

An Opportunistic Vehicle-Based Task Assignment for IoT offloading

Khaled Srieddine*, Hassan Artail, Haidar Safa

American University of Beirut, Bliss Street, Beirut, Lebanon

ARTICLE INFO

Keywords:

Mobile computing node
MCN
Mobile computing clouds
SINR
Estimated duration of connection

ABSTRACT

IoT is one of the most prolific origins of data that is collected from sensory inputs. IoT devices are characterized by low computational power, thus motivating the computation offloading scheme, a promising technique that mitigates energy performance issues in limited power devices. Instead of offloading to fog or cloud, a new research track is emerging where devices unload their computational needs to vehicles roaming around, and this particular type of offloading is called vehicular fog offloading. In this paper, we propose the use of vehicles as Mobile Computing Nodes (MCNs) roaming around within an area to offer computational services to IoT devices. Assigning tasks to an appropriate MCN by an IoT device that wishes to offload specific communication tasks to it will be formulated based on the quality of the communication channel and on the time needed to offload the computation request. Specifically, the vehicle which will serve as an MCN must be within the range of the requesting fixed IoT device and offer suitable SINR and the estimated duration of connection values, while taking into consideration its mobility. Another challenge would be delivering the results back to the IoT device or another party, knowing that the vehicles will not be stationary. The performance results show that the proposed system represents a promising solution by offering computation offloading services and delivering the results within acceptable times, regardless of where the vehicle might be when it wants to return the results.

1. Introduction

The revolutionary changes in the automotive industry lead companies to invest in the incorporation of advanced technological features in smart vehicles. Modern cars like Tesla, are equipped with powerful onboard computers, storage devices, sensitive radio transceivers, collision radars, and GPS devices [1]. These smart vehicles are considered computers-on-wheels.

Two paradigms are dominating, cloud and fog computing. Fog Computing refers to bringing cloud intelligence near the edge. It facilitates the operation of computation and networking services between end devices (IoT) and cloud computing data centers. Unreliable latency, lack of mobility support, and location awareness are issues induced by cloud computing. The elasticity of the resources and services provided by fog nodes can address these challenges. Fog computing complements the cloud computing paradigm by moving intelligence from the core to the edge of the network [2]. Although these two paradigms are advancing, there are some challenges in adopting them in areas where connectivity to the wide-area network is not reliably available or is costly, thus motivating the transition to mobile vehicular fog computing (VFC) where the vehicles can service computation requests [3,4]. In a VFC environment, the nodes are highly mobile nodes, that have a large computation capacity with limited storage ability. Their existence

near the edge allows for low latency communication, thus allowing local decision-making. Whereas, in vehicular cloud computing (VCC), the nodes are a group of vehicles forming a cloud, providing a highly scalable storage facility, but with medium computation capacity. It is worth mentioning that decision-making in VCCs is remote with high deployment cost, high latency, and limited mobility support due to the lack of wide geo-distribution compared to VFCs [5].

Applications, like monitoring bridges' health and conditions, require low computation and limited power devices [6], which fall under the general category of Internet of Things (IoT). These devices are not designed to run complex algorithms for inferring or predicting times of potential failures or replacement. Instead, they can resort to adopting the offloading paradigm, where they migrate their computation needs to nearby computation resource providers, which could be the cloud, the fog, or, as we propose, the vehicular fog computing. More specifically, in our proposed framework, IoT devices opportunistically take advantage of passing by mobile vehicles. In this paper, we describe the design of this offloading system and study its performance. The proposed system builds on our earlier work in [4] by tailoring the services that vehicular fog computing nodes offer to IoT devices, changing many of the design parameters that were adopted, and detailing the used communication protocol and other functions.

* Corresponding author.

E-mail addresses: kss19@aub.edu.lb (K. Srieddine), hartail@aub.edu.lb (H. Artail), hs33@aub.edu.lb (H. Safa).

The primary motivation for an opportunistic vehicle-based computing architecture to service IoT applications lies in the unique advantages of enabling IoT services in hazardous and difficult to access scenarios, where resources and connectivity are inherently limited and cannot be sufficiently supported by static fog or cloud services. Clouds are distant and suffer from unreliable long latency over WANs [7], and the lack of mobility. Moreover, the energy consumption of data centers is a daunting problem that has gained significant attention from the research community [8–10]. The fog computing paradigm decreases latency and the energy consumption of cloud computing [11] by placing serving resources at the edge of the internet. That is the core idea behind several solutions like cloudlets or fog nodes. However, this type of solution suffers from limited computation ability and a lack of mobility. Installing fog nodes is quite challenging, and the nodes that are installed might get overwhelmed by the amount of data they receive. Add to that, the cost incurred in scaling up resources (computing power) is high. Cars in a Vehicle Ad hoc Network (VANET), on the other hand, can act as mobile fog nodes offering opportunistic services as they move around the city. This is a cost-effective solution that makes use of idle resources roaming around without additional cost for installation. Indeed, vehicles are being equipped with powerful computers, moving around the cities carrying idle resources that are yet to be fully utilized. Such a solution was proven in previous research as a great alternative to the traditional offloading and computational paradigms [12–14]. It is worth mentioning that the vehicular fog computing paradigm helps in decreasing energy consumption incurred by cloud and fog computing, and the shift towards a green mindset that the world has been seeing lately can be translated by motivating the shift to vehicular fog computing.

This work assumes feasible offloading of tasks of IoTs, where tasks are offloaded to a vehicle to achieve the benefits of Edge computing. The main challenges in this scenario are mobility, intermittent connectivity, and delay. While previous work has shed the light on the utilization of MCNs; however, to the best of our knowledge we are the first to introduce a novel delay-tolerant communication protocol that involves decentralizing the task assignment/offloading scheme by sending requests by the IoT devices and receiving the results from the vehicles directly, while utilizing the RSU topology to reduce retrieval delay whenever needed. In this work, we propose a framework that simplifies the offloading mechanism from IoT devices to vehicles directly without the overhead of having a resource manager, or any intermediate agent. We study the mobility and quality of signal aware task assignment mechanisms that aid our offloading system taking into consideration the mobility of the vehicles by estimating the duration of connection (EDOC) and the quality of signal reception (Signal Interference Noise Ratio (SINR)) as a selection criterion for choosing the vehicle to serve the nearby IoT devices. To this end, we frame the contributions of this work as follows:

- Proposing a vehicle-based computing framework for IoT offloading. This framework defines the means of communication between the IoT devices and the mobile vehicular node in a Vehicle Ad hoc Network (VANET) environment.
- Evaluating experimentally the decentralized vehicular-based offloading system and proving the effectiveness of our task assignment selection criterion in enabling direct IoT offloading to mobile computing nodes.
- Presenting numerical and simulation results that prove the reliability of the proposed MCN framework in improving the throughput satisfying IoT devices.

The remainder of this paper is organized as follows. In Section 2, we survey related work in vehicular and edge-cells offloading. We also highlight the uniqueness of our approach. Section 3 presents the proposed approach. In Section 4, we describe the environmental setup and evaluate the performance of the proposed approach. In Section 5, we make some concluding remarks and present our future work.

2. Related work

In this section, we survey the most relevant related work to our proposed framework. We classify them into two main topics: vehicular offloading, where vehicles offload tasks to other MCNs, and edge-cell offloading, where different types of edge devices (mobile phones, IoTs, etc.) offload their tasks to other MCNs.

2.1. Vehicular offloading

A vehicular cloud is a distinctive kind of cloud server that is mobile and has high resource availability along with internet access, where individual mobile devices can be cloud users with a pay as you go model [13]. Vehicular clouds are a means of sharing resources between mobile entities where in some cases, this paradigm proves itself to be more efficient by serving tasks locally instead of sending them to the cloud, which would be more expensive and require more time [12]. Different approaches have been discussed and proposed in the literature.

A mobility-oriented data retrieval protocol for computational offloading in vehicular edge computing was proposed in [14] to efficiently retrieve results of offloaded tasks by using vehicles and RSUs as relaying entities. In that paper, a hybrid protocol between a topological-based protocol and a distance-based forwarding protocol was investigated, where RSUs allocate tasks to suitable vehicles in the vicinity. Another V2V offloading approach was proposed in [15]. Unlike other related work, this approach takes into account mobility as a criterion to select the vehicle responsible for handling the requests. It motivates the use of a fixed infrastructure (RSU) to service computation requests. Moreover, a system where vehicles offer services to other cars (consumers), such as internet access, was proposed in [4]. Ordinarily, RSUs handle such services, but because high demand in urban areas might lead to congestion of the network and low quality of service, the RSUs in [4] store, for each offering vehicle, the type of resources, their attributes, and the required price per resource unit. The RSUs also share the data with each other, thus forming a dynamic registry. Consequently, consumer cars request the required resources from the closest RSU, which in turn, would find the best vehicle that can offer its services and satisfy the consumer's requests. Similar to [4], an approach in which a car can offload some of its tasks to other selected vehicles was proposed in [16]. In this approach, four different selection strategies were studied. It was reported that the multi-attributed selection strategy, which depends on various parameters to select a node, outperformed the random selection, computation capacity-based selection, and distance-based selections. Furthermore, [17] proposed a vehicular cloud model in which the vehicle provides the drivers as well as the occupants of the vehicle with computing resources.

