© 2022 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works. [10.1109/TITS.2022.3186401]

FedServ: Federated Task Service in Fog-Enabled Internet of Vehicles

Minu Tiwari, Ilora Maity, *Member, IEEE*, and Sudip Misra, *Fellow, IEEE*

***Abstract*—In this paper, we present FedServ, a federated task service system for fog-enabled Internet of vehicles (IoV), using which we aim to minimize the delay of vehicle task service. FedServ considers different task service requirements such as delay, computation-intensiveness, processing units required, and energy consumption. The existing literature mainly focuses on the delay requirement of tasks and considers the energy consumption for task processing, whereas FedServ aims to achieve the Quality of Service (QoS) of the task while optimizing different resource requirements for task service. FedServ provisions the federated task service by fragmenting the tasks, which minimizes the task complexity while preserving task fragment dependency. To determine the suitable fog nodes for task fragment service, we formulate the problem as a coalition graph game. Analytical results depict that the proposed scheme minimizes the delay of the task service and the energy consumption while reducing the number of QoS violated tasks by 34.32%, 25.32%, and 22.78% compared to the existing schemes.**

***Index Terms*—Fog computing, task offloading, coalition game, decision-making, resource-utilization, Internet of Vehicles**

I. Introduction

Fog computing involves multiple fog nodes to which resource-constrained devices offload tasks, and the fog nodes (FNs) process the tasks collaboratively [1]. The distributed processing and latency-awareness of fog computing are beneficial for Internet of Things (IoT) networks because IoT devices have limited resources, and the majority of the IoT tasks are latency-sensitive [2]. Therefore, the potential of fog computing is widely acknowledged for different types of IoT networks, including IoV [3]. However, enabling fog computing in the IoV environment poses an additional challenge due to the mobility of vehicles [4]. The vehicles in the IoV demand an uninterrupted service of their tasks even during their movement. To achieve uninterrupted service, the vehicles try to serve the tasks themselves. However, the limitation of resources restrains them from doing so. Consequently, the vehicles offload the tasks to the suitable fog nodes having sufficient resources to handle the tasks. The selection of suitable fog nodes for task offloading should adhere to the heterogeneous QoS demands of the tasks and the movement trajectory of the offloading vehicles [5]. Moreover, the computationally intensive tasks lay the probability to fragment the tasks so that the QoS demands of the tasks can be fulfilled [6]. However, the dependent task fragments should be processed in an ordered manner.

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

The increasing task demands of the users and the varying requirements of each task overwhelm the FNs. The resource limitation of FNs requires the utilization of resources in the most efficient way possible. The problem becomes challenging in the IoV, where the mobile vehicles tend to offload their tasks while expecting a high-quality service for their tasks. Since the base stations are fixed, the mobility of vehicles becomes a hurdle in fulfilling the expected service. Due to mobility, the vehicles move out of the range of the base stations, which results in impeded service. To overcome this problem, the researchers [7], [8] proposed the idea of utilizing the resources of other vehicles that are capable of serving the offloaded tasks. However, in a vehicular network, it is uncertain that a vehicle will be available and is willing to serve the task of others. Also, the decision of serving the task depends on various factors such as the availability of the service, the feature requirement of the task, the processing capability of the serving vehicle, and the cost charged for the service.

The varying feature requirements of the task pose another challenge in addition to the serving capability. Literature [9], [10] discusses the works that mainly focus on the various feature requirement of the tasks. The proposed solution describes the method of offloading each task to a single serving entity. However, the resource limitation raises the question of the availability of an entity that can fulfill all requirements. Federated service is a solution to the mentioned problem where the serving entities can collaborate to serve a task that is presented in various prior works [11], [12]. This improves the QoS of the task by minimizing the delay while effectively utilizing the fog resources. However, the computation intensiveness of the tasks becomes a problem for the entities handling the offloaded task. The solution can be to split the task into fragments and then offload the task fragments to the federation of FNs. But, this raises another issue if the fragmented tasks are dependent on each other.

The dependency of the task fragments becomes a challenge when offloading the fragments to different entities. The out-of-order processing of task fragments produces erroneous results, which in turn results in unsatisfied user service. The solution can be offloading the dependent chunks to a single fog node [13], [7], which restricts the opportunity of utilizing other available resources.

The aforementioned issues raise the following research questions —

- How to ensure the availability of the offloaded task service when both device and the FN are mobile?
- How to ensure the efficient offloading of dependent tasks while effectively utilizing the available resources?

For Personal Use Only

In this work, we propose a game-theoretic scheme, named FedServ, to assign task fragments to FNs considering the heterogeneous latency demands of the tasks. The primary *contributions* of our work are as follows:

- We formulate an optimization problem for vehicle task service while optimizing the delay and effectively utilizing the resources.
- We model a coalition graph game to assign task fragments to mobile FNs considering latency requirements of the task fragments, available processing capability, and the residual energy of the mobile FNs.
- We present an algorithm for the service of the task fragments by a suitable entity, namely, mobile or static FN while minimizing the task service delay.

II. Related Work

A. Offloading in vehicular networks

Tan and Hu [14] presented a caching and computational offloading scheme considering the mobility of vehicles and delay of tasks. A mixed integer linear programming optimization problem is presented by Tran and Pompili [15] for task offloading and resource allocation. They aimed to optimize the task completion time and energy consumption. An coalition game-based task offloading algorithm is proposed by Tiwari *et al.* in an IoV environment considering best-effort and time-sensitive tasks. Peng *et al.* [9] proposed a double auction game between the vehicles requesting the task services and the vehicles offering the services. They mainly focused on the various attributes related to task service so that a fair resource allocation could be achieved. To minimize the response time of the tasks, Tang *et al.* [16] proposed a three-layer vehicular fog architecture for addressing the issue. They aimed at utilizing the resources of vehicles by designing a resource allocation and task scheduling strategy. Huang *et al.* [8] proposed an offloading scheme in vehicular networks considering the computation-intensive and time-sensitive tasks. Nonetheless, the aforementioned works did not consider the possibility of fragmenting the tasks to minimize the task complexity and ignored the dependency of task fragments.

B. Federated Task Service

Towards the service of the task by forming the federation of FNs, we discuss the works [11], [12] in the state-of-the-art. Al-khafajiy *et al.* [11] proposed a fog collaboration framework where the tasks are offloaded to the FNs based on the load and the processing capabilities. Further, towards the federated service, Tiwari *et al.* [12] proposed a method where the task is offloaded to a group of FNs for its service while minimizing the delay. Notably, the works did not consider the provision the fragmenting the task, which minimizes the complexity of the task. Also, if fragmented, the dependency of the task fragments becomes another concern.

