimizes remote execution overhead, while not requiring any modification in the devices operating system [17]. Zhou *et al.* proposed a convex-concave-procedure-based contract optimization algorithm for server recruitment, which aims to maximize the expected utility of the operator with asymmetric information. Then, a low-complexity and stable task offloading mechanism are also proposed to minimize the total network delay based on the pricing-based matching [18]. However, most of the current works improving the performance of task services, and they lack to consider the energy consumption of overloaded CN which affects the performance significantly.

Based on the study of computation offloading strategies with vehicle-vehicle and vehicle-to-infrastructure communication modes, Zhang *et al.* proposed an efficient predictive scheme for offloading tasks to fog nodes via predictive or direct relay transmission mode [19]. Feng *et al.* proposed an ant colony optimization based scheduling algorithm for vehicular tasks [20]. Chen *et al.* proposed a layered network architecture for vehicular networks' computational capabilities [21]. Gerla *et al.* proposed that vehicles can be utilized as mobile cameras by a centralized server to provide photo services, where vehicles serve as sensors for acquiring pictures. The collected data can be uploaded to cloud servers via a cellular network such that users can retrieve information from the cloud [22]. Zhou *et al.* studied the energy-efficient workload offloading problem and proposed a low-complexity distributed solution based on consensus alternating direction method of multipliers. By incorporating a set of local variables for each UE, the original problem, in which the optimization variables of UEs are coupled together, is transformed into an equivalent general consensus problem with separable objectives and constraints [23].

Another critical challenge in VFC is how to allocate resources to the offloaded tasks. Lots of works have addressed the resource allocation problem with different optimization strategies. Otternwalder *et al.* has proposed a policy that reduces network congestion. It is a plan-based operator placement and migration policy for Mobile Complex Event Processing applications. They have also focused on the mobility of the users, which creates a time-graph model to identify possible migration targets. Considering the shortest path from the data source, it selects the appropriate target instance from the time graph model. In the selected instance, the policy applies coordination to accommodate the migrating operator. The main intentions of the proposed policy are to reduce network overhead and end-to-end delay [24]. Souza *et al.* has proposed a service allocation model for Fog-to-Cloud architecture. In this model, the Fog environment is divided into slots, while services are decomposed into atomic services; afterward, such atomic services are allocated to the available slots. Furthermore, each slot is connected to one of the underlying nodes for executing service requests. The model focuses on the allocation of the services at fog nodes to optimize power consumption, processing loads, and services latency [25]. Nevertheless, most of these research works assume that fog nodes are either fixed along a fixed path, while generally ignoring the highly dynamic nature of the vehicular network.

In contrast to existing schemes that suffer from a poor energy-latency tradeoff. In this paper, we concentrate on the computation offloading process in a vehicular fog environment and propose a heuristic approach based offloading policy to improve the energy consumption and latency of overloaded CN while guaranteeing the required latency constraint, resources capacity constraint, and VN mobility constraint.

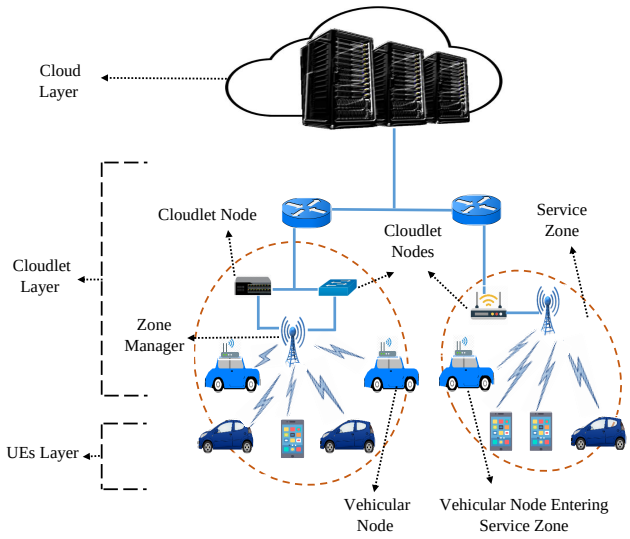


Fig. 1. System Architecture of Vehicular Fog Computing

III. System Model and Problem Formulation

A. System Architecture Overview

The system architecture of VFC is shown in Fig. 1, which organized in a hierarchical order such as the Cloud layer, Cloudlet layer, and UEs layer. In the cloudlet layer, there exist the nodes (such as CNs and VNs) which take charge of computation resource allocation, communication resource coordination, and task placement. During peak time, when the CN is overwhelmed by the incoming computation demands, a set of VNs is employed for sharing computational resources to solve the overload problem via task offloading. The full details of the proposed architecture are described as follows:

1) ***Cloud Layer****:* The cloud data center consists of heterogeneous resources which consider as topmost layer called Cloud Layer. It's considered as a standalone computational platform in this work.

2) ***UEs Layer****:* UEs layer is considered in the lowest layer in this architecture. Any typical UE equipped with sensitive radio transceivers, sensors, collision radars, cameras, on-board computers, and GPS devices. This layer generates a massive amount of raw data but cannot process it due to energy, latency, and resource constraints.

3) ***Cloudlet Layer****:* In this layer, nodes communicate bi-directional with each other to perform data and process management in order to support UE task requirements. CNs are linked top to down in hierarchical order, and resources of these nodes heterogeneous. The nodes closer to the UEs layer have

less Computational, networking, and storage capabilities than the higher or closer to the cloud layer. This layer consists of two types of nodes, which are shown in Fig 1 and describe as follows.

- *Cloudlet Nodes:* The small-size data centers (such as gateway routers, switches, set-top boxes, roadside units (RUs), etc.) are known as CNs. The computational resources of these CNs are heterogeneous for executing UE tasks. Unlike Cloud data centers, CNs are equipped with limited-capacity batteries and limited computational resources due to their inherent physical structure and can be deployed across the edge of the networks [13]. Each CN has a dedicated CN overload prediction module to monitor the utilization of its resources.
- *Vehicular Nodes:* These computing nodes carried by smart vehicles with on-board dedicated short-range communication, high-speed computer, and LTE communication devices. In addition, we assume that computational capacities and other resources of all vehicular nodes are different. When a VN establishes communication with *zone manager* or enters the service zone, *zone manager* responsible for assigning offloaded UE tasks to the VN for processing and can finish this processing before leaving the service zone.

Service Zone: In urban areas, all corners of the cities are divided into the zone and each zone fully connected with cellular networks. Similarly, we divided cities into *service zone*, and it includes CNs as well as VNs. In each *service zone* have a *zone manager*, which coordinates all other nodes. In the proposed architecture, we always select an LTE base station to be the *zone manager* and assume that VNs are deployed on commercial fleets, such as buses and taxis. Whenever VN enters or leaves the service zone, always inform the *zone manager*, similar to cellular registration mechanisms. Moreover, the *zone manager* periodically gathers all information about VN moving directions, locations, and available resources.

B. System Model Overview

In the conventional fog system, computation and communication requests of all UEs have to be served by the CNs, which maximize energy consumption, service latency and make the *service zone* overloaded problem even worse. An alternative solution is offloading the high volumes of UEs tasks from CNs to vehicular nodes by joint optimization of energy and latency under considering mobility factor in VFC environment. We consider a VFC environment composed of l CNs $F^f = (f_1, f_2, ..., \hat{f_k}, ..., f_l)$, and n vehicular nodes $F^v = (v_1, v_2, ..., v_i, ..., v_n)$, geographically distributed and connected with each other. F^f and F^v are the sets of CNs and VNs respectively. There is a set of UE which includes m UEs $U = (u_1, u_2, ..., u_j, ..., u_m)$ and it is associated with a set of task $T = (t_1, t_2, ..., t_j, ..., t_m)$. Throughout, let T be set of all tasks processed by $F^f \cup F^v$.

Remark 1: The proposed VFC model is different from a fog system where the task of UEs is offloaded to micro-servers. Firstly, the computation and communication resources of fog infrastructures are deployed and owned by the same network operator and the location of these micro-servers is fixed. However, in our proposed VFC environment, vehicles are owned by individual users and mobility of VNs and UEs is taken into consideration. Secondly, in both models, the service area divided into *service zones* and in the fog system, UEs within the same *service zone* can connect one to three micro-servers because each *service zone* consist of one to three micro-servers. However, in VFC, due to the high mobility of vehicular nodes in the same *service zone* UE may be surrounded by multiple heterogeneous VNs.