In [18], a contract theory-based incentive mechanism that maximizes social welfare was proposed. It incentivizes the neighboring vehicle to collaborate and share their resources by rewarding contributing nodes. It utilizes RSUs to provide rewards to each vehicle contributing to the network. The proposed scheme shows a 28% higher computing resource utilization, 17.2% lower energy consumption per computing resource utilization, and 17.1% lesser energy

2.2. Edge-cells offloading

Edge-cell offloading is a variant of opportunistic fog-based computing where vehicles might offload their data to store them. In addition, other computationally limited devices might collaborate to service each other or offload their data to fog nodes whether they are mobile or stationary. In this regard, an approach that considers vehicles as cache edge data cells and is controlled directly by the ISP was proposed in [19]. In this approach, devices might query directly (via WiFi or 802.11p) other devices for faster retrieval of information instead of querying the infrastructure, which minimizes the load on the cellular

infrastructure. When a user requests content that is not immediately available in nearby vehicles, he or she would agree to wait until a car with the content moves within range. Therefore, this approach centralizes the decision within the ISP (actually using the system of RSUs or a stationary fog managed by the ISP).

A decision-making model based on the object's characteristics such as computational power, storage, memory, energy, bandwidth, packet delivery, and hop count was proposed in [20]. Depending on the characteristic, an IoT device can choose the best stationary cars to be used as fog nodes according to their characteristics. The issue with this approach is that stationary vehicles might get their energy depleted when serving other devices since their batteries are not being recharged.

An opportunistic fog model that takes advantage of stationary and mobile fog nodes was proposed in [21]. Here, the concept of a context-aware fog cluster was introduced. It is managed by a fog manager which is a fog node that has access to the cloud. The manager is responsible for collecting context data, such as network information, environment data, rates of new joining nodes, and existing nodes disconnecting. The cluster manager node's inference and predictions are used to formulate policies related to task scheduling, node mobility, scaling, fault tolerance, security, and pricing. It also chooses the appropriate cluster member to be used. If a better candidate exists, the role is migrated to it.

The problem of offloading computation from a mobile device to the cloud, fog nodes, or other mobile nodes in the vicinity was tackled in [22]. A layer composed exclusively of mobile devices that collaborate opportunistically, as a first resort for offloading, was added. Four different scenarios were studied: Device-to-Device (D2D), cloud-only mobile devices, D2D and cloud, and finally, the fourth scenario, which adds to the previous scenario fog nodes that are located at the edge of the network. It was evident that the existence of a crowd computing layer below the fog nodes layer (drop computing) is beneficial and suitable for restricted mobile networks.

Moreover, a holistic view of the use of ML-based techniques to manage computation offloading was introduced in [23]. It provides a taxonomy that is classified into 3 main fields: reinforcement learning, supervised, and unsupervised learning mechanisms. On the other hand, [24] provided a comprehensive and systematized literature review of game theory-based decentralized mobile edge computing. It discussed approaches that are categorized into classical game mechanisms, auction theory, evolutionary game mechanisms, and hybrid-base game mechanisms. Whereas, in [25] a comprehensive and detailed survey, focusing on stochastic based offloading mechanisms, was proposed. The proposed taxonomy considers the Markov chain, Markov process, and Hidden Markov Models.

Furthermore, a machine learning-based computation offloading and resource provisioning mechanism was proposed in [26]. By utilizing learning automata as a decision-maker to offload incoming workload into the edge or cloud servers, and using machine learning models to make scaling decisions, the proposed mechanism was able to increase CPU utilization and reduce execution time and energy consumption. In [27], a hybrid deep learning offloading approach was proposed to offload data from mobile devices to fog and cloud servers. The mechanism was able to achieve near-optimal accuracy regarding offloading decision-making, latency, and energy consumption predictions in a self-management framework. Whereas, the approach proposed in [28] finds the best place to run modules on the edge devices, fog, or the cloud. It utilizes a deep reinforcement learning algorithm to find the best destination to execute a computation while compromising power consumption, execution cost, delay, and network resource.

2.3. Discussion

In our approach, we adopted the vehicular cloud concept that offloads tasks from stationary IoT devices to mobile vehicles. We considered cars as edge-computing cells, where unlike [19], nearby IoT

devices might contact an MCN directly. Also unlike [4,16,29], we took into consideration the SINR when selecting an appropriate car to perform a task or computation, and the estimated duration of connection, bearing in mind the mobility of the MCN. Furthermore, IoT devices can directly discover MCNs and choose the most suitable one instead of crowding the RSU's registry, hence reducing delays and traffic in the network. In addition, we did not consider the computation power because we assume that all vehicles have the needed computation power and computation resources to offer. It is worth mentioning that in our approach we achieve a 5.6 s average delay in the worst case compared to the 11 s average delay achieved by the multi-attributed selection strategy suggested in [16]. The reason behind this difference is that in [16], tasks were partitioned into smaller ones. Beyond that, we tackled some of the challenges that VANET clouds suffer from, which are not dealt with in [17]. These pertain to resource allocation and sharing, communication, and coordination between VANETs and clients.

Furthermore, in our approach, factors such as the high mobility, communication signal strength, unstable communication links, and efficient selection mechanism of participants are embodied in two main factors namely, the EDOC and SINR. Unlike [15,19], mobility in our approach depends on more than the mere speed of a vehicle, as it also accounts for the direction and the network of roads.

Table 1 shows the features of our approach as compared to other related works. [14] tackled offloading to vehicles to decrease network and RSU congestion. However, it considered vehicles, RSUs, and mobile phone users who want to do the offloading. We, on the other hand, have an environment of IoT devices (clusters of large numbers of devices that are characterized by small packets and periodic or event-driven transmissions) that look for MCNs that are public vehicles. Along with that, we exploited the fact that nearby IoT devices may have similar traffic and transmission patterns for group communication. Soto et al. [14] utilizes the RSU for task assignment to nearby vehicles. Although, RSUs are distributed, however, with a limited number of deployed RSUs in a certain area our approach decentralizes the task assignment process by directly associating entities (IoT-MCN) with each other without the need for a third party, which allows our approach to scale efficiently. Also, the mobility model in [14] does not allow for calculating the expected estimated duration of the connection, which aims to find the vehicle that stays in the range of the IoT device for the maximum time. Moreover, [14] did not consider the quality of the received signal (SINR) at each IoT device, in contrast to our approach that does. Also, that work reportedly achieves a 48% task retrieval rate on average, whereas our approach achieves a retrieval rate that is at least 80% by taking into consideration QoS metrics into consideration (EDOC and SINR). This not only shows that the features we use for MCN selection are better but also demonstrates that the system as a whole is designed in a robust way to accommodate for the variability and the change in such systems. Additionally, [30] suggested offloading from vehicle to other vehicles and using RSUs as coordinators and cluster managers without taking into consideration the SINR and the expected duration of the connection. [21,22] considered mobile devices and IoT devices offloading to MANET devices, which typically are energy limited. However, only [21] took into consideration a coordinator who orchestrates the offloading mechanism, which is a fog or a mobile node, and performs some form of context-aware clustering. Although they consider the quality of service when choosing a node to offload to, however, they disregard the estimated duration of connection and provide a general framework rather than an explicit one, as we do. Moreover, [26,27], and [28], used a centralized method to offload from edge-cells to fog and cloud servers, while utilizing machine learning techniques. Along the same line, [18] utilized RSUs as a central authority in managing the incentives for the vehicle to vehicle offloading. The central authorities proposed by [18,26–28] will become a bottleneck in highly dense areas. Whereas, in our approach we decentralize decision

Table 1

Comparison with existing systems based on the existence of a Management Unit (MU), Quality of Service (QoS), Mobility (M), Scalability (S), Process Computing Handover (PH) and Result Handover (RH).