C. Offloading dependent tasks

To handle the dependency in the tasks, various works [17], [13], [10] are presented in the literature. Intending to minimize the latency of tasks, Kao *et al.* [17] formulated an NP-hard problem for computation offloading. Sundar *et al.* [13] studied the problem of scheduling the task-dependent applications for minimizing their execution cost. They formulated an NP-hard problem and provided a heuristic-based solution. However, the problem was studied in the cloud computing environment, where the challenges introduced due to mobility were ignored. Similarly, Zhao *et al.* [10] presented the problem of offloading the dependent tasks in mobile edge computing. They provided a convex programming-based solution for their formulated problem. The work again did not consider the mobility of IoT devices and focused on service caching of the edge servers.

Coalition games have been widely used in literature for resource management and optimizing resource allocation [18], [19]. Rajendran and Duraisamy [18] proposed a coalition formation game for efficient spectrum utilization. They used an incentive-based mechanism to motivate the secondary users to form a coalition and detect false alarms. Maity et al. [19] presented a coalition graph game to group and schedule the migration of flows in software defined networks (SDN) to address their QoS demands. The aforementioned works show that the coalition games are effective in proper resource allocation while achieving the task objectives.

Synthesis: The study of existing works reveals that the works consider mainly the time-sensitiveness and the computation-intensiveness of the tasks while overlooking the dependencies in the tasks fragments and the possibility of fragmenting the tasks. Further, it is important to consider the task requirements along with the resource availability of the vehicles. On the contrary, the works considering the dependency of tasks do not consider the mobility of IoT devices and the probability of FNs being mobile. In our previous work [20], we considered full tasks with no fragmentation focusing on minimization of latency. Different from it, in this work, we propose a task offloading scheme in a vehicular network by considering the mobility of the devices, the possibility of fragmenting the tasks for reducing the computation intensiveness while efficiently utilizing the resources of vehicles.

III. System Model

We consider a vehicular network architecture consisting of mobile vehicles, static FNs, mobile FNs, and the cloud as depicted in Figure 1. Let $V = \{v_1, v_2, \ldots, v_N\}$ denote the set of mobile vehicles that request to offload their tasks for service. In the considered IoV environment, the vehicle task request comprises real-time applications such as augmented reality, audio, and video requests, remote monitoring of homes and offices, home automation features, and location services [10]. We consider roadside units (RSU) as static FNs. Vehicles periodically send status data to the RSUs. Therefore, RSUs have updated information of the road network, such as traffic in the area, type of traffic, and the type of vehicles. Further, the vehicles with high processing capability serve as FNs to offer their services. We represent the set of mobile FNs as $V^f = \{v_1^f, v_2^f, \ldots, v_M^f\}$, where $V^f \subseteq V$. The vehicles submit the task request to the RSU for the service. The RSU can either serve the task by itself or offload it to either mobile

For Personal Use Only

TABLE I: Summary of Existing Works

Work	Energy	Delay	Mobility	Collaborative service	Computation-intensiveness	Task fragment dependency
Tan and Hu [14]	✓	✗	✓	✗	✗	✗
Al-Khafajiy et al. [11]	✗	✓	✗	✓	✗	✗
Abkenar et al. [21], Tiwari et al. [12]	✓	✓	✗	✓	✗	✗
Tang et al. [16]	✗	✓	✗	✗	✗	✗
Sundar et al. [13]	✗	✓	✗	✓	✗	✓
FedServ	✓	✓	✓	✓	✓	✓

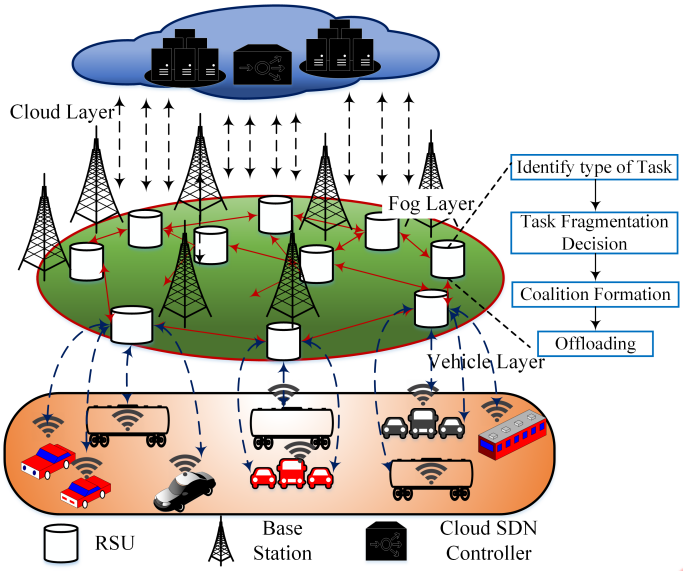


Fig. 1: System Model Architecture

FNs or the cloud. Following the software-defined network (SDN) architecture for supporting the distributed virtualized fog platform, we consider the SDN controller placed in the cloud, which keeps the global information of all the RSU controllers along with the mobile FNs and other vehicles. RSUs communicate with the cloud controller to update the information of the vehicles in its vicinity and forward the task request to the cloud if it cannot be served by itself or any FN.

It is noteworthy that the vehicles are represented by the tuple set $(vid, speed, direction)$, which consists of the unique identity given to each vehicle, the speed of the vehicle, and the direction of its mobility. Moreover, the association of vehicles with the RSU changes with their movement. Thus, RSUs update the information of vehicles entering and leaving their vicinity.

A. Fog Module Structure

We consider two types of FNs in our architecture — (1) static FNs, namely, RSU, and (2) mobile FNs.

(a) Static Fog Nodes: RSUs act as static FNs with storage and processing capability. Each RSU is associated with a local SDN controller that keeps itself updated with the current information of the road traffic. It also keeps the details of each vehicle, such as processing capability, type, location, and currently available resources. RSUs are responsible for deciding where to serve the task of the vehicle — either by itself or by mobile FN. If the decision is to offload the task to mobile FN, it should find out suitable FNs for offloading.

(b) Mobile Fog Nodes: Since the RSU is loaded with its responsibilities of the road network and to fulfill the QoS requirement of the tasks, it tries to offload the tasks of vehicles to other entities. We term the vehicles with onboard processing capabilities and are willing to serve the task of other vehicles as the mobile FNs. As mentioned, RSU contains information regarding all the vehicles. Therefore, if any vehicle is willing to serve the offloaded task, the RSU offloads the task. However, the decision of offloading and serving depends on several factors, including processing units and energy required for the service of the task.

B. Task Model

We consider two types of vehicle tasks in our system model — (a) tasks that cannot be partitioned and (b) those that can be fragmented. A task that cannot be partitioned has a single task fragment, and it is served by a single entity entirely that is either it can be fully offloaded, or it can be fully served by itself. On the other hand, a task that can be partitioned consists of multiple task fragments, and each of the fragments is offloaded to an entity based on the requirement of the fragment. This speeds up the processing of the task. However, the challenge is to handle the dependent fragments offloaded to different entities. Let $T = \{t_1, t_2, \ldots, t_\tau\}$ represent the set of tasks. The set of fragments of the i^{th} task is represented as $t_i^{frag} = \{t_i^1, t_i^2, \ldots, t_i^{n_i}\}$.