The preference of a VN towards resource allocation is quantified as its energy-latency cost. A VN with lower energy-latency cost is more willing to allocate resources to UEs and compared to a high energy-latency cost VN. Thus, it is intuitive for the *zone manager* to employ lower energy-latency cost VN in a same *service zone*. Since each *service zone* have a finite number of VNs. So, the set of VN for sharing their resources is a discrete and finite space. We describe the energy consumption model in VFC is a combination of computing and communication energy consumption. We are discussing computational energy consumption and communication energy consumption as follows.

Computational Energy Consumption: In VFC, computing energy consumption is determined by the energy consumed by CPU in CNs. The computational energy consumed by nodes is determined by the energy consumes per clock cycle directly propositional to the square of the supply voltage. In the CPU chip of node, the dynamic voltage scaling technology is often used to save the energy consumption, and the supply voltage is assumed approximately proportional to the clock frequency [26]. Computing energy consumption model of the node is described as follows:

$$E^p_{j,k} = \lambda S_{j,k} \varphi (U_{j,k})^2 \tag{1}$$

where $(U_{j,k})$ represents computing capacity of node in cloudlet layer is allocated to UE u_j task as GHz, $S_{j,k}$ represents total size of UE u_j task in bits, λ is computation workload, i.e., the total number of CPU cycles required for each bit of UE u_j task. The computation power of the CN CPU in Joule/Mc is given as $\varphi(U_{j,k})^2$, where $\varphi = 10^{-8}$ [27].

Transmission Energy Consumption:

When a task offloaded to VN required communication link to exchange the information. it includes two phases in sequence: (i) transmitting phase and (ii) receiving phase. In the transmitting phase, UE task t_j input data transmits with the certain specification to the VN v_i through a wireless uplink channel. In the receiving phase, UE u_j receives output data of task t_j from the VN v_i through the wireless downlink channel. The energy consummation of the transmitting task t_j is based on the total transmission time of UE task t_j. We show how to derive the explicit communication consumption model, which is giving by

$$E^t_{i,j} = \eta (P^{base}_{i,j} + p_a P^t_{i,j})(l^t_{i,j} + ß) \tag{2}$$

where $E^t_{i,j}$ represents energy consumes by multiple hops for transmitting UE u_j task to VN v_i. η, p_a and ß represents

number of hops that task t_j traverse hops to reach VN for processing, efficient factor of power amplifier, and a round-trip overhead between a VN v_i and UE u_j respectively. $P_{i,j}^{base}$ and $P_{i,j}^{t}$ represents the static power consummation, transmission power of UEs respectively. The $l_{i,j}^{t}$ represents task transmission latency and the value of $l_{i,j}^{t}$ is estimated using equation 4.

Then, overall energy consumption is defined as summation of both equation (1) and (2), which is given as

$$E_{i,j} = E_{i,j}^{p} + E_{i,j}^{t} \tag{3}$$

*1) **System Latency Model**:* In offloading mode, the *zone manager* offload UEs computation tasks from overloaded CNs to VNs to reduce the total number of transmission hops. The latency in the VFC environment is composed of the task transmission delay and the task computation delay discussed as follows.

Task Transmission Latency: Given the limited bandwidth, the task transmission delay depends on the offloaded task size, which required to be transmitted from UEs to VN with the help of *zone manager*. In this work, we ignored the co-channel interference among UEs by assuming that each UE is allocated with an orthogonal spectrum resource block. Furthermore, the large-scale fading is modeled by using a free-space propagation path-loss model. If the task of UE u_j offloaded to vehicular node v_i, latency $l_{i,j}^{t}$ of an offloaded task is depends on the available bandwidth, task size S_j, $j \in T$, and transmission data rate r_j. Since the task is completely offloaded to vehicular node $v_i \in F^v$, the transmission time required by u_j can be obtained as

$$l_{i,j}^{t} = \frac{S_j}{r_j} \tag{4}$$

$$r_j = B_{i,j} Log_2\Big(1 + \frac{P_{i,j}^{t} d_{i,j}^{-\alpha} |h_{i,j}|^2}{N_0}\Big) \tag{5}$$

where $d_{i,j}$ denotes the transmission distance between vehicular node v_i and UE u_j. $h_{i,j}$ is the Rayleigh channel coefficient with a complex Gaussian distribution. α denotes the path-loss exponent and N_0 is the power noise.

Task Computation Latency When the task is offloaded to VN v_i, execution time of the task is depends on computational size S_j of the task t_j and computation capacity U_i of VN v_i. Then the task computation latency $l_{i,j}^{c}$ can be expressed by

$$l_{i,j}^{c} = \frac{\delta_{i,j} \lambda S_j}{U_i} \tag{6}$$

where A binary variable $\delta_{i,j} \in (0,1)$ is defined to indicate whether the UE u_j task offloaded to VN v_i. Under the above system latency model, when task t_j of UE u_j is assigned to VN v_i, the total latency $L_{i,j}$ is a sum of task computation latency and task transmission latency. It can be expressed as

$$L_{i,j} = l_{i,j}^{c} + \eta\big(l_{i,j}^{t} + ß\big) \tag{7}$$

The communication link between the UE u_j and CN is considered to be one-hop. When a task t_j of UE u_j is offloaded to VN, the number of hops task traverse affects the overall latency. In this work, η represents the total number of hops task t_j traverse. ß refers to a round trip overhead between a VN v_i and a UE u_j.

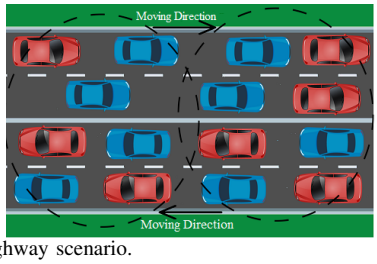


Fig. 2. The highway scenario.

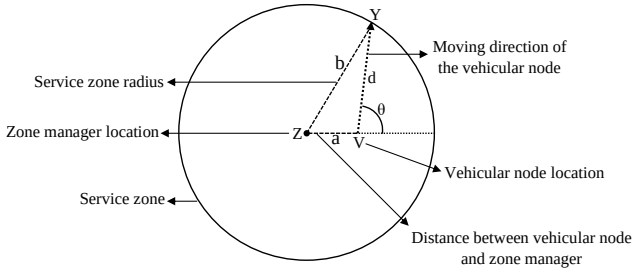


Fig. 3. Proposed mobility model for initialise VN probability

Definition 1 ***(Energy and Latency Efficient Cost (ELEC)).*** *ELEC is defined as the weighted sum of total energy consumption ($E_{i,j}$) and total latency ($L_{i,j}$). Thus the ELEC of UE u_j task is given by*

$$ELEC = \Psi_{i,j}^{e} E_{i,j} + \Psi_{i,j}^{l} L_{i,j} \tag{8}$$

where $\Psi_{i,j}^{e}$ and $\Psi_{i,j}^{l} \in [0,1]$ denote the two scalar weights of energy consumption and overall latency for CN f_k (k^{th} cloudlet node) making decision on UE u_j task, respectively. To provide rich modeling flexibility, we allow that CN changed weighting parameters ($\Psi_{i,j}^{e}$ and $\Psi_{i,j}^{l}$) as per the current characteristics (CPU-intensive and latency-sensitive) of UE task. For example, when a CN is at a low battery state and it is running CPU-intensive task which required more energy, then it would be better to put more weight ($\Psi_{i,j}^{e} \in [0,1]$) on energy consumption(a larger $E_{i,j}$) in the decision making, in order to save more energy. On the other-hand, when a CN is running a latency-sensitive UE task (video streaming), then put more weight ($\Psi_{i,j}^{l} \in [0,1]$) on the overall latency (a larger $L_{i,j}$), in order to reduce the delay.

C. Constraints

Mobility: The mobility of VNs plays a significant role in overall service latency and communication failure. Since we consider the highway scenario, we assume that VNs travel on a two-lane two-directional road is shown in Fig 2. Due to the fast mobility of the VN v_i, it might be move out the same service zone of UE u_j during data transmission, thereby an offloading failure. Therefore, the mobility model based on the article in [28], dwell time $D_{i,j}^{t}$ of VN v_i within the same *service zone* of UE u_j plays significant role because an offloading failure occurs if $l_{i,j}^{t} > D_{i,j}^{t}$. The dwell time of a VN is defined as a time of VN stay in one service zone. Therefore, estimating the probability of an offloading failure on VN v_i is $P_{i,j}^{F}$ within the *service zone* and probability of an offloading success is $P_{i,j}^{S}$.