	Mode	Approach	MU	QoS	M	S	H	
							PH	RH
Our approach	M2V	IoT device direct offloading to a vehicle based on SINR and EDOC. Results routed back to IoT device via the VANET	✗	✓	✓	✓	✗	✓
[14]	V2V	Vehicle to Vehicle offloading using roadside infrastructure. Results are routed back using a topological-distance based routing	✓	Partial	✓	✗	✓	✓
[15]	V2V	Vehicle to vehicle direct offloading assisted by the RSU over the DSRC channel.	✗	Partial	✓	✗	✗	✓
[4]	V2V	RSU assisted vehicle to vehicle offloading, which enables vehicles to discover and consume the services of mobile cloud servers within a certain area.	✓	Partial	✓	✗	✗	✓
[16]	V2V	Multi-attributed selection strategy for offloading computation from vehicle to vehicle.	✓	Partial	✓	✗	✗	✓
[17]	V2V	VANET-Cloud computing model that allows VANET network to access such cloud resources for computation offloading	✓	✗	✗	✗	✗	✗
[30]	V2V	Task allocation framework for V2V computation offloading that considers quality loss, fog capacity, and service latency.	✓	✗	✓	✗	✓	✗
[19]	V2V	Vehicular cache system that combines the mobility of vehicles with delayed content access to increase the number of cache hit	✓	✗	✗	✗	✗	✗
[20]	M2V	Opportunistic mechanism to select vehicular computing resources based on computation resources, power consumption, hop count...	✗	✗	✗	✗	✗	✗
[21]	I2M	Opportunistic fog model that takes advantage of stationary and mobile fog nodes. It depends on a fog cluster manager to orchestrate the offloading mechanism.	✓	Partial	✓	✗	✗	✗
[22]	M2M	Offloading and computation from a mobile device to the cloud to fog nodes or other mobile nodes in the vicinity	✗	✗	✓	✗	✗	✗
[18]	V2V	Incentive mechanism to offload computation to neighboring vehicles, while utilizing RSUs to manage rewards to contributing nodes.	✓	✗	✗	✗	✗	✗
[26]	M2M	Mobile offloading mechanism to fog and cloud servers, managed and provisioned by machine learning	✓	✗	✗	✗	✗	✗
[27]	X2M	Autonomous framework as a control model for self-management offloading mechanism for various edge cells	✓	✗	✗	✗	✗	✗

making by allowing IoT devices and mobile vehicles to take decisions, thus decreasing the congestion for management units.

In addition to what has been discussed above, [Table 1](#) shows the metrics which we used to assess the related works. We evaluated them based on the existence of a Management Unit (MU) that acts as a central coordinator between the different entities. We also evaluated them based on the Quality of Service (QoS), which translates to factors like the EDOC and SINR. We also considered mobility, scalability, and Handover in our evaluation. PH represents if the proposed approaches provide any computing process handover if the computing device cannot continue with a task, whereas RH stands for result handover, to assess if the approaches provide multi-hop result handover.

3. Proposed approach

In this section, we give a detailed description of the design of our framework and discuss its operations and how it works.

3.1. Basic idea

We propose an edge cell opportunistic offloading framework, in which the stationary IoT devices can directly communicate with the mobile vehicles and utilize their resources. IoT devices are clustered to reduce network traffic and improve scalability. In the clustering setup, the cluster members (i.e., stationary IoT devices) forward their offloading requests to their cluster's clusterhead. The cluster head which is associated with a suitable MCN, as described below, sends the requests to the MCN, receives the returned results, and forwards them to the member devices. We do not describe the details of the clustering mechanism of the stationary IoT devices, since it is outside the scope of this work and left for future work. However, a simple K-means

clustering using geometric distance as a metric is adequate. Because of adopting this metric, members of the same cluster are most likely in the same LAN, which reduces the amount of network traffic and communication delay [31]. The assumed clustering technique enables the interference-prone distribution of cluster heads thus, mitigating the interference of cluster heads. In this regard, the cluster head's role can be rotated among the devices in the cluster in a round-robin fashion so that the burden of establishing a connection with a vehicle is shared among all cluster members. Thus, decreasing the toll of the communication overhead on the energy consumption of the cluster head. It is worth mentioning that to be able to rotate the role in a round-robin fashion between IoT devices, all devices share the same characteristics and abilities.

The mobile vehicle selection strategy is based on the SINR and the time needed for sending the offloading request. Moreover, we decentralize the vehicle selection process and run it on the cluster heads instead of assigning an entity the role of a central coordinator.

It is worth noting that our proposed approach is especially effective and useful in disaster scenarios where there are possible damages to the mobile communication infrastructure. In such cases, vehicular fog computing would offer a replacement to the infrastructure [13], thus further motivating the transitioning from regular clouds to the vehicular fog computing paradigm.

As compared to the above-reviewed literature, our work is unique in several aspects. It opportunistically considers roaming/mobile vehicles (mostly public vehicles) for use as mobile computing nodes (MCNs) which offer computation offloading services. As battery energy availability is not a concern in such computing nodes, our criterion for selecting the MCN to associate the IoT device with is the quality of the signal from the MCN, specifically the Signal Interference Noise Ratio (SINR), in addition to the expected estimated duration of connection during which the IoT device can stay connected with the MCN.

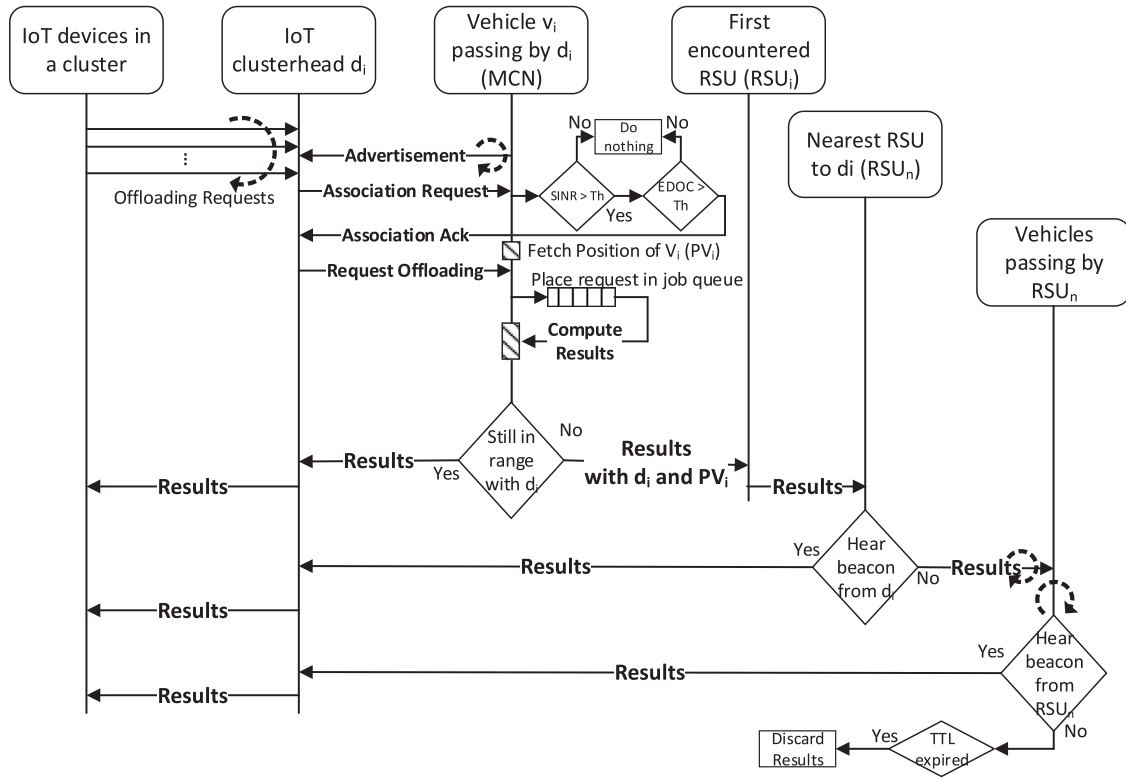


Fig. 1. General sequence diagram of interactions between IoT device and MCN.

Our approach can open the door for a new business model in which public transportation companies and those that offer delivery services could benefit from enabling their vehicles to perform computation services for subscribers and compete to provide improved services to IoT applications.

3.2. Proposed framework

In our approach, the cars usually communicate with each other on the control channel of the Dedicated Short Range Communication (DSRC) protocol to exchange safety messages and advertisements. Whenever the car is in the control channel and willing to share resources, it broadcasts beacon messages to advertise its services periodically. MCNs roaming around might not be willing to participate in the offloading framework, thus some incentives should be put in place to motivate vehicles to share resources [18]. Nearby IoT cluster heads, that have pending computation offloading requests from their cluster members, send association requests if they are not already associated with MCNs. In response to the received request, the car computes the SINR and the expected EDOC with the IoT device (i.e., the vehicle's time within the communication range of the cluster head). Then the MCN sends it an offer (acknowledgment) message that includes these computed values. From here on forward, we refer to the IoT cluster head device as simply the IoT device. The IoT device chooses the MCN in a range that offers a sufficient metric combining SINR and the estimated duration of connection based on the predefined thresholds. It is worth mentioning that the decision-making is at the cluster head and the MCN so that any of the entities can abort continuing the communication if the metrics do not meet a certain threshold to decrease unnecessary traffic, and decrease energy consumed. Moreover, the decision at the IoT device is immediate whenever it finds a vehicle with the appropriate metrics. A challenge lies in delivering the results back to the device by the moving car, which may necessitate the use of other elements in the VANET network, like other cars or the system of roadside units (RSUs). The steps described above are illustrated in

the sequence diagram of Fig. 1. Moreover, to deliver the results back by the MCN to the device directly or through another party, different scenarios are possible as described later.

In our approach, the network is composed of several stationary IoT devices that are dispersed throughout the area. They can be found in forests, parks, bridges, roads, etc. Moreover, to establish a network where we can send and receive data, these IoT devices need to communicate with vehicles that play the role of MCNs.