For each of the task in T, there is a maximum allowable delay which the task can undergo. Additionally, if the task is fragmented, each of the fragments has its own maximum allowable delay. Also, if the dependent task fragments t_i^l and t_i^k have the maximum allowable delay as $\delta_{i,l}^{max}$ and $\delta_{i,k}^{max}$, for $l < k$, then their respective allowable delay should also be $\delta_{i,l}^{max} < \delta_{i,k}^{max}$.

Definition 1. *The two task fragments are said to be dependent when the input of one task fragment t_i^k depends on the output of the task fragment t_i^l.*

C. Delay Model

The delay incurred by a task fragment for its service includes the uploading delay, the processing delay, and the result downloading delay. In addition to this, if the task is offloaded to another entity, an additional migration delay is added to the total delay.

The uploading delay includes the transmission delay and the propagation delay of the task. Mathematically, the transmission delay is defined as, $\delta_{i,l}^{tr} = \frac{\chi_{i,l}}{r_j}$, where $\chi_{i,l}$ is the size

For Personal Use Only

of the l^{th} task fragment and r_j is the transmission rate. We considered a M/M/1 queue at the mobile fog nodes for the service of the task. Due to the limited computational resource at the MFN, we consider the M/M/1 queue which allows only one task to be served at a time, however, motivated by the works in the literature [3], [22], [23], for simplicity, we have used an infinite buffer at the queue. The queuing delay for serving the task is modelled as M/M/1 queuing model [24], defined as, $\delta^q_{i,l} = \frac{1}{s_j - \lambda_{i,l}}$, where s_j is the service rate and $\lambda_{i,l}$ is the arrival rate of the task fragment. Further, the processing delay of the task is computed based on the computation cycles required for the task fragment and the processor's CPU frequency, $\delta^{proc}_{i,l} = \frac{P_{i,l}}{f_j}$, where $P_{i,l}$ is the amount of processing units required for l^{th} and f_j is the CPU frequency of the j^{th} node.

Therefore, the total delay incurred by a task fragment for its service includes the transmission delay ($\delta^{tr}_{i,l}$) for uploading the task, propagation delay ($\delta^{prop}_{i,l}$), queuing delay ($\delta^q_{i,l}$), processing delay ($\delta^{proc}_{i,l}$), and the transmission delay of downloading the result, which is defined as,

$$\delta^{tot}_{i,l} = \delta^{tr}_{i,l} + \delta^{prop}_{i,l} + \delta^q_{i,l} + \delta^{proc}_{i,l} + \delta^{tr}_{i,l} \qquad \forall t_i \in \mathcal{T}, t^l_i \in t^{frag}_i \tag{1}$$

Further, the total task service delay incurred by task t_i is given by,

$$\delta^{tot}_i = \sum_{t^l_i \in t^{frag}_i} (\delta^{tot}_{i,l}) \tag{2}$$

D. Energy Model

The processing of any task requires the serving entity to activate its processing unit for the service. This activation of the processing unit requires the energy consumption of the node. Let the activation of one processing unit requires f CPU cycles. If $P_{i,l}$ is the number of processing units required by any task fragment t^l_i then the activation of processing units for the computation of the task fragment t^l_i requires $fP_{i,l}$ CPU cycles. If ϵ units of energy are required to activate one processing unit, then the energy required to activate the processing units for the execution of the task fragment t^l_i is given by,

$$E_{i,l} = \epsilon f P_{i,l} \tag{3}$$

E. Problem Formulation

The vehicles that need a task service request the RSU. RSU tries to offload the task to any other entity willing to serve the task. Moreover, the association of the vehicles with the RSU changes with the speed and the direction of their movement. Further, the task can be of any type as described in Section III-B, and therefore, the decision of offloading is also dependent on the nature of the task. If the task is not splittable, then the entire fragment should either be served or offloaded as a whole. Thus, it needs to find a capable and willing mobile FN to handle the task while fulfilling its requirements. However, if the task is splittable, it is fragmented, and each fragment can be offloaded to an entity. In addition, while offloading the task fragment to any FN, the RSU aims to minimize the delay of the task processing.

Moreover, the task fragments are offloaded according to their requirement for which either a single FN can serve the task or multiple FNs are required to serve the task. To preserve the dependency among the task fragments while minimizing the delay of processing, we need to group the tasks according to their dependency and delay requirements as shown in Figure 2. Thus, we formulate the problem of which task is to be served by which FN and by how many FNs as a coalition graph game. We consider that each RSU executes the coalition graph game where each coalition signifies a mobile FN that is present within the communication range of the RSU and will remain so for a predefined duration of α units. For each RSU, we term these mobile FNs as candidate mobile FNs. The candidate mobile FNs are estimated based on the vehicle mobility data available at each RSU. Additionally, the value of α is selected as the maximum value of the maximum allowable delay among all tasks reaching the RSU. The players of the coalition graph game are the individual task fragments of the tasks received by the corresponding RSU.

For each of the candidate mobile FN i, a coalition c_i will be formed. Let $\mathcal{C} = \{c_1, c_2, \ldots, c_L\}$ denote the coalition structure which is the set of coalitions. The members of the coalition define the task fragments to be executed by a particular FN. After the formation of a stable coalition structure, the corresponding RSU offloads the task fragments of each coalition c_k to the respective mobile FN v^f_k.

An RSU may receive multiple task fragment requests either from a vehicle or from multiple vehicles. As mentioned, each task fragment t^l_i will have its own maximum allowable delay ($\delta^{max}_{i,l}$) for its service. The players of the coalition game aim to maximize their utility by forming the coalition. We define the utility of any task fragment t^l_i for coalition c_k as:

$$U^l_i = N^l_i \left(\frac{\delta^{max}_{i,l} - \delta^{tot}_{i,l}}{\delta^{max}_{i,l}} + \frac{P^{res}_k - P_{i,l}}{P^{res}_k} + \frac{E^{res}_k - E_{i,l}}{E^{res}_k} \right), \tag{4}$$

where N^l_i is the number of task fragments in the coalition for task t_i or the dependent task fragments, P^{res}_k denote the residual processing units of mobile FN v^f_k, E^{res}_k denote the residual energy of mobile FN v^f_k, $P_{i,l}$ is the number of processing units required for the execution of task fragment t^l_i, and $E_{i,l}$ is the energy required for the activation of the processing units required for the execution of task fragment t^l_i. The utility of the task fragment defines the satisfaction of the user achieved from the service of the task, which depends on the delay in service.