As shown in Fig 3, we assume that *zone manager* of UE u_j *service zone* located at point Z having coverage radius of

b and the current location of VN v_i located at point V which are moving towards the point Y direction with a angle θ. The d and a denoted as a distance of moving VN v_i within the *service zone* of UE u_j and distance between *zone manager* & VN v_i respectively. Since we assume that all VNs are uniformly distributed within the *service zone* circle with fixed radius b [29], the probability density function (pdf) of VNs with Cartesian coordinates (x,y) is given as:

$$f(x,y) = \frac{1}{\pi b^2}, \quad \forall (x,y) \in (x^2 + y^2 \leq b^2) \tag{9}$$

Transforming equation (9) from Cartesian (x,y) to Polar (a,θ) coordinates is given by

$$f(a,\theta) = \frac{a}{\pi b^2}, \quad a \in [0,b] \quad \& \quad \theta \in [0,2\pi] \tag{10}$$

where $x = a\cos\theta, y = a\sin\theta$ and a denoted the distance between the VN v_i and *zone manager* within a *service zone* of UE u_j. Therefore, the marginal pdf of a can be derived by integrating out θ from equation (10) is given as

$$f_a(a) = \frac{2a}{b^2} \tag{11}$$

We assume that the current location of VN v_i and moving direction θ are independent. The moving direction θ of VN v_i follows the uniform distribution, where $\theta \in$ [0,2π] [30]. As we shown in Fig. 3, we can limit the value of $\theta \in [0,\pi]$ due to the symmetry of the *service zone*. Thus the joint pdf of a (distance between current location V of VN v_i and *service manager* location Z) and θ (moving direction of VN v_i) are given as

$$f_{a,\theta}(a,\theta) = \frac{1}{\pi} \times \frac{2a}{b^2} \tag{12}$$

Due to fast mobility of VN v_i might be move out the same *service zone* of UE u_j during the data transmission, as result of that an offloading failure. Therefore, we need to obtain the distribution of d (distance between current location V of VN v_i and end position Y of *service zone*) by determine its relationship with a and θ using law of cosines is given as $b^2 = a^2 + d^2 + 2ad\cos\theta$. Thus θ is a inverse cosine function can be written as $I(a,d) \equiv \arccos\left(\frac{b^2-a^2-d^2}{2ad}\right), d \in [b-r, b+r]$. It can be proved that $I(a,d)$ is a monotonically increasing function of d. Therefore, applying Jacobian transformation into equation 11 and making change of variable in inverse cosines function $I(a,d)$ [31]. Afterword the joint distribution of distance $|ZV|$ a and distance $|VY|$ d in Fig. 3 is given as

$$\begin{aligned} f_{d,a}(d,a) &= \frac{2a}{\pi b^2} \times det \begin{vmatrix} 1 & 0 \\ \frac{\delta I(a,d)}{\delta a} & \frac{\delta I(a,d)}{\delta d} \end{vmatrix} \\ &= \frac{2a}{\pi b^2} \times \left| \frac{\delta I(a.d)}{\delta d} \right| \end{aligned} \tag{13}$$

$where,$

$$\frac{\delta I(a,d)}{\delta d} = \frac{\frac{b^2-a^2+d^2}{2ad^2}}{\sqrt{1 - \left(\frac{b^2-a^2+d^2}{2ad^2}\right)^2}} \tag{14}$$

By the substituting of the equation (13) into equation (14) to derived distribution function $f_{d,a}(d,a)$:

$$f_{d,a}(d,a) = \frac{b^2 - a^2 + d^2}{\pi b^2 d^2 \sqrt{1 - \left(\frac{b^2-a^2+d^2}{2ad^2}\right)^2}} \tag{15}$$

Thus the marginal pdf of remaining distance of VN v_i in a UE u_j *service zone* d can be derived by integrating out a from equation (15) along with the integral region where VN v_i move from initial distance to diameter of *service zone* is $2b$, which is described in equation (10) is given as

$$f_d(d) = \int_{|b-d|}^{b} f(d,a)da, \quad d \in [0,2b] \tag{16}$$

With the help of equation (16) and (15), we can rewrite $f_d(d)$ function is given as

$$\begin{aligned} f_d(d) &= \frac{1}{\pi b^2} \int_{|b-d|}^{b} \frac{b^2 - a^2 + d^2}{d^2 \sqrt{1 - \left(\frac{b^2-a^2+d^2}{2ad^2}\right)^2}} da \\ &= \frac{1}{\pi b^2 d} \int_{|b-d|}^{b} \frac{2a(b^2 - a^2 + d^2)}{\sqrt{(2ad)^2 - (b^2 - a^2 + d^2)^2}} da \end{aligned} \tag{17}$$

By making change of variable as $x = a^2$, we have rewrite the equation (17)

$$f_d(d) = \frac{1}{\pi b^2 d} \int_{(b-d)^2}^{b^2} \frac{b^2 - x + d^2}{\sqrt{(2ad)^2 - (b^2 - x + d^2)^2}} dx \tag{18}$$

Then one more time we have apply change of variable rule as $t = b^2 - x + d^2$, we have rewrite the equation (18)

$$\begin{aligned} f_d(d) &= -\frac{1}{\pi b^2 d} \int_{2bd}^{d^2} \frac{t}{\sqrt{(2ad)^2 - t^2}} dt \\ &= \frac{1}{\pi b^2 d} \int_{d^2}^{2bd} \frac{t}{\sqrt{(2ad)^2 - t^2}} dt \\ &= -\frac{1}{\pi b^2 d} \sqrt{(4b^2 d^2 - t^2}\Big|_{d^2}^{2bd} \\ &= \frac{1}{\pi b^2} \sqrt{(4b^2 - d^2} \end{aligned} \tag{19}$$

To estimate the VN v_i dwell time $D_{i,j}^t$ in a *service zone* of UE u_j, simply use the average velocity $\vartheta_{i,j}$ and $d_{i,j}$ (distance between current location of v_i and end point Y of the *service zone* diameter in the VN heading direction). So, the dwell time of the VN v_i is derived as $D_{i,j}^t = \frac{d_{i,j}}{\vartheta_{i,j}}$. Thus the pdf of dwell time $D_{i,j}^t$ is given as

$$\begin{aligned} f_{D_{i;j}^t}(t) &= v_{i,j} f(t\vartheta_{i,j}) \\ &= \frac{\vartheta_{i,j}}{\pi b} \sqrt{4b^2 - (\vartheta_{i,j}t)^2} \end{aligned} \tag{20}$$

As we discussed earlier, an offloading failure of UE u_j task to VN v_i occurs if $l_{i,j}^t > D_{i,j}^t$. Therefore, dwell time also plays important role in data transmission because UE u_j can only offload the tasks to VN v_i when they remain connected. Thus, an offloading request is admissible *iff* $l_{i,j}^t < D_{i,j}^t$. So, the probability of an offloading failure within the communication range of UE u_j can be obtained as follows

$$\begin{aligned} P_{i,j}^F &= P(D_{i,j}^t < l_{i,j}^t) \\ &= \int_0^{l_{i,j}^t} f(t)dt \end{aligned} \tag{21}$$

We used integration by substitution rule in equation (21) to estimate the probability of an offloading failure.

$$P_{i,j}^{F} = \begin{cases} 1 & l_{i,j}^{t} \geq \frac{2b}{\vartheta_{i,j}}, \\ \dfrac{2arcsin\left(\frac{l_{i,j}^{t} v_{i,j}}{2b}\right) + l_{i,j}^{t} v_{i,j} \sqrt{1 - \left(\frac{l_{i,j}^{t} \vartheta_{i,j}}{2b}\right)^{2}}}{\pi} & 0 \leq l_{i,j}^{t} < \frac{2b}{\vartheta_{i,j}}, \end{cases}$$

Therefore, the probability of an offloading success on VN v_i within the *service zone* of UE u_j is derived as $P_{i,j}^{S} = 1 - P_{i,j}^{F}$. The higher value of $P_{i,j}^{S}$ means v_i is a most desirable VN for process offloaded task.

Remark: 2 In this work, we assume that the real-time GPS information of VN is known by the *zone manager*. The simulation scenario was built using a realistic vehicular mobility pattern from Luxembourg SUMO Traffic [32]. In this simulation scenario, 2070 different real-time bus traces were used to evaluate the $\vartheta_{i,j}$ and $D_{i,j}^{t}$.