We assume that the cars communicate with each other using the DSRC protocol to exchange safety messages and advertisements. Whenever an MCN is willing to share resources, it broadcasts a beacon message (advertisement) to IoT devices stationed around it, thus advertising its availability. Consequently, the IoT device needs some computation replies by sending an association request. In response to this request, the MCN computes the SINR level. If this level is greater than a given threshold, it computes the EDOC that should also be greater than a specific threshold. If the EDOC is sufficiently large, the MCN sends an association acknowledgment that it is successfully paired with this IoT clustered device.

We consider two metrics for the IoT device to pair with the most suitable MCN: Signal Interference and Noise Ratio and the Estimated Duration of Connection, the time a vehicle is within the communication range of a clusterhead. To ease the burden on the IoT cluster head device, both metrics are computed by the MCN upon receiving an association request, and the MCN only sends back an association acknowledgment if both values meet their criteria (i.e., above-given thresholds).

3.3. IoT device-MCN association

The communication steps between the MCN and the IoT device are described in the following steps:

1. The association procedure begins by the MCNs periodically transmitting service advertisements that include information

- about their position and availability (e.g., percent of received job queue that is occupied).
2. After receiving the MCN's advertisement, an IoT cluster head device that holds pending computation offloading requests from its cluster members, sends to this MCN an association request for computation offloading. We note here that the IoT device sends a request to every MCN it hears an advertisement from, as long as it did not receive an offer yet (next step).
 3. The MCN computes the SINR level and checks if it is above a given threshold. If it is not, it discards the request. Else, it proceeds and computes the EDOC.
 4. If the EDOC is sufficiently large, the MCN accepts the request if its queue has a room available, and sends the IoT cluster head device an association offer. In its offer, the MCN includes an estimate of the time that it takes to fully service the request, which mainly depends on the number of offloading jobs that are currently waiting in its queue. We note that the request service time does not include the time consumed in forwarding the results back to the cluster head.
 5. Upon receiving the association offer, the IoT cluster head device sends the MCN the set of requests that it has received from its cluster members. If the IoT device receives multiple offers (in response to sending multiple requests), it computes a weighted average of the normalized SINR and EDOC values and selects the MCN corresponding to the largest average.
 6. The MCN performs the computations for all received requests after they are dequeued from the MCN's queue.
 7. The MCN uses a forwarding mechanism to send the results back to the cluster head. The details of this mechanism are described in the next subsection.

$$\text{Weighted Average} = \alpha \times \frac{\text{SINR}}{\text{Average SINR}} + (1 - \alpha) \times \frac{\text{EDOC}}{\text{Average EDOC}}$$

where the weighted average of the normalized SINR and EDOC values are depicted above. The value of alpha can be set based on the distance between the IoT device and the MCN initially but would be increased or decreased to maximize throughput, or using machine learning to account for other factors that influence throughput. In the next two subsections, we describe how the SINR and EDOC are computed by the MCN. In this regard, we indicate that the task of computing the SINR and estimated duration of connection was assigned to the MCN for several reasons. The most important of which is the fact that the IoT device, as we consider it, is a computationally and energy-limited resource. Moreover, the MCN has information about the mobile environment in addition to the position of the IoT device which was piggy-backed by the association request sent to the MCN.

3.3.1. Signal to Interference Noise Ratio (SINR)

The SINR is generally defined as:

$$\text{SINR} = \frac{P_r}{I + N}$$

where P_r is the desired signal power, I is the total interference power, and N is the thermal noise power. Here, the received signal from the IoT device making the request is the useful signal at the MCN, while the signals from the neighboring IoT cluster head devices and vehicles driving in the area and transmitting at the same time produce the interference. The interference power from the IoT devices other than the one submitting the request and the vehicles in the area other than the considered MCN can be computed similarly to the wanted signal, where only the transmitters that are using the same frequency resources are considered.

We should remind that in this work, we assume all vehicles communicate on the control channel of the DSRC protocol and use the distributed coordination function (DCF) to contend for channel access. It follows that interference from vehicles in neighboring areas could take place but not within the same area. This is because DCF

prevents simultaneous transmissions by multiple transmitters within transmission range of each other (same area). On the other hand, we suppose that IoT devices communicate amongst each other using the ZigBee protocol although the cluster head communicates with the vehicles using some form of a gateway (not discussed in this paper), and hence can be considered as a DSRC transmitter. This means that other neighboring IoT cluster head devices (i.e., other than the one that transmitted the association request to the considered MCN) do contribute to the interference component of the SINR equation.

We now describe how we calculate the SINR using a modified model that is adopted from [32]. Since we are targeting an outdoor scenario (unlike [32]), the penetration loss, which is listed in Table 2, reduces to 0 dB. To handle the worst-case scenario, we model fading using Rayleigh with $\sigma = 10^{0.06}$ (6 dB margin) and $\mu = 1$. The interference power from the other vehicles in nearby areas (as we discussed earlier) can be computed similarly to the wanted signal (i.e., using the same parameters and propagation model). It follows that by using the model in [32], the SINR at the receiving MCN can be expressed as:

$$\text{SINR}_{MCN} = \frac{k_1 \times h \times d^{-\gamma_2} \times P_t}{\sum k_2 \times h \times d_j^{-\gamma_2} \times P_t + N}$$

where P_t is the power transmitted by the IoT clustered device that is d meters away from the MCN, k_1 and k_2 are propagation constants (see Table 2), N is the added thermal noise, while h is used to model fading over the Rayleigh channel. The denominator shows the cumulative interference power resulting from other vehicles and IoT cluster heads transmitting at the same time, as discussed before, and each of which is at a different distance d_j .

3.3.2. Estimated duration of IoT device-MCN connection

We consider a deployment where certain vehicles play the role of mobile computing nodes. For the most part, these are public vehicles, like delivery cars, taxis, and buses, whose owners may be compensated based on the number of computations the onboard units (OBUs) of their cars perform. A computing vehicle is equipped with a navigation system that associates the global positioning system (GPS) positions to road maps, to be able to know its locations (i.e., geometric coordinates). Similar to [33,34], we calculate the estimated duration of connection, where the UE in [33,34] resembles the lightweight IoT device that we describe in this work. In this work, it is the vehicle (not the IoT device) that calculates the estimated duration of connection, unlike how it is done in [33,34], to lessen the computation burden on the IoT device, taking into consideration that the IoT device does not know its surroundings or environment. After the MCN receives the association request, it sends an association acknowledgment (offer), which is sent only if the SINR is sufficiently large, and the estimated duration of connection calculated by the MCN is sufficient for the IoT device to associate with the vehicle and send its request (EDOC > time threshold).

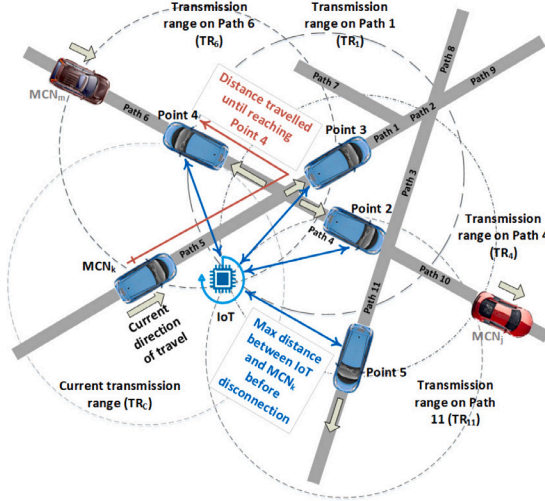
A simple general scenario is illustrated in Fig. 2(a), where each MCN periodically broadcasts a beacon which includes its position. This allows the IoT device to know if the MCN is within its communication range. We note that the location of the vehicle is needed since the fact that the IoT device is within the transmission range of the vehicle is not sufficient, given that the transmission range of the IoT device may be much smaller as compared to that of the MCN. Fig. 2(a) shows the current location of an MCN (MCN_k) in the bottom-left region of the map, along with its possible farthest positions (Points 2, 3, 4, and 5) before the IoT device disconnects from it due to exiting its transmission coverage. The transmission coverage of MCN_k in the different positions is illustrated using dotted circles, while its range is depicted through the two-headed arrows that connect the IoT device.

The MCN can change its path or direction after reaching an intersection. Depending on which road segment the vehicle goes to, the estimated duration of the connection between a specific IoT device and an MCN varies widely. The example in Fig. 2(a) shows MCN_k ,

Table 2
SINR parameter values.