Consequently, the utility of a coalition c_k is the total utility of the members. We express the utility of coalition c_k as,

$$U_k = \sum_{t_i \in T} \sum_{t^l_i \in t_i} (U^l_i) \tag{5}$$

The association between a task fragment t^l_i and a mobile FN v^f_k is expressed as,

$$z(i,l,k) = \begin{cases} 1 & \text{if } t^l_i \text{ is served by } v^f_k, \\ 0 & \text{otherwise.} \end{cases} \tag{6}$$

For Personal Use Only

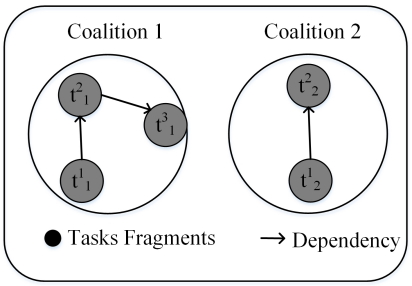


Fig. 2: Coalition Formation

F. Objective

The primary objective of this work is to optimally obtain the maximum utility of task fragments which maximizes the QoS of the tasks. Further, the objective is to choose an FN or multiple whose service will maximize this utility. Mathematically, the objective function is denoted as,

$$\max_{\mathcal{C}}(U_k) \qquad \forall c_k \in \mathcal{C} \tag{7}$$

subject to

$$\delta_i^{tot} \leq \delta_i^{max} \qquad \forall t_i \in \mathcal{T}$$

$$\delta_{i,l}^{tot} \leq \delta_{i,l}^{max} \qquad \forall t_i^l \in t_i^{frag}, t_i \in \mathcal{T}$$

$$\sum_{v_k^f \in V^f} z(i,l,k) = 1, \forall t_i^l \in t_i^{frag}, t_i \in \mathcal{T}$$

$$\sum_{t_i \in T} z(i,l,k) P_{i,l} \leq P_k^{res}, \forall v_k^f \in V^f$$

$$\sum_{t_i \in T} z(i,l,k) E_{i,l} \leq E_k^{res}, \forall v_k^f \in V^f$$

The first two constraints express that the total task service delay should not exceed the maximum allowable delay for each task and each task fragment. The third constraint states that each task fragment is assigned to a single mobile FN. The final two constraints express the processing capacity constraint and energy consumption constraint for each mobile FN.

IV. FedServ: The Proposed Scheme

This section presents the solution approach to the coalition graph game. As mentioned, each of the task fragments forms the coalition, and the RSU is responsible for finding the candidate mobile FNs for processing the task fragments. It is noteworthy that the FNs chosen for the service of the task of other vehicles should remain in the vicinity of the RSU where the request is received. Moreover, the RSU controllers are aware of the vehicle's mobility information. Thus, when finding the candidate FNs, the RSUs only consider the FNs whose duration of staying in the vicinity of this RSU is at least α units.

For each RSU, we construct the coalition graph $G(\mathcal{T}, \mathcal{D})$ with the task fragments received by the RSU $\mathcal{T}$ as the vertices and the edges $\mathcal{D}$ between the vertices represents the fragment dependency on each other. The task fragments t_i^m have an edge to t_i^n if the processing of task t_i^n depends on the output of the task t_i^m. Nevertheless, the allowable delay of task t_i^m is less than t_i^n. Notably, the task with minimum delay requirement forms the coalition head; that is, the more latency-sensitive task is the coalition head of the coalition.

The formation of coalitions is guided by several rules of coalitions games, namely, merge rule, preference rule, and split rule. We present the definition of these rules as,

Definition 2. *The preference order defines the sequence in which one partition is preferred over the other. Suppose P_1 and P_2 are two partitions of the set $\mathcal{T}$ then the preference order denoted as $P_1 \prec P_2$ represents that P_1 is preferred over P_2.*

Definition 3. *The merge rule says that any set of coalitions is merged into one coalition if the players of the coalitions prefer the merging. If $\{c_1, c_2, \ldots, c_z\}$ represents the coalition set that can be merged then its merged form is represented as $\bigcup_{k=1}^{z} c_k$.*

Definition 4. *The split rule says if the players of a coalition prefer the split of it, the coalition $\bigcup_{k=1}^{z} c_k$ is split. If $\{c_1, c_2, \ldots, c_z\} \prec \bigcup_{k=1}^{z} c_k$, then according to split rule, $\bigcup_{k=1}^{z} c_k \longrightarrow \{c_1, c_2, \ldots, c_z\}$.*

We present the algorithm for the formation of the coalition graph. The algorithm takes the task fragments received by the corresponding RSU as the input and outputs the coalition structure formed, and the edges between the dependent task fragments. If the task request received can be fragmented, it is partitioned into fragments. The task fragment is added to the coalition c_p and becomes the coalition head c_p^{lead}. The coalitions in $\mathcal{C}$ are arranged in the descending order of maximum allowable delay of c_j^{lead}. The coalitions undergo merging and splitting operations to form new coalitions. If task fragments t_i^l and t_i^k belong to the coalition c_p and $\delta_{i,l}^{max}$ is greater than $\delta_{i,k}^{max}$ then there is an edge between t_i^l and t_i^k which is stored in $\mathcal{D}$. The coalition head is the task fragment with the least δ^{max}. The coalitions are arranged in descending order of δ^{max} and are added to $\mathcal{C}$ accordingly. The algorithm then outputs the $\mathcal{D}$ and $\mathcal{C}$. The steps of the algorithm are presented in 1.

Definition 5. *A Defection function $\mathcal{D}$ defines the stability of any coalition, i.e., whether any participant of a partition can be benefited by leaving or joining other partitions.*

Theorem 1. *The coalition structure $\mathcal{C}$ formed from the Algorithm 1 is $\mathcal{D}_{hp}$ stable.*

Proof. Let $\mathcal{C}^*$ be the final coalition structure resulting from the Algorithm 1 which repeatedly applies merge and split rules. A coalition $c_i \in \mathcal{C}^*$ cannot be split or merged further [25] in successive iterations. In other words, no player can increase its utility by leaving the $\mathcal{C}^*$ and forming a new coalition structure. Therefore, the resulting coalition structure from Algorithm 1 is in its final form or $\mathcal{D}_{hp}$ stable. □

Theorem 2. *The coalition structure $\mathcal{C}$ resulting from the Algorithm 1 is $\mathcal{D}_c$ stable satisfying the necessary conditions,*

- *For any two coalitions q_1 and q_2 with $q_1 \cap q_2 = \phi$, $\{q_1 \cup q_2\} \subset c_i$, $c_i \in \mathcal{C}$, the utility of union of q_1 and q_2 is greater than the summation of their individual utility,*