Capacity: The offloaded task of UE u_j required certain amount of computing (C), network (N), and storage (M) resources from VN v_i to execute these tasks. However, the total demand received by a VN v_i can't exceed its capacity $\Omega_i(C, M, N)$. $\Omega_i(C, M, N)$ is denoted as the total resources capacity of VN v_i, and $\omega_j(C, M, N)$ denoted as the resources demand of UE u_j task. The resources capacity constraint is given as.

$$\sum_{\forall u_j \in U} \omega_j(C, M, N) \leq \Omega_i(C, M, N), \quad \forall v_i \in F^v \tag{22}$$

Assignment: A binary variable $\delta_{i,j}$ is defined to indicate whether the UE u_j task offloaded to VN v_i. The value of $\delta_{i,j} = 1$ indicate that offloaded UE u_j task is assign to VN v_i otherwise $\delta_{i,j} = 0$ indicate that offloaded UE u_j task is not assign to VN v_i. In assignment constraint, each offloaded UE u_j task assign to only one VN v_i. It is given as.

$$\sum_{\forall v_i \in F^v} \delta_{i,j} = 1, \quad \forall \quad u_j \in U \tag{23}$$

D. Problem Formulation

The purpose of this work is to relieve the heavy burden of the CN, minimise energy consumption and reduce the overall latency by leveraging the under-utilized resources of VNs. Hence, the energy-latency tradeoff is investigated and the joint objective function is defined as $\Psi_{i,j}^{e} E_{i,j} + \Psi_{i,j}^{l} L_{i,j}$, where $\Psi_{i,j}^{e}$ and $\Psi_{i,j}^{l} \in [0, 1]$ are two scalar weights. $\chi = \delta_{i,j}$ is used to assignment of task decision between n UEs and m VNs. $\delta_{i,j}$, is defined as a binary value. $\delta_{i,j} = 1$ means that the UE u_j offloaded task is assigned to VN v_i. Otherwise, $\delta_{i,j} = 0$. The optimization problem is formulated as:

$$\Upsilon : \min_{\chi} \sum_{i=1}^{n} \sum_{j=1}^{m} \delta_{i,j} (\Psi_{i,j}^{e} E_{i,j} + \Psi_{i,j}^{l} L_{i,j}) \tag{24}$$

$s.t.$

$$C_1 : L_{i,j} \leq L^{up}, \quad \forall v_i \in F^v, \forall u_j \in U,$$
$$C_2 : \sum_{u_j \in U} \omega_j(C, M, N) \leq \Omega_i(C, M, N), \quad \forall v_i \in F^v,$$
$$C_3 : \sum_{v_i \in F^v} \delta_{i,j} \leq 1, \quad \forall u_j \in U,$$
$$C_4 : \sum_{u_j \in U} \delta_{i,j} \leq 1, \quad \forall v_i \in F^v,$$
$$C_5 : P_{i,j}^{S} = 1 - P_{i,j}^{F}, \quad \forall v_i \in F^v, \forall u_j \in U,$$
$$C_6 : \omega_j(C, M, N) \geq 0, \quad \forall u_j \in U,$$
$$C_7 : \Psi_{i,j}^{e} + \Psi_{i,j}^{l} = 1$$

Here, constraint C_1 is latency constraint which specifies that the overall latency of UE u_j task is bounded by the required maximum latency (L^{up}). A second constraint (C_2) have to add to ensure that available resources of VN v_i is sufficient for processing the UE u_j tasks. C_4 and C_3 guarantee that there is a one-to-one correspondence among VNs and UEs. C_5 ensures that offloaded UE u_j task successfully assign to the VN v_i and v_i is within same *service zone* as UE u_j. C_6 is the non-negative constraint on the resources requirements of the UEs task. C_7 ensure that the sum of energy and latency weight parameters always equal to one.

Remark 3: It is infeasible to find a polynomial-time solution for Υ due to integer constraint $\delta_{i,j} \in \{0, 1\}$. It's made Υ an integer programming problem, which is in general NP-hard. The NP-hardness of the problem Υ implies that the problem size of Υ grows enormously fast with the number of UEs. Therefore, it is impractical to obtain a globally optimal solution in a real-time manner because the VN has to collect every detailed piece of information from all UEs. Thus, an efficient scheme design with low-complexity is highly desirable. Motivated with these facts we introduced the ECOS scheme based on a heuristic approach.

Algorithm 1: Cloudlet Overload Detection

Input: *The UE task workload* $T_{j,k}^{cpu}$ *and* $T_{j,k}^{mem}$ $\forall f_k \in F^f$
Output: *CloudletNodeOverload*

begin
  Sort All Element Of F^f *as per total utilization in descending order*
  function `isCloudletNodeOverUtilized(`Z_k^{cpu}, Z_k^{mem}`)`
  CloudletNodeOverload $\leftarrow$ *false*;
  $U_k^{cpu} \leftarrow \sum T_j^{cpu}, \quad \forall k \in F^f : \psi(k) = j;$
  $U_k^{mem} \leftarrow \sum T_j^{mem}, \quad \forall k \in F^f : \psi(k) = j;$
  if $(U_k^{cpu} > C_k^{cpu} \parallel U_k^{mem} > C_k^{mem})$ **then**
    CloudletNodeOverload $\leftarrow$ *true*;
  end
  return *CloudletNodeOverload*;
  end function
end

IV. Computation Offloading Mechanism for ECOS

In this section, we discuss the computation offloading procedure in detail. Firstly, we provide a brief introduction

of the CN overload prediction policy. Then, we introduce the computation offloading selection policy. Next, we provide a detail introduction of the resources allocation policy. Finally, we analyze the complexity properties.

A. Phase 1: Cloudlet Overload Prediction (COP) Policy

Instead to predict all overloaded CNs, our proposed algorithm (shown in Algorithm 1) gives only one overloaded CN at a time and solves a smaller optimization problem to decide where to offload UEs task that is currently processing on the predicted overloaded CN. In each CN in the *service zone* is dedicated a COP policy module. The COP policy module assigned for fetches required CPU and memory demands for the UE task that is executing on the CN f_k and calculates predicted aggregate CPU and memory demands of CN f_k. Whenever new task comes to CN for processing the COP policy will estimate the overall current workload of CN, if the current workload exceeding the capacity of the resources of CN then the current CN expected to an overload. The aggregate CPU and memory demands of CN f_k is given as.

$$U_k^{cpu} = \sum T_j^{cpu}, \quad \forall k \in F^f : \psi(k) = j \tag{25}$$

$$U_k^{mem} = \sum T_j^{mem}, \quad \forall k \in F^f : \psi(k) = j \tag{26}$$

where T_j^{cpu} and T_j^{mem} denoted UE u_j task required the CPU and memory for execution respectively. The CN f_k CPU and memory capacity denoted as C_k^{cpu} and C_k^{mem}, respectively. $\psi : U \rightarrow F^f$ is the UE to CN mapping function, with $\psi(k) = j$ meaning that UE u_j task is executed on CN f_k. Afterwords, *zone manager* compares the calculated aggregate memory and CPU demands, U_k^{mem} and U_k^{cpu} respectively. If $U_k^{cpu} > C_k^{cpu}$ or $U_k^{mem} > C_k^{mem}$, then the COP policy module notifies the *zone manager* that CN f_k is expected to have an overload in the coming period. The COP policy module also forwards to the *zone manager* the predicted CPU and memory demands, T_j^{cpu} and T_j^{mem}, for each UE u_j task processed on CN f_k. These predictions will be used for selecting the computation task for offloading.

An Algorithm 1 shown a basic procedure of detecting overload CN. At first, the algorithm procedure inquires in the context of the current CN utilization. The current utilization of CN is calculated using equation 25 and equation 26 (from line 3 to line 5). If CPU and memory utilization of the current node exceed overall resources capacity then this CN recognised as an overloaded node. Afterward, *zone manager* gather current information and enable offloading mode.

B. Phase 2: Computation Offloading Selection (CoS) Policy

The *zone manager* equipped with the CoS module which is responsible for selecting a task from overloaded CN. Computation offloading selection is an initial task after CN f_k detected overload. The CoS policy aims to determine which tasks of UEs (where $\forall k \in F^f : \psi(k) = j$) are offloaded onto the VN such that cost of energy consumption, cost of offloading, and risk of overload is minimized while task-precedence requirement is preserved. This policy selects a UE u_j task for offloading from overload CN f_k given as.