Description	Value
Propagation factor on the direct link (between the clusterhead and MCN) k_1	$10^{-9.84}$
Propagation factor on the link from other vehicles and clusterheads k_2	$10^{-14.178} \times 10^{\frac{-2L}{10}}$
Path loss exponent γ_2	2
Penetration loss	0 dB

(a) MCN paths of roads showing its farthest locations before it disconnects from the IoT device



(b) Zoom-in of Figure 2a

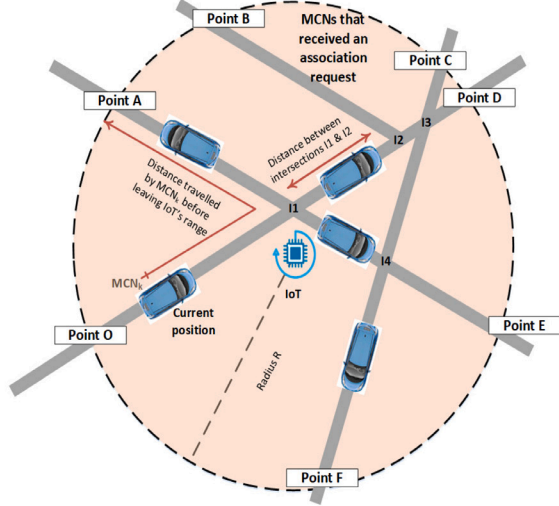


Fig. 2. Estimated duration of IoT device-MCN connection.

which is in range with an IoT device, is approaching the intersection at the center of the map. After reaching the junction, the MCN can take one of the three possible routes, each of which results in a different EDOC. After crossing points 2, 3, 4, and 5 on paths 4, 1, and 11, respectively, MCN_k loses the IoT device coverage and becomes out of the communication zone.

Given the MCN's location along with that of the IoT device, the MCN can compute the coordinates of the exit points 2, 3, 4, and 5 after receiving the IoT device coordinates. It can then compute, using its knowledge of the road map (with the aid of the GPS) the driving distances to each of these points. Given the MCN's knowledge of its average speed, it can translate the distances to estimated times. Out of these times, the MCN computes the expected estimated duration of the connection, and consequently, sends the IoT device an association offer if both the SINR and the EDOC are sufficiently large. With this, the IoT device becomes aware of the fact that the MCN received an association offer from is suitable to perform the computation task for it.

Moreover, Fig. 2(b), which is a zoom-in of Fig. 2(a), describes in more detail the concept of the estimated duration of the connection. In this figure, MCN_k is currently within the IoT device's communication range and approaching intersection I_1 , after which it takes one of the three possible paths. An MCN is equally likely to take any path upon reaching a junction. However, different probabilities can be assigned as a function of some measurements or statistics, like the average traffic density on each of the outgoing paths. Such data may be obtained from the traffic authority database, or learned by the RSUs that periodically receive beacon messages from the passing by MCNs.

We denote by M the number of exit points and P the number of distinct paths from the IoT device to exit points. We assume the MCNs have the same transmission range R_{MCN} , and so, when an MCN needs to calculate its estimated EDOC, it draws a circle having a radius R_{IoT} and the IoT device as a center. It then identifies the intersection points of this circle with the roads in range. As shown in Fig. 2(b), the points

are A, B, C, D, E, and F. They represent exit points beyond which the MCN loses the connection with the IoT device, and can be referred to as the set $\{p_1, p_2, \dots, p_M\}$. Next, we identify the set of possible distinct paths from the MCN's location to the exit points and their corresponding lengths in meters $\{l_1, l_2, \dots, l_P\}$, where $P \geq M$.

Given the car's average speed (including stopping at red lights and stop signs and slowing down on turns), there are corresponding P times, i.e., $\{t_1, t_2, \dots, t_P\}$, which are obtained by dividing the lengths of the paths by the average speed of the MCN as depicted in Eq. (1). Next, given that the MCN may take any road upon reaching an intersection with equal probability, the average (expected) time that it may take until it disconnects from the IoT device is specified in Eq. (2).

$$t_i = \frac{l_i}{AverageCarSpeed} \quad (1)$$

$$EDOC = \frac{t_1 + t_2 + \dots + t_P}{P} \quad (2)$$

3.4. Delivery of results to IoT device

To deliver the results back to the device directly or through another element of the network, three different scenarios are possible as illustrated in Fig. 3.

After the MCN (call it MCN_i) receives the request from the IoT cluster head device (d_i), it fetches its current position (pv_i) and associates it with the address of the IoT cluster head device (d_i, pv_i), and then adds the computation task in the request to its offloading jobs queue (along with d_i and pv_i). When MCN_i gets to work on the task and finishes it, it checks if it is still within the communication range of d_i . If it is, MCN_i directly sends the results back to d_i as depicted in Fig. 3 (Scenario 1). However, if MCN_i has left the communication range of d_i , it caches the results until it hears a beacon from an RSU (i.e., the first encountered RSU on its path, which we denote as RSU_j), after which it relays the results to it. Next, the RSU checks if it is in range with the IoT device, and if it is, it transmits the results to the device (d_i) after checking

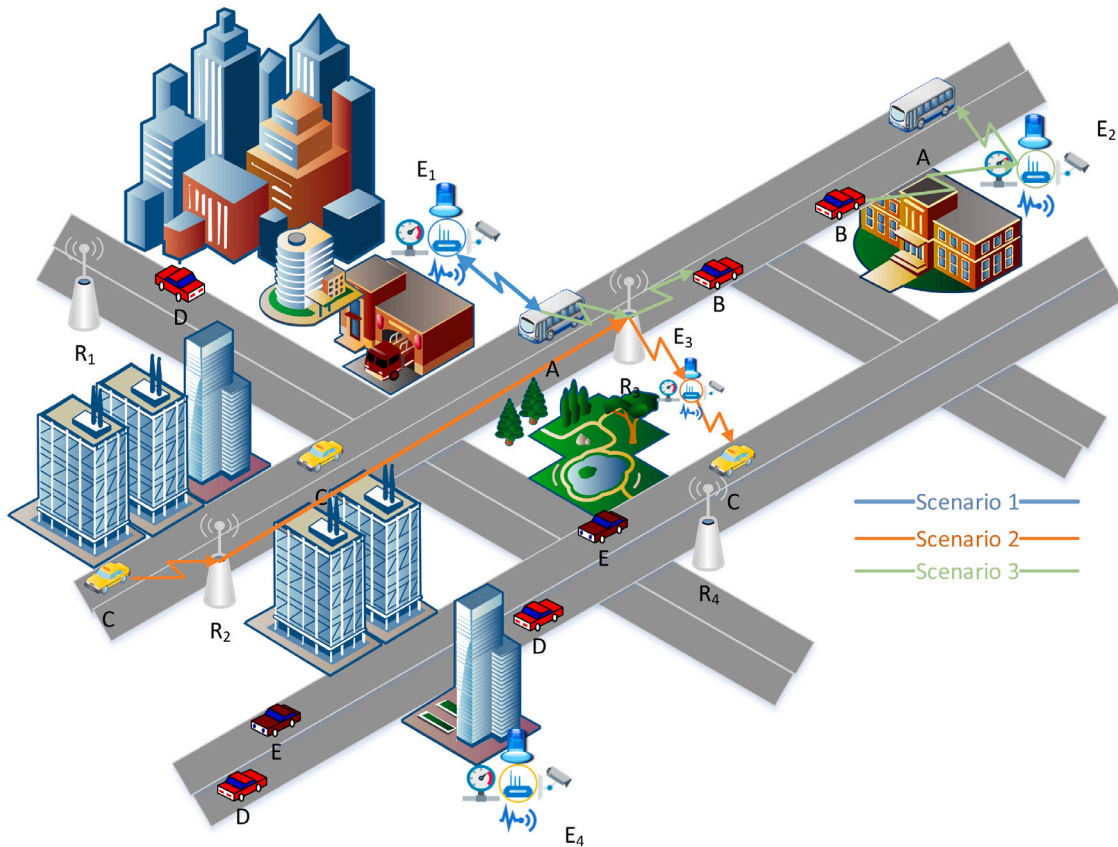


Fig. 3. Scenarios for delivering the results to the IoT device.

Table 3
Retrieval mechanisms.

Retrieval mechanism	Description
Mechanism 1	Results are retrieved from the same vehicle that performed the computation
Mechanism 2	Results are retrieved from a nearby RSU.
Mechanism 3	Results are retrieved by another vehicle moving towards the IoT device.

the QoS condition (SINR). The results packet includes the address of the IoT clustered device (d_i) and the position pv_i of the MCN when it received the request from d_i . Knowing pv_i , this particular RSU, RSU_i , now forwards the results (along with d_i and pv_i) to the nearest RSU to pv_i (RSU_n), which should be the nearest RSU to d_i . RSU_i can determine RSU_n because it knows the whole RSU network in the surrounding area. We note that RSU_n could be RSU_i itself. We refer to this second scenario as Scenario 2.