For Personal Use Only

Algorithm 1: Coalition graph formation

Inputs: $\mathcal{T}$.
Outputs: D, $\mathcal{C}$.
for $t_i \in T$ **do**
  for $t_i^l \in t_i$ **do**
    if $t_i^l \in \mathcal{T}$ **then**
      $c_p \longleftarrow c_p \cup t_{i,p}^l$
      $c_p^{lead} \longleftarrow t_{i,p}^l$
      $\mathcal{C} \longleftarrow \mathcal{C} \cup \{c_p\}$
$\mathcal{C}^{init} \longleftarrow \emptyset$
while $\mathcal{C}^{init} \neq \mathcal{C}$ **do**
  orderedList $\longleftarrow$ Arrange $\mathcal{C}$ in non-increasing order of the maximum allowable delay of the coalition heads
  $\mathcal{C} \longleftarrow$ orderedList
  $\mathcal{C}^{init} \longleftarrow \mathcal{C}$
  for $c_p \in \mathcal{C}$ **do**
    Apply merge rule or split rule according to the Definitions 2 and 3
    if $t_{i,p}^l \in c_p$ *and* $t_{i,p}^k \in c_p$ **then**
      if $\delta_{i,l}^{max} > \delta_{i,k}^{max}$ **then**
        $\mathcal{D} \longleftarrow \mathcal{D} \cup (t_{i,p}^l, t_{i,p}^k)$
    c_p^{lead} = min $\{\delta_{i,l}^{max}\}$;
    Arrange c_p in non increasing order of δ^{max}
  Update $\mathcal{C}$
return $\mathcal{D}, \mathcal{C}$

$U_{1\cup2}^q > U_1^q + U_2^q$, *where* $U_{1\cup2}^q$ *denotes the combined utility of* $\{q_1 \cup q_2\}$, U_1^q *is the utility of coalition* q_1, *and* U_2^q *is the utility of coalition* q_2.

- *For the coalition structure* $\mathcal{C}$, *coalition* $c^{'} \subset \mathcal{T}$ *whose players belong to different coalition* $c_i \in \mathcal{C}$ *is termed as* $\mathcal{C}-$*incompatible. For strict* $\mathcal{D}_{hp}$ *stability,* $\sum_{c_i \in \mathcal{C}} U_i^{''} > U^{'}$, *where* $U_i^{''}$ *and* $U^{'}$ *denote the utilities of coalitions* $\{c_i \cap c^{'}\}$ *and* $c^{'}$, *respectively.*

Proof. Let q_1 and q_2 be two disjoint coalitions such that $\{q_1 \cup q_2\} \subset c_i$ where $c_i \in \mathcal{C}$. The players included in a coalition c_i signifies the task fragments to-be-assigned to the mobile FN v_i^f. Therefore, the task fragments included in q_1 and q_2 are assigned to v_i^f. However, according to Equation (4), the utility of a player increases with the number of dependent task fragments in the coalition. Therefore, $U_{1\cup2}^q > U_1^q + U_2^q$ because the number of dependent task fragments of each player in the combined coalition is more than the individual ones.

For each $\mathcal{C}-$incompatible coalition $c^{'}$, the condition $\sum_{c_i \in \mathcal{C}} U_i^{''} > U^{'}$ satisfies if $U^{'} = 0$. The players in $c^{'}$ belongs to different coalitions $c_i \in \mathcal{C}$. Therefore, each task fragment in $c^{'}$ are assigned to different mobile FNs. Consequently, each task fragment in $c^{'}$ has no dependent task fragments and the $\mathcal{C}-$incompatible coalition $c^{'}$ has utility $U^{'} = 0$.

Hence, the coalition structure $\mathcal{C}$ resulting from the Algorithm 1 is $\mathcal{D}_c$ stable. □

The task service of any vehicle involves various interactions between the entities. The objective of each is to minimize the delay while maximizing their utility. The vehicles requesting the task service submit the tasks t_i to the RSUs. To efficiently utilize the resources, the RSU that receives the task request tries to offload the tasks to any mobile FN willing to offer its services. However, since the vehicles are limited in their resources and the task sizes are computationally high, it partitions the task into fragments. These fragments form coalitions based on their dependency to maximize their utility using Algorithm 1. For each of the coalition c_p, RSU then finds the suitable candidate mobile FNs to process these tasks. It is noteworthy that the RSU considers a mobile FN v_j^f for coalition formation only if v_j^f remains within the communication range of the RSU for a duration of at least α units. Let β_j^f denote the duration for which v_j^f remains within the communication range of the RSU. Therefore, for coalition formation an RSU considers only those mobile FNs for which $\beta_j^f \geq \alpha$. Further, an FN should also have sufficient resources to serve the task of the coalition. Nevertheless, RSU is updated with information such as mobility and current resources of vehicles in its vicinity. However, if RSU r is unable to find any capable FN to serve the tasks of the coalition, it checks the availability of resource $\mathcal{R}_r^{avl}$. If $\mathcal{R}_r^{avl} > \mathcal{R}_{i,l}^{req}$, then the RSU serve c_p by itself. Otherwise, it forwards the entire task to the cloud. We present the steps of task processing in Algorithm 2.

Algorithm 2: Task Service

Inputs: T, $\mathcal{C}$.
Output: $\mathcal{A}$.
Initialize $\mathcal{A} \longleftarrow \emptyset$.
for $t_i \in T$ **do**
  if *task* t_i *can be fragmented* **then**
    Fragment t_i
  Call Algorithm 1 to generate $\mathcal{C}$
  for $c_p \in \mathcal{C}$ **do**
    for $v_j^f \in V^f$ *and* v_j^f *in vicinity of RSU* **do**
      if $\beta_j^f \geq \alpha$ **then**
        if $\mathcal{R}_j^{avl} > \mathcal{R}_{i,l}^{req}$ **then**
          Assign c_p to v_j^f
          $\mathcal{A} = \mathcal{A} \cup (c_p, v_j^f)$
    if *no* v_j^f *in vicinity of RSU* **then**
      if $\mathcal{R}_r^{avl} > \mathcal{R}_{i,l}^{req}$ **then**
        Serve the task fragments in c_i by the RSU
      else
        Forward the task t_i to the cloud
return $\mathcal{A}$

We analyze the time complexity of the proposed algorithm for the task service of any vehicle. It involves the process of formation of coalitions of the task fragments by the Algorithm

For Personal Use Only

1 and the selection of suitable mobile FNs for serving each coalition. It is noteworthy that the complexity of coalition formation depends on the number of merge and split operations called during the process. Therefore, for the τ number of tasks, a maximum of T coalitions can be formed. In the uttermost, a coalition tries to merge with all other coalitions. In such a case, the first coalition tries to attempt the $T-1$ number of the merge operations. Similarly, the second coalition can attempt $T-2$ number of merge attempts and so on. Thus, the number of merge operations at most is $\frac{|T|(|T|-1)}{2}$. However, in a real-time scenario, each coalition merges with the suitable candidate coalitions where the candidates are those which have an edge between the task fragments of two coalitions. Hence the number of merge operations is less than the maximum number. Further, the maximum number of split operations tends to find all the partitions, which can be computed using the Bell number [26]. Essentially, the number of partitions depends on the number of players in coalition games and increases exponentially as the number of players rises. But, in reality, the coalition split according to the preference rule 2 and does not attempt to split further. Therefore, the number of split operations is also significantly less in practical scenarios.