Algorithm 2: Computation offloading Selection

Input: *CloudletOverload*
Output: *OffloadingTask*

```
1  begin
2      TaskCount ← Min;
3      OffloadingTask ← Null;
4      TaskList ← CloudletOverlod.getTaskList();
5      OptimalNodeCluster← Null;
6      for (task:TaskList) do
7          TaskRatio ← S^k_{i,j};
8          if (TaskRatio > TaskCount) then
9              TaskCount ← TaskRatio;
10             OffloadingTask ← task
11         end
12     end
13     return OffloadingTask
14 end
```

$$max\Big[S_{i,j}^k = \frac{T_j^{cpu}}{T_j^{mem}}, \quad \forall k \in F^f : \psi(k) = j\Big] \tag{27}$$

It is clear that the risk of CN overload will minimise by maximise the equation (27). The intuition here is that select a high CPU task j which required higher CPU utilization and lower memory utilization, thereby reduce the risk of overload and reduce offloading cost by lowering the network bandwidth requirement. A lower value of T_j^{mem} means that task has less memory requirement need to transfer or offload less number of bits from CN f_k to VN v_i, thereby cost of offloading also reduce. It means CPU utilization directly proportional to the CN overload, and the required memory (size of the task) is directly proportional to the offloading cost. Therefore, select an UE u_j task ($j \in U$) with the maximum value of $S_{i,j}^k$ for task offloading.

An Algorithm 2 shown the basic procedure to select a computation offloading task from overloaded CN, which helps to reduces offloading cost and service latency. At first, the algorithm makes a list of tasks running on overloaded CN (line 4). Next, it calculate TaskRatio of each task on that list using the equation (27) (line 7). Finally, the proposed algorithm selects a task from overloaded CN whose TaskRatio maximum than other tasks on the same overloaded node (from line 8 to 11 line). The next phase is a resource allocation for computation offloading.

C. Phase 3: Resources Allocation for Computation Offloading

Instead of deciding where to assign all UEs computation tasks that are currently processing on the overloaded CNs, our proposed heuristic algorithm (shown in Algorithm 3) takes only one UE task which is selected using Algorithm 2. Another feature adopted by a proposed heuristic that minimise the complexity further is to consider only a set of VNs within the same *service zone* of UE. The real-time mobility of the VNs formed by the mobility function ($PickSameZoneNodes$) using $P_{i,j}^S = 1 - P_{i,j}^F$. The all VNs giving real-time information about their location to the *zone manager* for providing all this valuable information to Algorithm 3 module. Using this information Algorithm 3 module calculate $P_{i,j}^S$ of all VNs in

Algorithm 3: Resources allocation for Offloaded Computation

Input: *Selected Offloaded UE Task*
Output: Allocation of Offloaded UE Task

```
1  begin
2      MinELAC← Max;
3      AllocateVehicularNode← Null;
4      SameZoneNodeList ← PickSameZoneNodes();
5       Sort SameZoneNodeList in descending order;
6      for (VehicularNode:SameZoneNodeList) do
7          if (Vehicular node has enough resorces & satisfied all
             constraints) then
8              valueELAC ← EstimateELAC();
9              if (valueELAC< MinELAC) then
10                 AllocateVehicularNode← VehicularNode;
11                 MinELAC ← valueELAC;
12             end
13         end
14     end
15     return Allocation of Offloaded UE Task
16 end
```

the same *service zone* as overloaded CN and then Algorithm 3 module returned list of VN with their $P^S_{i,j}$ (line 4). Next, using the $P^S_{i,j}$ value of all VNs in the return list is sorted in descending order. The intuition here is that the VN with the largest $P^S_{i,j}$ value has the highest probability to finish the processing of offloaded task before leaving a *service zone*. The heuristic then iteratively compares joint energy and latency cost of $\forall$ VN $\in SameZoneNodeList$. Afterward, it returned efficient VN (from line 5 to line 15) that have a higher probability to stay longer time in a UE *service zone* and minimum value of ELEC metric. The intuition here is that the VN the higher probability to stay longer time in a UE *service zone* have enough time for processing the offloaded task which lower chances of offloading failure. The heuristic algorithm will be executed until adding any of the possible options such as offloading task causes in-feasibility, and the minimum optimal joint energy and latency cost of task offloaded to VN is larger than the minimum joint cost of the previous iteration. Hence, the proposed heuristic algorithm minimized offloading cost, energy consumption, overall latency, and probability to trigger an overload CN.

D. Time Complexity and Overhead Analysis

In the process of overload CN detection, the sorting of all CNs and selecting an overloaded CN are $O(l \log l)$ and $O(l)$, respectively. Assuming worst-case scenario that all UEs task running on overload CN then time complexity for selection computation offloading task is $O(m)$. In the process of resources allocation for computational offloading, for each selected UE task, the complexity is $O(n)$ then total complexity of all UEs task is $O(mn)$.

The real-time mobility information gathering from VNs is the major overhead while resource allocation in phase-3 considering the scaled vehicular fog networking scenario in realistic traffic environment. However, our consideration of zone-based implementation of computation offloading and resource allocation effectively address this concern for real traffic environment.

TABLE I
SIMULATION PARAMETERS SETTING

Parameter	Value
Number of UEs	5-20
Velocity of vehicular node	2 - 20 m/s
Cell diameter	300 - 600 m
Latency constraint L^{up}	0.05 - 0.7 s
Cloudlet layer resources	2-4GHz, 2-3GB
Noise power	-114 dBm
Data size of UE's task	100 - 200 Mb
Path loss exponent	-3.4
Bandwidth of UEs	20 MHz
Transmission power of UEs	31 dBm
Efficient factor of power amplifier	18
Static transmission power	26 dBm

V. PERFORMANCE EVALUATION

In particular, we evaluate the ELEC metric of the proposed ECOS scheme by numerical analysis in Section 5B and demonstrate the impact of weights, $\Psi^e_{i,j}$ and $\Psi^l_{i,j}$ on the total latency and the energy consumption in Section 5D. Furthermore, we compare energy saving, total latency, and the probability of moving out in Sections 5C, 5E, and 5F, respectively.

We compare the proposed ECOS scheme with the following approaches:

1) LocalOnly: in which all UE task processed locally in the CN.

2) CloudOnly: in which tasks from overloaded CN offload only to the centralized cloud server [4].

3) Random: in which computation offloading task select randomly and randomly allocate computation resources to offloaded task without considering the mobility pattern of VN.

4) Chen: in which computation tasks offload to the cloud servers and an offloading decision is depend on the application characteristics [33].

5) Sun: in which computation tasks offload to the cloud servers and an offloading decision is based on the multi-armed bandit theory [9].

6) Mao: in which computation offloading policy based on Lyapunov optimization. At each time slot task is offloaded to the node without considering mobility [10] .

A. Simulation Setup Under Synthetic Scenario

To evaluate the performance of the proposed ECOS scheme, we carry out simulations in this section. We first consider a simple urban simulation scenario. In this synthetic scenario, parameters are based on the simulation scenario was built using real-world mobility traces of buses to simulate vehicular mobility patterns from Luxembourg SUMO Traffic [32]. We evaluate proposal solution under different VN mobility speed and different density scenario. The overall parameters used in our simulation are shown in Table I. the performance of the

proposed ECOS scheme is evaluated by extensive simulation on Matlab. Then simulate a realistic highway scenario using system level simulator VeinsLTE [34] to further verify the proposed ECOS scheme.

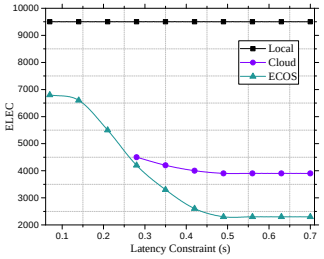


Fig. 4. ELEC vs L^{up}

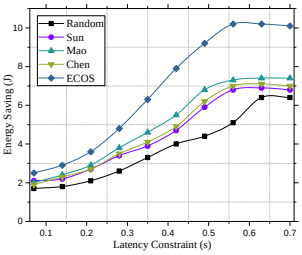


Fig. 5. Energy Saving vs L^{up}.

1) Comparison of Energy and Latency Efficient Cost: In this subsection, we compare the proposed ECOS scheme with the other two baselines execution policy, i.e., CloudOnly policy and LocalOnly policy [35], under the hard latency deadline constraint L^{up}. CloudOnly policy demonstrates that all UEs tasks are offloaded to the cloud servers for execution. LocalOnly policy implies that all UEs tasks are executed locally on the CN. Fig 4 plots the comparison of energy and latency efficient costs of the between proposed ECOS scheme, CloudOnly policy, and LocalOnly policy for the same application profile.