If the IoT device is not within communication range with any RSU, meaning that RSU_n is not in range with d_i , RSU_n starts forwarding the packet to passing by MCNs until the TTL of the packet expires. To allow the MCNs that carry results packets to forward these packets to their rightful destinations, each IoT device that had sent a computation request periodically transmits a short beacon that includes its address, the request-id, and a flag indicating that it is waiting for results. As an MCN that is passing by the device and is carrying the results packet hears the beacon, it transmits the packet to the device. With this scheme, if the IoT device receives multiple packets with the same request-id from MCNs, it processes the first packet and discards the others. Here we also mention that the vehicles that carry the results discard these results after a specified TTL. We refer to this scenario as Scenario 3. We utilize RSUs to make use of their distributed nature. We plan in our future work to also utilize the V2V forwarding mechanism to enhance the success rate of our approach whenever VANETs infrastructure is lacking. Consequently, we summarize the retrieval mechanisms utilized in this work in Table 3.

3.5. Overhead and time complexity

In this section, we discuss the time complexity of our approach. Remember that cluster heads utilize MCNs found in the vicinity when receiving an advertisement from a passing vehicle. Thus, the complexity of the approach is based on the existence of the vehicles and their density in the vicinity. We study the time complexity of the approach with a minimum number of vehicles (20 \ \text{V}\ \text{HR}) which shows the worst-case scenario and with a maximum number of vehicles (100 \ \text{V}\ \text{HR}) for the best-case scenario. We choose the aforementioned numbers for the best-case and worst-case scenarios to try simulating a real-world scenario. The time complexity achieved in the worst-case scenario is 1.4 s whereas the time complexity in the best-case scenario is 0.2 s. It is worth mentioning that these values are based on the opportunistic availability of vehicles in the vicinity.

4. Environment setup, performance evaluation, and results

In this section, we evaluate our proposed approach using various metrics, but first, we describe the setup of the evaluation environment and the tools used.

4.1. Environment setup

To conduct our simulations, we set up our environment using the ns-3 simulator [35] and the Simulation of Urban MObility (SUMO) tool [36].



Fig. 4. Ottawa urban center [14].

Table 4

Simulation parameters with their default values.

Parameter name	Default value
RSU power (dBm)	46.0206
MCN power (dBm)	46.0206
IoT clusterhead device power (dBm)	5
RSU urban transmission range (m)	145
MCN urban transmission range (m)	145
IoT urban transmission range (m)	40
Number of RSUs	21
Number of MCNs	20–100
Number of IoT clusterhead devices	17–337
SINR threshold (dB)	-7
EDOC threshold (s)	4
Simulation time (s)	3600

The area used in the simulation was adopted from [14], which is a map of the downtown area of Ottawa Fig. 4. Twenty-one RSUs were distributed with a separation distance of approximately 300 m. All service cars which were inserted into this simulation are public vehicles that have predefined trajectories and schedules, from buses following a bus schedule, taxis with predefined routes, and delivery cars driving to destinations and back to their origins (see Fig. 4).

Table 4 lists the used simulation parameters in our system and their default values, as inferred from [37,38] for most of them. Moreover, we use the Log Distance propagation loss model since it was identified to be more appropriate for the urban environment [37]. We also used the Two Ray Ground model in simulating rural environments with fewer obstructions and added Nakagami-m fading, which is an essential means to make a simulation of propagation more realistic. We account for the clustering mechanism, consequently, IoT devices use the Zigbee protocol [39] to communicate with each other and share data.

For the IoT network setup, we have temperature sensors that collect 16 readings per hour, where each reading is a floating-point value of 4 bytes, resulting in a payload of 64 bytes. After adding the 8 bytes source address found in the Zigbee MAC header, we end up with 72 bytes. Given the maximum payload size in the DSRC packet (1438 bytes) [40], the cluster head can service up to 19 IoT devices at a time in each data request. Because of this, we set the maximum size of the cluster to 19 devices. In our setup, the sensory readings are offloaded to apply predictive machine learning algorithms, which is considered a substantial task to be handled locally by an IoT device. To simulate the processing times of the predictive machine learning tasks allocated to the vehicles, we range task computation times between 0 and 5 s, which is more than sufficient given the instantaneous output that is typically produced by a trained deep learning model seen in [41,42]. We highlight that we use SINR and EDOC thresholds that are neither lenient nor restrictive similar to [34]. Moreover, we plan in our future work to study the effect of varying the thresholds by utilizing predictive machine learning algorithms to assist in analyzing the performance of our approach. The used thresholds provided Task computation times

Table 5

OBU specification.

Microprocessor	2.5 GHz Intel Core i5-3210M
Memory	4 GB 1600 MHz DDR3
Video graphics	AMD Radeon HD 7670M (2 GB DDR3 dedicated)
Hard drive	750 GB SATA (5400 rpm)

are directly proportional to the complexity of the task. Moreover, the communication range of the cluster heads with vehicles is 40 m. We assume the OBU of a vehicle possesses the minimum shown specifications mentioned in Table 5 below, although we mention that a modern vehicle's OBU is more powerful than the specifications.

To simulate our approach, we spread on average 1 IoT device per 1 m², which results in 6421 IoT devices spread throughout our simulation area. We cluster the 6421 devices into groups of 19 devices each, as we have described before. This gives rise to 337 IoT cluster heads that are dispersed randomly across the area.

To calculate the number of runs required for achieving at least a 90 percent confidence level, we ran a scenario with default parameter values ten times. For each simulation run, the pseudo-random generator seed, the movement file, and the dispersion of IoT cluster heads were changed. The average successfully retrieved computation ratio and the average serviced IoT cluster heads were calculated. Consequently, the number of runs required to achieve 90 percent confidence was computed using the central limit theorem, as discussed in [43,44]. The +/- precision value for the ratios mentioned above is 0.03. However, the number of runs was recomputed as we varied the vehicle density from twenty to a hundred cars per hour of simulation. This variation resulted in a different number of simulation runs depending on the vehicle density. The ten samples consistently output the mean and standard deviation in close range to each other for a certain vehicle density. The maximum simulation run was 10. On average, we performed 15 simulation runs for each vehicle density.

4.2. Results and evaluation

In our simulation, we configured each IoT device to offload 16 computation requests per hour that are randomly distributed in time. After the cluster head receives the data from the cluster members, it concatenates the 8-byte source address of each cluster member to its respective Zigbee payload. The size of the payload in the computation request packet that the cluster head sends to the MCN is 1368 bytes. This amount of data is the maximum that can be stuffed in a DSRC packet. Moreover, MCNs which are public vehicles mostly commute at low speeds with an average speed of 30 km/h. We define in Table 6 the performance metric used to evaluate the proposed approach.

4.2.1. Varying queue size (1–50)

To study the effect of implementing a task buffer on the MCNs for queuing the incoming IoT devices computation requests, we varied the queue size of the vehicles from 50 tasks to 1 task (in the latter case, the MCN would reject a request if it is currently busy processing one) as shown in Fig. 5. We fixed the number of cars and IoT devices to 37 MCNs and 337 IoT cluster heads (corresponding to more than 6400 total IoT devices in the simulation area). Each MCN follows a random trajectory in the map similar to [14]. Our framework recorded a retrieval rate (proportion of received responses to first-time-submitted requests) of more than 90%. On the other hand, a cluster head starts a timer upon submitting a request, and if it does not receive a response within the timeout interval, it resends the request, possibly to a different MCN after it associates with it.

Fig. 5 refers to the original requests, not the retransmitted ones. It shows a sharp increase initially as the queue size of the MCNs is increased, but then increases mildly after a queue size of 20 computation tasks. The reason for the change in slope could be attributed to reaching the processing throughput of the MCN, as configured in the simulations.

Table 6

Performance metrics.

Performance metric	Description
Retrieval rate	Proportion of received responses to first-time-submitted requests
Rate of serviced IoT devices	The percentage of IoTs that successfully offloaded and retrieved their results out of the total number of IoTs.
Retrieval delay	The wait time of the IoT devices minus the processing time at the MCN
Rate of delayed requests	The percentage of requests delayed due to not receiving the results for a prior request.

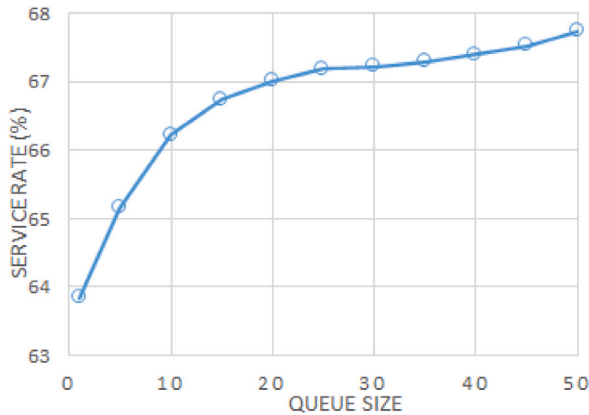


Fig. 5. Percentage of serviced IoT clusterheads upon their initial requests.