V. Performance Evaluation

In this section, we assess the performance of our proposed system. We discuss the simulation settings, performance metrics and the benchmark schemes used for comparing the performance.

TABLE II: Simulation Parameters

Parameters	**Values**
Number of RSUs	30-100
Number of tasks	20-500
Network topology	Barabasi-Albert [27]
Mobility model	Gauss Markov
Vehicle CPU frequency	1-3 GHz
Fog Node CPU frequency	2.9-4.2 GHz [15]
Data transmission rate	20-100 Mbps
Amount of computation of task	1500-2500 Megacycle [15]
Average task size	450 KB [15]
Transmission power of fog node	20 dBm [15]
System bandwidth	20 MHz [15]
Velocity of vehicles	2–20 m/s
Maximum allowable delay (T_1)	5–50 ms
Maximum allowable delay (T_2)	51–100 ms
Maximum allowable delay (T_3)	101–200 ms

A. Simulation settings

We evaluate the performance of FedServ through simulation on the MATLAB platform. We use the road traffic data of the M11 motorway in Ireland [28] of March 2021. To ensure that the performance of the system does not change if the network size changes, we consider the network topology of RSUs following the scale-free Barabasi-Albert [27] topology. The Gauss Markov mobility model is followed by the vehicles for their movement in the network. We consider three types of tasks with different delay requirements. The simulation parameters and their values are presented in Table II. All the results are plotted using the confidence interval of 95% by running each of the experiments for 100 iterations.

B. Benchmark Schemes

To analyze the performance of FedServ, we compare it with an existing scheme – heuristic-based offloading [8] and online secretary-based offloading [12], named *HBO* and *SBO*, which adopt a heuristic algorithm and online secretary algorithm, respectively, to find out the candidate vehicles for offloading the task. In addition, we compare the performance of the proposed scheme with two traditional schemes — *Random* and *Nearest*. In Random, RSU chooses a vehicle in its vicinity randomly, while in Nearest, a vehicle with minimum distance from the RSU is chosen as the candidate to offload the task. Random and Nearest pave the motivation for the design of the proposed scheme over the traditional methods, while HBO shows the efficiency of the proposed scheme over the heuristic method by considering the delay of the task and resource availability of any vehicle. SBO, on the other hand, evaluates the performance of FedServ compared to the online algorithm for candidate selection for task offloading.

C. Performance Metrics

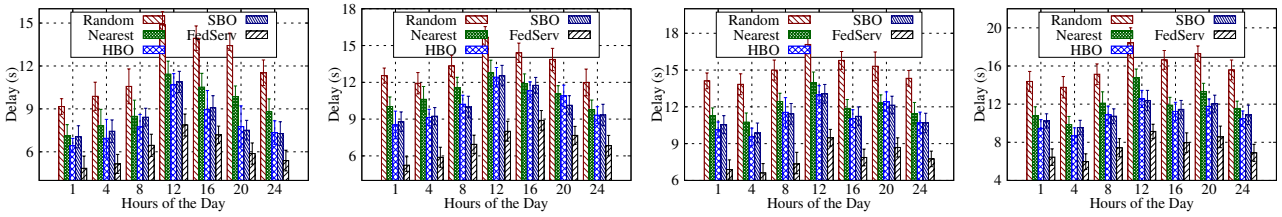
We evaluate the performance of the aforementioned benchmark schemes compared with FedServ with respect to the metrics stated below,

- **Service Delay**: It is defined as the total delay computed using Equation 2 incurred in the task service of the vehicle by the RSU either by offloading or by serving by itself. The delay metric defines the efficiency of FedServ for latency-sensitive applications.
- **Energy Consumption**: The energy consumed by the vehicles to activate their processing units for the execution of the task computed using Equation 3 is defined as the energy consumption. Since the proposed method guarantees the effective utilization of the resources, energy consumption defines its resource utilization efficiency.
- **QoS Violation of Tasks**: It measures the deviation of the QoS achieved by the task service from its expected QoS. We compute it by evaluating the difference between the total incurred delay from the expected delay of the tasks. This metric paves the way for showing the user satisfaction achieved by FedServ in comparison to the other systems.
- **Cost**: It measures the cost of activating the processing units required for the service of the task. This metric assists in comparing the performance of the schemes in terms of cost incurred for the task service.

D. Results

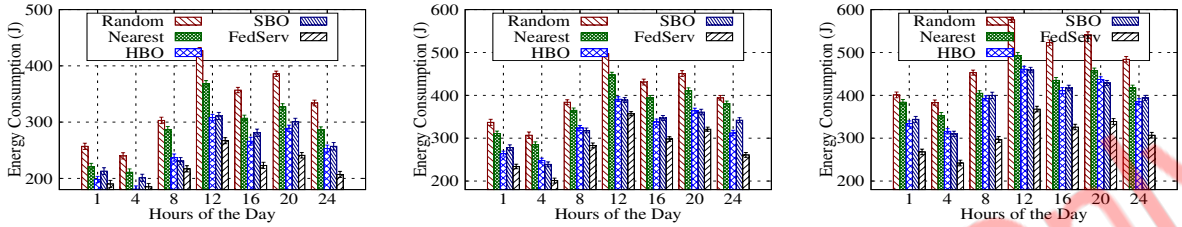
1) Service Delay: We compare the delay incurred in the service of the task by all the schemes. The performance of FedServ in comparison to the benchmark schemes is shown in Figure 3. It is evident from the figure that the delay is maximum at the peak time of the days when the traffic is maximum due to more number of task requests. Also, the delay increases with the increase in the amount of computation of the tasks as shown in Figure 3(a), Figure 3(b), and Figure 3(c). Also, Random incurs the highest service delay while FedServ

For Personal Use Only



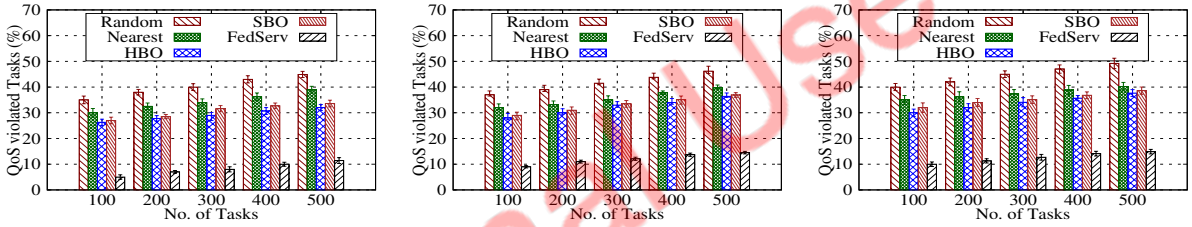
(a) p_i = 1500 Megacycles (b) p_i = 2000 Megacycles (c) p_i = 2500 Megacycles (d) Avg speed = $10m/s$