We can draw several observation from Fig. 4. Firstly, compared to LocalOnly policy, the proposed ECOS scheme significantly minimize 47% the energy and latency efficient cost. This is because that proposed ECOS scheme can optimally select UEs $u_j \; \forall \in U$ task to be offloaded on the VN v_i to execute according to the energy consumption cost and overall service latency. Secondly, compared to CloudOnly policy, the proposed ECOS scheme has also lower energy and latency efficient cost when CloudOnly policy becomes applicable. In the case of low latency deadline constraint (L^{up}), CloudOnly policy for the cloud execution can't be used because of longer time taken by the CloudOnly Policy for the remote execution to transmit the input data, which violate constraint C_1 of the UE task. Thirdly, as L^{up} value increases the energy and latency efficient cost decreases rapidly, then become stable when $L^{up} = 0.5$. This is because at the low value of L^{up} , most of the UEs tasks are executed on the CN. Moreover, Whenever the value of L^{up} is low the task process in CN or locally because lower value of L^{up} violate service latency deadline constraint, thereby task could not offloaded to VN.

2) Comparison of Energy Saving: In this subsection, we first consider the impact of the latency constraint L^{up} on the energy-saving with respect to LocalOnly method. The proposed ECOS scheme compare with the offloading game policy in [33] named Chen, the task scheduling algorithm in [10] named Mao, the resources allocation policy in [9] named Sun, and the random allocation algorithm for different value of L^{up}. These policies are implemented on the real testbed with same parameter setting and channel condition. Fig. 5 depicts the energy saving of the above given policies.

We can observe that the proposed ECOS scheme always has more energy saving compared to Random, Sun, Mao, and Chen policies, which is justified since ECOS scheme not only select an efficient task from overloaded CN for reducing communication cost but also takes energy-efficient offloading mechanism to save more energy. Fig. 5 shows that as the value of L^{up} increases, the number of offloading tasks increases from overloaded CN f_k to VN v_i for exploit under-utilized resources rather than being processed by the overloaded CN with limited computation resources. This will dramatically save more energy. Numerical results also demonstrate that the proposed ECOS scheme can achieve close to optimal performance under all the investigated scenarios.

In Fig.6 shows the energy-saving versus the *service zone* coverage diameter with different numbers of UEs. Expanding the coverage area of *service zone* has a positive impact on energy saving. It does not only reduce the probability of offloading failures which satisfied constraint C_5 but also increase the eligible VNs for processing offloaded tasks. However, when the *service zone* coverage area reaches a certain value, the improvement in energy saving saturated because the optimal energy and latency have already been achieved. Furthermore, when the number of UEs is increases, energy-saving also increases. The reason is that numbers of offloaded UEs tasks from overloaded CN reduce the energy consumption of CN.

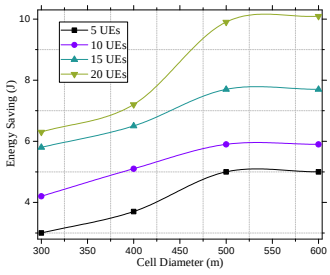


Fig. 6. Energy vs Cell diameter

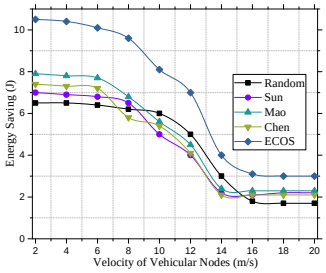


Fig. 7. Energy vs VN's Velocity

Fig. 7 shows energy-saving versus velocity of VN. Increasing the velocity of VN has a negative impact on energy saving. It does not only increase the chances of offloading failures but also maximize the energy consumption of overloaded CN. The reason is that the number of eligible VN for processing offloaded tasks are decreases because more and more VN leaving the *service zone* that violate the probability constraint C_5. Thereby, it is less likely for offloading UE task to satisfactory VN, and the corresponding task can only be executed locally on overloaded CN. In comparison, energy-saving saturated constant as the velocity of VN is increased. Simulation results demonstrate that the proposed ECOS scheme is more robust to negative impact caused by the high mobility of VN.

3) Impact of Weights $\Psi^e_{i,j}$ and $\Psi^l_{i,j}$: In this subsection, we evaluate the impact of weights, $\Psi^e_{i,j}$ and $\Psi^l_{i,j}$ on ELEC metric and overall latency. Fig. 8 depicts the comparison of latency versus number of UEs for different setting of $\Psi^e_{i,j}$ and $\Psi^l_{i,j}$. We can observe that increasing the value $\Psi^l_{i,j}$ has a positive impact on overall latency of the task. If the application is latency-sensitive, a large value of $\Psi^l_{i,j}$ is beneficial to improve the Quality of Experience (QoE). Fig. 9 shows total reverse

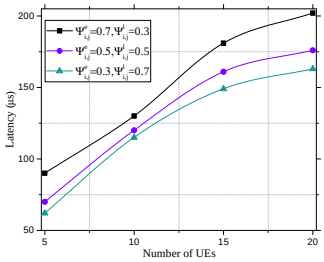


Fig. 8. Latency vs UEs

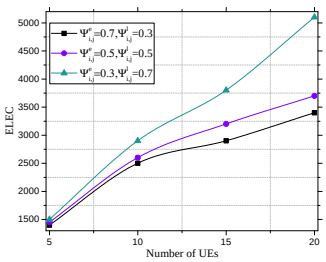


Fig. 9. Energy vs VN's Velocity

relation to Fig. 8. We can observe that increasing value $\Psi^l_{i,j}$ has a negative impact on ELEC metric but increasing value $\Psi^e_{i,j}$ has a positive impact on ELEC metric. Here, we can observe that if the application is not latency-sensitive, a large value of $\Psi^e_{i,j}$ saves more energy for overloaded CN. The reason is that a large $\Psi^e_{i,j}$ increases the number of eligible VN for executing offloaded task from overloaded CN.

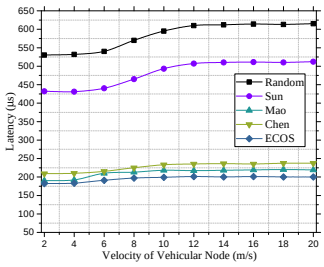


Fig. 10. Latency vs VN's Velocity

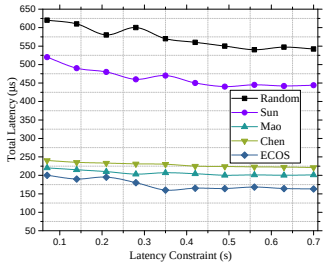


Fig. 11. Latency vs L^{up}

4) Comparison of Total Latency: In this subsection, we compare the overall service latency versus velocity of VN and latency constraint L^{up}. In the simulation, we employ the constant-velocity model [13], [36], and randomly generated velocity of VN within the range (from 2 to 20) meter/second. Fig. 10 plots total latency for a different settings of VN velocity. We can observe from Fig. 10 that the total latency increases with the VN velocity. The reason is that higher velocity leads to frequently offloading failures. Thereby, the offloading cost or communication cost of the UE task increases. In comparison, the total latency of the proposed scheme remains constant when the VN velocity increases from 12 to 20 m/s. The reason is that UE tasks are less likely to satisfied the constraint C_5, which means VN moving out from *service zone* communication range. Simulation results demonstrate that the proposed ECOS scheme is more robust to the negative impact caused by the high velocity of VN.

Fig. 11 shows the total latency versus latency constraint of the computation task. The number of eligible VNs increases, as the latency constraint increases. Thereby, more and more task can be offloaded from overloaded CNs to VN to exploit underutilized computational resources, which is dramatically reduce the total latency for UE. Simulation results also demonstrate that the proposed ECOS scheme giving optimal performance compare to other policies under all investigated scenarios.

5) Comparison Probability of Moving Out : In this subsection, we evaluate the impact of VN mobility on the probability of VN moving out from UE u_j *service zone* range with different setting of *service zone diameter* (from 300 to 600 m). Fig. 12 shows the comparison of VN probability of moving out from *service zone* range plotted against the velocity of VN. The simulation results demonstrate that the probability of VN moving out from *service zone* dramatically increases with the increment of VN velocity or mobility. Clearly, as the velocity of VN increases, the proportion of satisfactory VNs decreases for processing offloaded task but we can also observed that increasing the diameter of *service zone* positively impact on moving out probability.