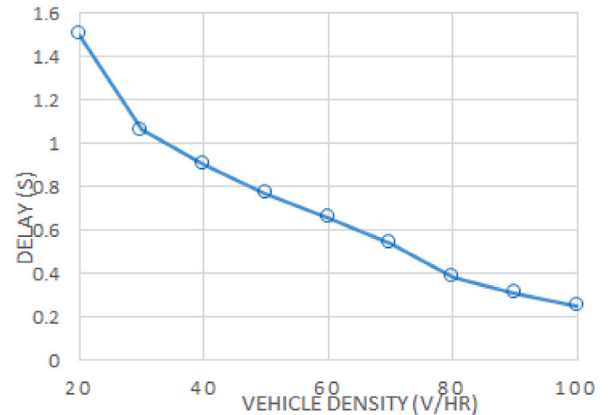


Fig. 7. Retrieval delay vs. vehicle density.

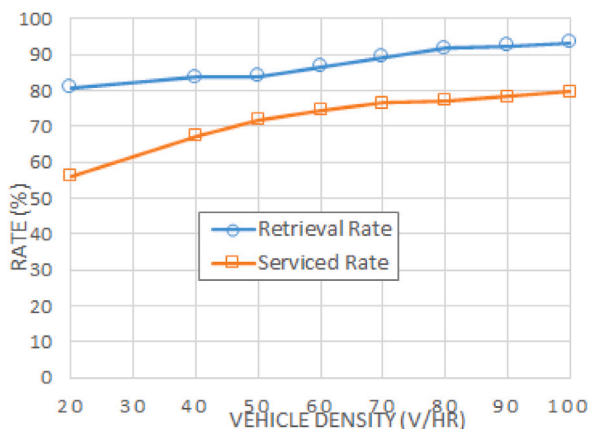


Fig. 6. Successful retrieval rate while varying vehicle density.

4.2.2. Varying vehicle density (20–100) while varying queue size (50–1)

In this second experiment, we varied the vehicle density from 20 cars to 100 cars in the area while randomizing the route of every vehicle and altering the distribution of the IoT cluster randomly. We randomize vehicle routes to simulate real-life scenarios, where the route is decided prior to the simulation, and rarely do vehicles follow the same route. The retrieval rate (defined earlier) ranged from 80% to 95% with minor fluctuations. This nearly-constant rate can be explained by the fact that responses from the MCNs can be routed back to the requesting cluster heads by the MCNs that performed the computations, the RSU network, or other passing-by vehicles using the carry-and-forward mechanism. On the other hand, the vehicle density and the queue size have a direct effect on the serviceability of IoT devices.

The percentage of serviced IoT devices is depicted in Fig. 6. The percentage of serviced IoTs ranges between 58% and 80% and conveys the almost-linear relationship between the rate of serviced IoTs and vehicle density. Due to the randomized vehicle routes during the simulations, some minor fluctuations appear in the serviced rate results. We define the rate of serviced IoT devices as the percentage of IoTs that successfully offloaded and retrieved their results out of the total

number of IoTs. As the vehicle density increases, we, therefore, expect an increase in serviced IoT devices that attempted to be serviced.

As part of the same experiment (i.e., varying the vehicle density) we measure in Fig. 7 the data retrieval delay, which we define as the wait time of the IoT devices minus the processing time at the MCN. The non-smooth slope seen in the figure is attributed to the randomness of the vehicle trajectory, and the availability of cars within the transmission range of cluster heads. As the vehicle density increases, the delay decreases due to the availability of passing by MCNs. It is worth mentioning that higher densities lead to lower average speeds and contribute to a decrease in retrieval delay, which would increase the retrieval rate of the 1st retrieval mechanism described in Table 3.

4.2.3. Varying IoT clusterhead density (337 - 17)

An important performance metric is a scalability. Therefore, we study the system performance with decreased resources (less vehicle density) of only 35 vehicles, each with a queue size of 10. For this, we vary the IoT device density from 17 to 337 clusters and study the percentage of tasks retrieved successfully along with the rate of serviced IoT devices, as depicted in Fig. 8, and the retrieval delay as illustrated in Fig. 9.

Fig. 8 shows the retrieval rate versus the vehicle density, which shows an initial moderate increase, and then stays almost constant after the density reaches 150 IoT cluster head devices in the area. The retrieval rate is the portion of the submitted computation requests for which the results were delivered successfully to the IoT cluster heads. To explain the results, we note that the IoT devices (clusters) are distributed randomly in the area, and hence, as their number increases, it is more likely to access them and deliver the results to them. Indeed, that is because these results are sent by roaming MCNs and RSUs using one of the three delivery scenarios. However, there is a point beyond which adding more IoT devices in the area does not contribute appreciably to accessing them by the MCNs or the RSUs. We remind that the results of the offloading requests can reach the requesting IoT devices in three mechanisms. Namely, from (1) the MCN directly (if it is still in range with the requesting IoT cluster head device, (2) the RSU network (if an RSU is in range with the cluster head), or via a carry-and-forward vehicle that became in range with the cluster head device. Moreover, the retrieval mechanism adopted is directly affected

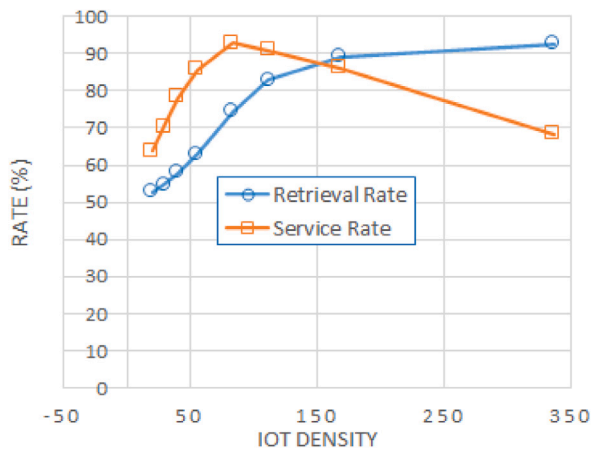


Fig. 8. Successful retrieval rate while varying IoT density.

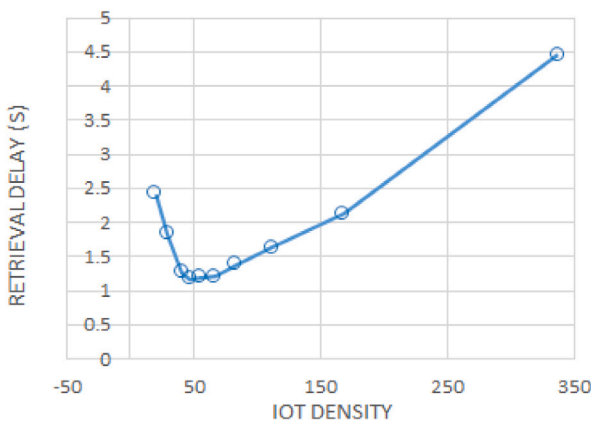


Fig. 9. Retrieval delay while varying IoT density.

Table 7
Average percentages of the retrieval mechanisms.

Variation mechanism	Mechanism 1	Mechanism 2	Mechanism 3
Experiment 1	54%	17%	29%
Experiment 2	62%	18%	20%
Experiment 3	57%	15%	28%

by the mobility pattern and the density of vehicles, especially that with the increase of the density the average vehicle speed would decrease leading to a change in the choice of retrieval mechanism.

Fig. 9 shows the retrieval delay versus device density. It shows that the delay decreases and then increases. This can be explained as follows. When increasing the number of devices in the area, the devices on average become increasingly closer to RSUs or vehicles that hold the results. However, as the number of devices continues to increase, the queues at the MCNs become full, thus making the requests wait longer on average before they are processed.

As can be noticed the service rate, the portion of submitted requests, exhibits a trend that is opposite to the delay that was described earlier. As more devices are added, it is more probable to find nearby MCNs that satisfy the SINR and EDOC criteria. However, as the number of devices continues to increase, the rejection rate by the MCNs grows due to their filled buffers.

Table 7 shows the average percentages of the retrieved results classified according to the delivery mechanisms described in Table 3. The experiments studied are varying queue size described in Section 4.2.1, varying vehicle density 4.2.2, and varying IoT density 4.2.3. We list the averages for the experiments after varying the queue size (while

Table 8
Default settings for stress-testing experiments.

MCN queue size	20
MCN density	50
MCN processing time (s)	0.1
IoT clusterhead density	337
IoT device density	6403

fixing the other parameters to their default values), varying the vehicle density, and varying the IoT device density. The results show that the majority of the tasks are of the first mechanism and that is mainly due to the conditions set for the selection method. This method, as described earlier aids in finding the vehicle candidate with the largest weighted average of the estimated duration of connection and SINR values. Mechanism 2 depends on finding RSUs that are in range with the IoT devices, which is not as helpful in delivering the results to the IoT devices as moving cars (Mechanism 3). It is worth noting that the mechanisms of retrieval depend on the communication signal which is affected by the propagation loss models. In our approach described in Section 3.4 a nearby RSU to an IoT device d_i forwards the result directly if it receives a beacon message from the IoT device along with checking the QoS conditions. Thus, due to the attenuation of signals induced by propagation loss models these beacon messages sent by IoT devices would be lost which explains the retrieval rate increase of the 3rd mechanism. Moreover, it is worth mentioning that different groups of tasks that would require more computational time (>5) to finish would also directly influence the retrieval mechanism adopted.