Fig. 3: Service Delay



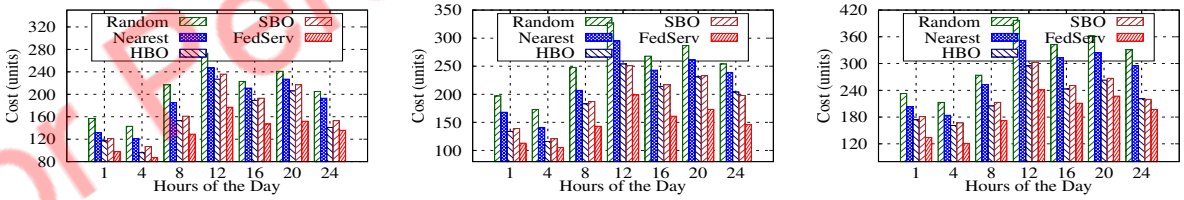
(a) p_i = 1500 Megacycles (b) p_i = 2000 Megacycles (c) p_i = 2500 Megacycles

Fig. 4: Energy Consumption



(a) p_i = 1500 Megacycles (b) p_i = 2000 Megacycles (c) p_i = 2500 Megacycles

Fig. 5: Qos violated tasks



(a) p_i = 1500 Megacycles (b) p_i = 2000 Megacycles (c) p_i = 2500 Megacycles

Fig. 6: Cost

For Personal Use Only

performs best. Random chooses a vehicle for offloading a task randomly without considering the computation power of the vehicle and its mobility time. Similarly, Nearest chooses a vehicle whose distance from the RSU is minimum, ignoring computation power and the mobility time. On the other hand, HBO considers the computation power, however, the vehicles tend to move out of the RSU's vicinity, which has offloaded the task leading to the increased delay of the task service. SBO increases delay for its computation in the algorithm. FedServ benefits with the minimal delay by partitioning the task while considering the delay requirement of the task, computation power of the vehicle, and the mobility time of the vehicle.

We compared the service delay of the tasks by keeping the average speed of vehicles as $10m/sec$ in Fig. 3(d). The figure shows that the service delay increased with the increase in the speed of the vehicles. This is because, with the high speed of vehicles, it is difficult to find mobile FNs for offloading the tasks. Again, Random, Nearest, and Heuristic incur high delay because of their inconsideration of the mobility time of vehicles due to which the chosen candidate for offloading

task moves out of the RSUs vicinity and thus increase the delay of task service. FedServ, on the other hand, induces lesser delay compared to benchmark schemes but high delay when compared with the varying speed. This happens due to the difficulty of finding a suitable candidate for offloading the task.

2) Energy Consumption: We analyze the energy consumption in the system for the service of the task submitted at the RSU, shown in Figure 4. The figure shows that the energy consumption of the systems is maximum at the peak time of the day when the traffic is high. Further, the energy consumption is higher with the increase in the amount of computation of the tasks as depicted from the Figures 4(a), 4(b),and 4(c). The energy consumption for the service of the task depends on the selection of a suitable candidate. Accordingly, Random chooses the FN randomly for its task service, which induces higher energy consumption, while nearest greedily chooses the nearest FN ignoring the computation capability. HBO, on the other hand, consumes higher energy for not considering the movement of vehicles from the vicinity of the RSU. SBO increases energy consumption in executing its algorithm while finding a suitable candidate for task offloading. FedServ outperforms all the schemes by minimizing the energy consumption of the task service by choosing a suitable FN while considering the delay requirement of the task, computation capability, and mobility of the vehicles.

2) QoS Violation of Tasks: We measure the percentage of tasks that violate their QoS requirement. Considering the delay as the QoS requirement of the task, Figure 5 depicts the percentage of QoS violated tasks. The figure depicts that the number of tasks that violate QoS are minimal in FedServ as compared to the other schemes. This is because, in FedServ, the tasks are partitioned, and fragments are assigned to FNs for their service, which minimizes their service delay. However, Random chooses an FN randomly and offloads the entire task, while Nearest chooses the closest FN and offloads the task. Both the schemes ignore the computation capability of the FN before offloading the task. HBO and SBO consider the computation capability of the vehicles before offloading; however, it fails to consider the mobility of vehicles which increases the delay of the task service. Thus, the percentage of QoS violated flows increases with the increase in the number and the amount of computation of the tasks, as shown in Figure 5(a), 5(b), and 5(c).

2) Cost: We measure the cost of activating the processing units required for the task service by all the schemes. Figure 6 presents the comparison of FedServ with the other schemes for the cost incurred in the task service. It is evident from the figure that the cost is maximum when the traffic is high, which is due to more number of task requests. This is expected as more tasks require more processing units to be activated for their service. It is noteworthy that the cost of activating the processing unit is higher at the server compared to the RSU and the vehicles. Also, the computation-intensive tasks require more number of efficient processing units. Thus, Figure 6(a), 6(b), 6(c) show that the cost also increases with the increase in the computation amount of the tasks. Moreover, Random performs the worst by choosing the FN for offloading the task randomly, ignoring the efficiency of the vehicles to serve the task, which compels it to offload more tasks to the server incurring a high cost. Nearest greedily chooses the closes FN to the RSU without considering the computation amount of the task and the resource availability of the vehicle. HBO, on the other hand, considers the computation intensiveness of the task for offloading; however, it tends to offload more tasks to the server because of suitable vehicle availability. SBO increases its cost of task service for its inconsideration of mobility of vehicles during the task offloading and its computation of the algorithms for finding the candidate. FedServ is benefited from the provision of fragmentation of the tasks. Since the tasks are fragmented before offloading to the mobile FNs, the computation amount of tasks is reduced in fragments, and thus, the vehicles with the available resources to process the fragments can be found, thereby reducing the cost. Thus, FedServ outperforms all the benchmark schemes.

VI. Conclusion

In this work, we presented a federated task service scheme, named FedServ, in a fog-enabled vehicular network while considering the different requirements of the tasks in the presence of mobile vehicles. The decision of where to serve the task is taken by considering the availability of mobile and static FNs depending on the requirements of the task. Moreover, to minimize the delay of computation-intensive tasks, we provisioned the method of fragmenting the tasks and offering a federated service to the task fragments. We formulate an optimization problem for maximizing the utility of the task fragments while considering the delay requirement, the energy of the nodes, and the processing units required for the task service. We model the problem as a coalition game for choosing the suitable and optimal number of fog nodes for the task service. The numerical results show that FedServ performs better in terms of QoS requirement of the tasks by minimizing the QoS violation by 34.32%, 25.32%, and 22.78% compared to the benchmark schemes, Random, Nearest, and Heuristic.