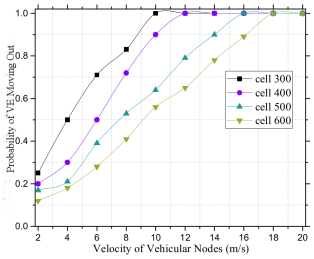


Fig. 12. VN's Velocity vs VN's Probability

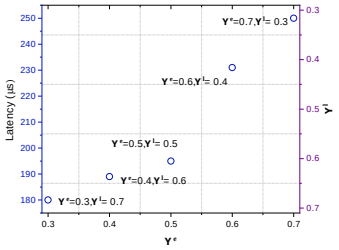


Fig. 13. Latency vs weight parameters at the arrival probability of VN is 0.1

B. Simulation Setup Under Realistic Highway Scenario

We consider system-level simulator VeinsLTE [34], in order to evaluate the average latency of the proposed scheme under a realistic highway scenario. The simulation platform VeinsLTE combined with a traffic Simulation of Urban Mobility (SUMO), a network simulator OMNeT++, and enables to use real maps from Open Street Map (OSM). Specifically, SUMO manages the mobility of the vehicular node, while with the framework VeinsLTE implementation, VNs can either exchange data with CN through Dedicated Short-range Communication (DSRC), or connect to zone manager over LTE networking technologies. In this work, VNs are used DSRC channels for data transmission between CNs. On the other hand, LTE is used for communications between VNs and the corresponding zone manager to notifying the current status of VNs, such as leaving or entering a service zone. To obtain a realistic simulation scenario, we used an 8 km segment of the Jingzang highway in Beijing, exported from OSM, and used it in our simulation, with two lanes and two ramps. We have homogeneously deployed a set of the RUs to cover the whole area. We define each RUs position based on its communication range. These RUs work as CNs, and thus each CN is responsible for managing UEs tasks within its coverage area based on services available in that area. We also define the zone managers, which can communicate with CNs and VNs. The maximum speed of VN is set to 20 m/s. The arrival of VNs is modeled by Bernoulli distribution, with probability ranging from 0.1 to 0.2.

1) Comparison of Latency: Fig. 13 and Fig. 14 explore the impact of VNs arrival rate on latency. The number of VNs in a corresponding *service zone* of UE increases, as the probability of arrival rate increases from 0.1 to 0.2. As a result, The number of eligible VNs increases for processing

UE tasks. Thereby, more and more tasks can be offloaded from overloaded CNs to VN to exploit under-utilized computational resources, which dramatically reduces the service latency for UE. Simulation results also demonstrate that the proposed ECOS scheme significantly reduces the service latency.

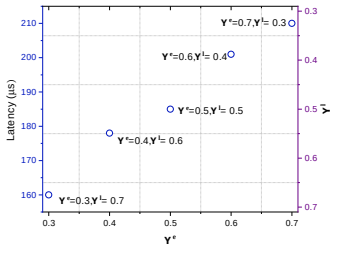


Fig. 14. Latency vs weight parameters with 0.2 VN arrival probability

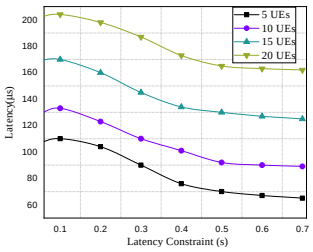


Fig. 15. Latency vs UEs at the arrival probability of VNs is 0.2

In Fig.15 shows the latency versus the latency constraint with different numbers of UEs. Increases latency constraint has a positive impact on service latency. It not only reduces the probability of offloading failures that satisfy constraint C_5 but also increases the eligible VNs for processing offloaded tasks. However, when the latency constraint reaches a certain value, the improvement in service latency saturated because the optimal latency has already been achieved. Furthermore, when the number of UEs is increases, service latency also increases. The reason is that the number of offloaded UEs tasks from overloaded CN is increased.

VI. Conclusions

In this paper, we have investigated joint energy and latency efficient task offloading and computation resources allocation for VFC with mobility and end-to-end latency constraints. We proposed an ECOS scheme to minimize the weighted sum of service latency and energy consumption of CN. We find that computation offloading selection policy is determined by not only the computing workload of a task but also memory used by the task. Furthermore, we observe that efficient tradeoff of energy and latency of task depends on the balance between the weight of computation latency and that of energy consumption, and the cost for the required application completion time deadline. Finally, we implement the ECOS scheme in a real based scenario and simulation results demonstrate that compared to the baseline policies, the proposed scheme can effectively minimize the energy consumption and service latency, by taking advantage of mobility constraint.

For future work, we will investigate the inter-operability of vehicular nodes. Because many edge facilities are involved in the task, it is important to seek an effective common inter-operability framework for heterogeneous devices and communication technologies.

Acknowledgment

This work is supported in part by the Key-Area Research and Development Program of Guangdong Province under grant No. 2019B010136001, the National Key Research and Development Plan under grant No. 2017YFB0801801, the National Natural Science Foundation of China (NSFC) under grant No. 61672186, 61872110, support this work. Corresponding author is Professor Weizhe Zhang.

References

[1] O. Kaiwartya, A. H. Abdullah, Y. Cao, J. Lloret, S. Kumar, R. R. Shah, M. Prasad, and S. Prakash, "Virtualization in wireless sensor networks: Fault tolerant embedding for internet of things," *IEEE Internet of Things Journal*, vol. 5, no. 2, pp. 571–580, 2017.

[2] 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 Transactions on Vehicular Technology*, vol. 68, no. 4, pp. 3113–3125, 2019.

[3] R. Yadav, W. Zhang, O. Kaiwartya, P. R. Singh, I. A. Elgendy, and Y.-C. Tian, "Adaptive energy-aware algorithms for minimizing energy consumption and sla violation in cloud computing," *IEEE Access*, vol. 6, pp. 55 923–55 936, 2018.

[4] R. Yadav, W. Zhang, K. Li, C. Liu, M. Shafiq, and N. K. Karn, "An adaptive heuristic for managing energy consumption and overloaded hosts in a cloud data center," *Wireless Networks*, pp. 1–15, 2018.

[5] Y. Cao, H. Song, O. Kaiwartya, B. Zhou, Y. Zhuang, Y. Cao, and X. Zhang, "Mobile edge computing for big-data-enabled electric vehicle charging," *IEEE Communications Magazine*, vol. 56, no. 3, pp. 150–156, 2018.

[6] Z. Zhou, H. Liao, B. Gu, S. Mumtaz, and J. Rodriguez, "Resource sharing and task offloading in iot fog computing: A contract-learning approach," *IEEE Transactions on Emerging Topics in Computational Intelligence*, 2019.

[7] B. Intelligence, "Automotive industry trends: Iot connected smart cars & vehicles," Dec. 2016, https://www.concentrix.com/infographs/connected-vehicles.pdf.

[8] A. Aliyu, A. H. Abdullah, O. Kaiwartya, Y. Cao, M. J. Usman, S. Kumar, D. Lobiyal, and R. S. Raw, "Cloud computing in vanets: architecture, taxonomy, and challenges," *IETE Technical Review*, vol. 35, no. 5, pp. 523–547, 2018.

[9] Y. Sun, X. Guo, S. Zhou, Z. Jiang, X. Liu, and Z. Niu, "Learning-based task offloading for vehicular cloud computing systems," in *2018 IEEE International Conference on Communications (ICC)*. IEEE, 2018, pp. 1–7.

[10] Y. Mao, J. Zhang, S. Song, and K. B. Letaief, "Power-delay tradeoff in multi-user mobile-edge computing systems," in *2016 IEEE Global Communications Conference (GLOBECOM)*. IEEE, 2016, pp. 1–6.

[11] T. Darwish, K. A. Bakar, O. Kaiwartya, and J. Lloret, "Trading: Traffic aware data offloading for big data enabled intelligent transportation system," *IEEE Transactions on Vehicular Technology*, 2020.

[12] E. Lee, E.-K. Lee, M. Gerla, and S. Y. Oh, "Vehicular cloud networking: architecture and design principles," *IEEE Communications Magazine*, vol. 52, no. 2, pp. 148–155, 2014.

[13] X. Hou, Y. Li, M. Chen, D. Wu, D. Jin, and S. Chen, "Vehicular fog computing: A viewpoint of vehicles as the infrastructures," *IEEE Transactions on Vehicular Technology*, vol. 65, no. 6, pp. 3860–3873, 2016.

[14] C. Zhu, J. Tao, G. Pastor, Y. Xiao, Y. Ji, Q. Zhou, Y. Li, and A. Ylä-Jääski, "Folo: Latency and quality optimized task allocation in vehicular fog computing," *IEEE Internet of Things Journal*, 2018.

[15] Z. Su, Y. Hui, and S. Guo, "D2d-based content delivery with parked vehicles in vehicular social networks," *IEEE Wireless Communications*, vol. 23, no. 4, pp. 90–95, 2016.