In this work, we considered that the vehicles submit their task to the RSU for the service. However, in a real-time environment, a vehicle can request other vehicles also for the task service. Thus, we plan to study the vehicle to vehicle offloading of the tasks as a future extension of this work. Further, we also plan to implement the proposed system in real-time vehicle networks.

Acknowledgements

Some of the preliminary concepts in this work have appeared in the paper entitled “LOAN: Latency-Aware Task Offloading in Association-Free Social Fog-IoV Networks”, Globecom 2021.

References

[1] A. Yousefpour, G. Ishigaki, R. Gour, and J. P. Jue, “On reducing IoT service delay via fog offloading,” *IEEE Internet of Things Journal*, vol. 5, no. 2, pp. 998–1010, 2018.

For Personal Use Only

[2] A. Lakhan, M. Ahmad, M. Bilal, A. Jolfaei, and R. M. Mehmood, "Mobility aware blockchain enabled offloading and scheduling in vehicular fog cloud computing," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 7, pp. 4212–4223, 2021.

[3] Z. Ning, J. Huang, and X. Wang, "Vehicular fog computing: Enabling real-time traffic management for smart cities," *IEEE Wireless Communications*, vol. 26, no. 1, pp. 87–93, 2019.

[4] G. M. N. Ali, M. Noor-A-Rahim, M. A. Rahman, S. K. Samantha, P. H. J. Chong, and Y. L. Guan, "Efficient real-time coding-assisted heterogeneous data access in vehicular networks," *IEEE Internet of Things Journal*, vol. 5, no. 5, pp. 3499–3512, 2018.

[5] P. Mach and Z. Becvar, "Mobile edge computing: A survey on architecture and computation offloading," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 3, pp. 1628–1656, 2017.

[6] A. Bozorgchenani, F. Mashhadi, D. Tarchi, and S. S. Monroy, "Multi-objective computation sharing in energy and delay constrained mobile edge computing environments," *IEEE Transactions on Mobile Computing*, 2020, doi: 10.1109/TMC.2020.2994232.

[7] 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.

[8] J. Huang, Y. Qian, and R. Q. Hu, "A vehicle-assisted data offloading in mobile edge computing enabled vehicular networks," in *in Proc. of IEEE Global Communications Conference (GLOBECOM)*, 2019, pp. 1–6.

[9] X. Peng, K. Ota, and M. Dong, "Multiattribute-based double auction toward resource allocation in vehicular fog computing," *IEEE Internet of Things Journal*, vol. 7, no. 4, pp. 3094–3103, 2020.

[10] G. Zhao, H. Xu, Y. Zhao, C. Qiao, and L. Huang, "Offloading dependent tasks in mobile edge computing with service caching," in *in Proc. of IEEE INFOCOM*, 2020, pp. 1997–2006.

[11] M. Al-Khafajiy, T. Baker, H. Al-Libawy, Z. Maamar, M. Aloqaily, and Y. Jararweh, "Improving fog computing performance via fog-2-fog collaboration," *Future Generation Computer Systems*, vol. 100, pp. 266–280, 2019.

[12] M. Tiwari, S. Misra, P. K. Bishoyi, and L. T. Yang, "Devote: Criticality-Aware Federated Service Provisioning in Fog-Based IoT Environments," *IEEE Internet of Things Journal*, vol. 8, no. 13, pp. 10 631 – 10 638, 2021.

[13] S. Sundar and B. Liang, "Offloading dependent tasks with communication delay and deadline constraint," in *in Proc. of IEEE INFOCOM*, 2018, pp. 37–45.

[14] L. T. Tan and R. Q. Hu, "Mobility-aware edge caching and computing in vehicle networks: A deep reinforcement learning," *IEEE Transactions on Vehicular Technology*, vol. 67, no. 11, pp. 10 190–10 203, 2018.

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

[16] C. Tang, X. Wei, C. Zhu, Y. Wang, and W. Jia, "Mobile vehicles as fog nodes for latency optimization in smart cities," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 9, pp. 9364–9375, 2020.

[17] Y.-H. Kao, B. Krishnamachari, M.-R. Ra, and F. Bai, "Hermes: Latency optimal task assignment for resource-constrained mobile computing," *IEEE Transactions on Mobile Computing*, vol. 16, no. 11, pp. 3056–3069, 2017.

[18] M. Rajendran and M. Duraisamy, "Distributed coalition formation game for enhancing cooperative spectrum sensing in cognitive radio ad hoc networks," *IET Networks*, vol. 9, no. 1, pp. 12–22, 2019.

[19] I. Maity, S. Misra, and C. Mandal, "DART: Data Plane Load Reduction for Traffic Flow Migration in SDN," *IEEE Transactions on Communications*, 2020, doi: 10.1109/TCOMM.2020.3042271.

[20] M. Tiwari, I. Maity, and S. Misra, "Loan: Latency-aware task offloading in association-free social fog-iov networks," in *IEEE Global Communications Conference (GLOBECOM)*, 2021, pp. 01–06.

[21] F. S. Abkenar, Y. Zeng, and A. Jamalipour, "Energy Consumption Trade-off for Association-Free Fog-IoT," in *in Proc. of IEEE International Conference on Communications (ICC)*, 2019, pp. 1–6.

[22] Z. Liu, P. Dai, H. Xing, Z. Yu, and W. Zhang, "A distributed algorithm for task offloading in vehicular networks with hybrid fog/cloud computing," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 2021.

[23] P. Dai, K. Hu, X. Wu, H. Xing, F. Teng, and Z. Yu, "A probabilistic approach for cooperative computation offloading in mec-assisted vehicular networks," *IEEE Transactions on Intelligent Transportation Systems*, 2020.

[24] C. Fricker, F. Guillemin, P. Robert, and G. Thompson, "Analysis of an offloading scheme for data centers in the framework of fog computing," *ACM Transactions on Modeling and Performance Evaluation of Computing Systems (TOMPECS)*, vol. 1, no. 4, pp. 1–18, 2016.

[25] K. R. Apt and T. Radzik, "Stable partitions in coalitional games," *arXiv preprint cs/0605132*, 2006.

[26] M. Ahmed, M. Peng, M. Abana, S. Yan, and C. Wang, "Interference coordination in heterogeneous small-cell networks: A coalition formation game approach," *IEEE Systems Journal*, vol. 12, no. 1, pp. 604–615, 2015.

[27] A.-L. Barabási and R. Albert, "Emergence of scaling in random networks," *science*, vol. 286, no. 5439, pp. 509–512, 1999.

[28] T. T. D. Site, "Transport infrastructure ireland," accessed: Nov 2021. [Online]. Available: https://www.nratrafficdata.ie

For Personal Use Only