[16] A. Zanni, S.-Y. Yu, P. Bellavista, R. Langar, and S. Secci, "Automated selection of offloadable tasks for mobile computation offloading in edge computing," in *2017 13th international conference on network and service management (CNSM)*. IEEE, 2017, pp. 1–5.

[17] J. L. D. Neto, S.-Y. Yu, D. F. Macedo, J. M. S. Nogueira, R. Langar, and S. Secci, "Uloof: A user level online offloading framework for mobile edge computing," *IEEE Transactions on Mobile Computing*, vol. 17, no. 11, pp. 2660–2674, 2018.

[18] Z. Zhou, H. Liao, X. Zhao, B. Ai, and M. Guizani, "Reliable task offloading for vehicular fog computing under information asymmetry and information uncertainty," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 9, pp. 8322–8335, 2019.

[19] K. Zhang, Y. Mao, S. Leng, Y. He, and Y. Zhang, "Mobile-edge computing for vehicular networks: A promising network paradigm with predictive off-loading," *IEEE Vehicular Technology Magazine*, vol. 12, no. 2, pp. 36–44, 2017.

[20] J. Feng, Z. Liu, C. Wu, and Y. Ji, "Ave: Autonomous vehicular edge computing framework with aco-based scheduling," *IEEE Transactions on Vehicular Technology*, vol. 66, no. 12, pp. 10 660–10 675, 2017.

[21] X. Chen and L. Wang, "Exploring fog computing-based adaptive vehicular data scheduling policies through a compositional formal method-pepa," *IEEE Communications Letters*, vol. 21, no. 4, pp. 745–748, 2017.

[22] M. Gerla, J. T. Weng, and G. Pau, "Pics-on-wheels: Photo surveillance in the vehicular cloud," in *International Conference on Computing*, 2013.

[23] Z. Zhou, J. Feng, Z. Chang, and X. Shen, "Energy-efficient edge computing service provisioning for vehicular networks: A consensus admm approach," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 5, pp. 5087–5099, 2019.

[24] B. Ottenwälder, B. Koldehofe, K. Rothermel, and U. Ramachandran, "Migcep: operator migration for mobility driven distributed complex event processing," in *Proceedings of the 7th ACM international conference on Distributed event-based systems*. ACM, 2013, pp. 183–194.

[25] V. B. Souza, X. Masip-Bruin, E. Marin-Tordera, W. Ramirez, and S. Sanchez, "Towards distributed service allocation in fog-to-cloud (f2c) scenarios," in *Global Communications Conference*, 2017.

[26] C. Xian, Y.-H. Lu, and Z. Li, "Dynamic voltage scaling for multitasking real-time systems with uncertain execution time," *IEEE Transactions on computer-aided design of integrated circuits and systems*, vol. 27, no. 8, pp. 1467–1478, 2008.

[27] J. Cheng, Y. Shi, B. Bai, and W. Chen, "Computation offloading in cloud-ran based mobile cloud computing system," in *Communications (ICC), 2016 IEEE International Conference on*. IEEE, 2016, pp. 1–6.

[28] K. Mohd Zaini, A. R. Mohd Shariff, and Z. Shi, "A calculation of wlan dwell time model for wireless network selection," *Journal of Telecommunication, Electronic and Computer Engineering*, vol. 8, no. 10, pp. 73–76, 2016.

[29] M. Abdulla and Y. Shayan, "Cellular-based statistical model for mobile dispersion," in *2009 IEEE 14th International Workshop on Computer Aided Modeling and Design of Communication Links and Networks*. IEEE, 2009, pp. 1–5.

[30] I. F. Akyildiz and W. Wang, "The predictive user mobility profile framework for wireless multimedia networks," *IEEE/ACM Transactions On Networking*, vol. 12, no. 6, pp. 1021–1035, 2004.

[31] J. A. Gubner, *Probability and random processes for electrical and computer engineers*. Cambridge University Press, 2006.

[32] L. Codeca, R. Frank, and T. Engel, "Luxembourg sumo traffic (lust) scenario: 24 hours of mobility for vehicular networking research," in *2015 IEEE Vehicular Networking Conference (VNC)*. IEEE, 2015, pp. 1–8.

[33] X. Chen, "Decentralized computation offloading game for mobile cloud computing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 26, no. 4, pp. 974–983, 2014.

[34] F. Hagenauer, F. Dressler, and C. Sommer, "Poster: A simulator for heterogeneous vehicular networks," in *2014 IEEE Vehicular Networking Conference (VNC)*. IEEE, 2014, pp. 185–186.

[35] W. Zhang, Y. Wen, K. Guan, D. Kilper, H. Luo, and D. O. Wu, "Energy-optimal mobile cloud computing under stochastic wireless channel," *IEEE Transactions on Wireless Communications*, vol. 12, no. 9, pp. 4569–4581, 2013.

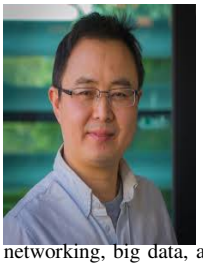
[36] Z. Su, Q. Xu, Y. Hui, M. Wen, and S. Guo, "A game theoretic approach to parked vehicle assisted content delivery in vehicular ad hoc networks," *IEEE Transactions on Vehicular Technology*, vol. 66, no. 7, pp. 6461–6474, 2016.

Rahul Yadav (M'20) is currently working as a Postdoctoral Research Fellowat the Peng Cheng Laboratory. He received his Ph.D. degree in Computer Science from Harbin Institute of Technology, Harbin, China, in 2020. He served as a Guest Editor in Wireless Communications and Mobile Computing journal. He is actively involved in the research on IoT, computation offloading, efficient-energy management, cloud/fog/edge computing, Vehicular fog computing, optimal utilization of data center resources, cost-efficient virtual machine consolidation, and delay estimation.

Weizhe Zhang (M06) is currently a Professor with the School of Computer Science and Technology, Harbin Institute of Technology, China. He has authored over 100 academic papers in journals, books, and conference proceedings. His research interests are primarily in parallel computing, distributed computing, cloud and grid computing, and computer network. He serves on a number of journal editorial boards. He has edited more than 10 international journal special issues as a guest editor and has served for many international conferences as the chair or a committee member

Omprakash Kaiwartya (M14, SM'19) is currently working as a Senior Lecturer at the Department of Computer Science, Nottingham Trent University, UK. Previously, He was a Research Associate at the Northumbria University, Newcastle, UK, in 2017 and a Postdoctoral Research Fellow at the University of Technology Malaysia (UTM) in 2016. He received his Ph.D. degree in Computer Science from Jawaharlal Nehru University, New Delhi, India, in 2015. He is the Fellow in Higher Education Academy (FHEA), UK. He is also Senior Member in IEEE, USA and Professional Member in British Computer Society (BCS), UK. His research interest includes Drone Enabled Networking, E-Mobility Centric Electric Vehicles, IoT Enabled Smart Services, Connected Vehicles, and Next Generation Wireless Systems. He is serving as Associate Editor and/or Guest Editor in IEEE Internet of Things Journal, IEEE Access, IET Intelligent Transport Systems, EURASIP Journal on Wireless Communication and Networking, MDPI Sensors and Electronics.

Houbing Song (M12SM14) received the Ph.D. degree in electrical engineering from the University of Virginia, Charlottesville, VA, in August 2012, and the M.S. degree in civil engineering from the University of Texas, El Paso, TX, in December 2006. In August 2017, he joined the Department of Electrical, Computer, Software, and Systems Engineering, Embry-Riddle Aeronautical University, Daytona Beach, FL, where he is currently an Assistant Professor and the Director of the Security and Optimization for Networked Globe Laboratory (SONG Lab, www.SONGLab.us). He served on the faculty of West Virginia University from August 2012 to August 2017. In 2007 he was an Engineering Research Associate with the Texas A&M Transportation Institute. He is the author of more than 100 articles. His research interests include cyber-physical systems, cybersecurity and privacy, internet of things, edge computing, big data analytics, unmanned aircraft systems, connected vehicle, smart and connected health, and wireless communications and networking. His research has been featured by popular news media outlets, including USA Today, U.S. News & World Report, Fox News, Forbes, WFTV, and New Atlas.



Shui Yu received the Ph.D. degree in computer science from Deakin University, Geelong, VIC, Australia, in 2004. He is a Professor with the School of Computer Science, University of Technology Sydney, Sydney, NSW, Australia. He initiated the research field of networking for big data in 2013. His H-index is 37. He has published two monographs and edited two books, over 280 technical papers, including top journals and top conferences. His current research interests include security and privacy, networking, big data, and mathematical modeling.