4.3. Stress testing

In addition to the previous experiments, we also ran additional experiments to test the limits of our proposed system. The default settings of the experiments are listed in Table 8.

4.3.1. Rate of requests

In this experiment, each IoT device collects 16 measurements, then relays them to the cluster head. Upon receiving these measurements, the cluster head offloads the computation requests from all the cluster members to nearby MCNs.

We varied the rate of requests from 1 request per second per IoT device to 1 request per hour. At the end of each interval, as set by the rate of requests, the IoT device generates a request randomly with a probability of p . If it already had sent a request and did not receive the results back, the device resends the same request and delays the new requests. Hence, it implements a delayed requests queue, in which new requests (generated with a probability p) are enqueued, and out of which delayed requests are dequeued when it is time to send a request (as per the rate of requests).

Fig. 10 shows the percentage of delayed requests while varying the rate of requests. As expected, the percentage of delayed requests rises with the increase in the rate of requests, while noting that there is a limited number of vehicles within a relatively large area. The figure also illustrates that the percentage of delayed requests decreases to zero as we decrease the rate to 1 per hour. Almost inversely proportional to that is the percentage of returned results from the original (non-delayed) requests (also shown in Fig. 10). These results make sense, as they have to do with the load on the limited (fixed) number of MCNs in this experiment.

4.3.2. Queue size

To stress-test the effect of the queue size on our framework, we varied the queue size from 20 to 250 while fixing the rate of requests to 1 per 60 s per IoT device. The choice of this request was made in accordance with the results from the previous experiment.

The results that are depicted above in Fig. 11 show an enhancement in the percentage of returned results for the original requests, and

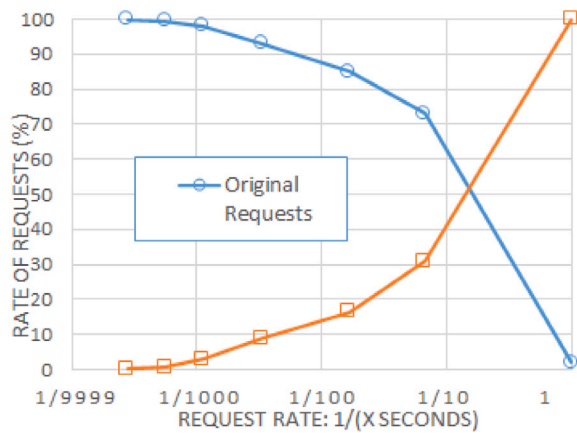


Fig. 10. Percentage of delayed requests as we vary the rate of requests.

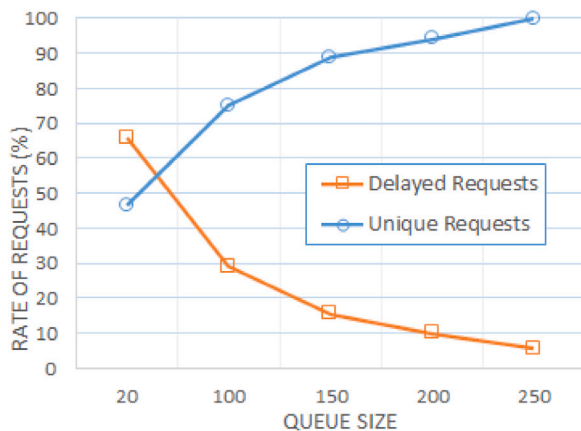


Fig. 11. Percentage of retrieved requests as we vary queue size.

conversely, for the delayed requests. These are expected results, since increasing the queue size at the MCNs increases their ability to absorb more incoming requests. Also, the slope of the rate of returned results of original requests decreases because the MCNs are approaching their processing capacity.

5. Conclusion and future work

The increased generation of data by IoT devices motivated the development of offloading schemes to help ease the computational burden of the IoT devices which suffer from limited computation ability and power supply. In this paper, we proposed an opportunistic vehicle-based computing framework for IoT offloading. In our approach, IoT devices can directly communicate with vehicles and utilize their idle resources. We proposed a selection strategy based on the SINR and the time needed for offloading data and receiving the results. Our proposed framework presents a reliable and sustainable replacement to the traditional paradigm and the surveyed related work. The selection mechanism studied in this work along with the framework that is put in place shows promising results in terms of usability. This framework is adaptable and can be used for different systems not necessarily RSUs but also can be used by other stationary fog nodes to share the load. A possible future work would be to consider the network of MCNs as a database with an interface that would permit submitting tasks to it without targeting particular vehicles or worrying about the details of the association. Such a virtual database could be queried for the status of the submitted jobs, and even for updating or deleting jobs. We also plan in our future work to study the best clustering mechanism for

our approach by taking into consideration latency, throughput, and interference. Moreover, we would also want to explore securing the vehicular network communication using Ethereum and smart contracts, similar to what was done in [45] to secure smart grids.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work was supported in part by a grant from the American University of Beirut Research Board (URB), Beirut, Lebanon (AUB-URB/2021).

References

- [1] S. Nellis, Qualcomm launches autonomous driving computer, aiming to hit roads by 2023, 2020, URL <https://www.reuters.com/article/us-tech-ces-qualcomm-idUSKBN1Z51YH>.
- [2] S. Yi, C. Li, Q. Li, A survey of fog computing: concepts, applications and issues, in: Proceedings of the 2015 Workshop on Mobile Big Data, 2015, pp. 37–42.
- [3] A. Boukerche, E. Robson, Vehicular cloud computing: Architectures, applications, and mobility, *Comput. Netw.* 135 (2018) 171–189.
- [4] K. Mershad, H. Artail, Finding a STAR in a vehicular cloud, *IEEE Intell. Transp. Syst. Mag.* 5 (2) (2013) 55–68.
- [5] A. Zekri, W. Jia, Heterogeneous vehicular communications: A comprehensive study, *Ad Hoc Netw.* 75 (2018) 52–79.
- [6] D. Zheng, A. Wu, Y. Zhang, Q. Zhao, Efficient and privacy-preserving medical data sharing in Internet of Things with limited computing power, *IEEE Access* 6 (2018) 28019–28027.
- [7] R.O. Aburukba, M. AliKarrar, T. Landolsi, K. El-Fakih, Scheduling Internet of Things requests to minimize latency in hybrid Fog-Cloud computing, *Future Gener. Comput. Syst.* 111 (2020) 539–551.
- [8] J. Zaugg, China's data centers emit as much carbon as 21 million cars, 2019, URL <https://edition.cnn.com/2019/09/10/asia/china-data-center-carbon-emissions-intl-hnk/index.html>.
- [9] R. Bryce, How google powers its 'monopoly' with enough electricity for entire countries, 2020, URL <https://www.forbes.com/sites/robertbryce/2020/10/21/googles-dominance-is-fueled-by-zambia-size-amounts-of-electricity/?sh=1c81b31968c9>.
- [10] Efficiency – Data centers. URL <https://www.google.com/about/datacenters/efficiency/#>.
- [11] F. Jalali, K. Hinton, R. Ayre, T. Alpcan, R.S. Tucker, Fog computing may help to save energy in cloud computing, *IEEE J. Sel. Areas Commun.* 34 (5) (2016) 1728–1739.
- [12] M. Gerla, Vehicular cloud computing, in: 2012 the 11th Annual Mediterranean Ad Hoc Networking Workshop (Med-Hoc-Net), IEEE, 2012, pp. 152–155.
- [13] M. Whaiduzzaman, M. Sookhak, A. Gani, R. Buyya, A survey on vehicular cloud computing, *J. Netw. Comput. Appl.* 40 (2014) 325–344.
- [14] V. Soto, R.E. De Grande, A. Boukerche, REPRO: time-constrained data retrieval for edge offloading in vehicular clouds, in: Proceedings of the 14th ACM Symposium on Performance Evaluation of Wireless Ad Hoc, Sensor, & Ubiquitous Networks, 2017, pp. 47–54.
- [15] J. Feng, Z. Liu, C. Wu, Y. Ji, Mobile edge computing for the Internet of vehicles: Offloading framework and job scheduling, *IEEE Veh. Technol. Mag.* 14 (1) (2018) 28–36.
- [16] B. Li, Y. Pei, H. Wu, Z. Liu, H. Liu, Computation offloading management for vehicular ad hoc cloud, in: International Conference on Algorithms and Architectures for Parallel Processing, Springer, 2014, pp. 728–739.
- [17] S. Bitam, A. Mellouk, S. Zeadally, VANET-cloud: a generic cloud computing model for vehicular Ad Hoc networks, *IEEE Wirel. Commun.* 22 (1) (2015) 96–102.
- [18] S.A. Kazmi, T.N. Dang, I. Yaqoob, A. Manzoor, R. Hussain, A. Khan, C.S. Hong, K. Salah, A novel contract theory-based incentive mechanism for cooperative task-offloading in electrical vehicular networks, *IEEE Trans. Intell. Transp. Syst.* (2021).
- [19] L. Vigneri, T. Spyropoulos, C. Barakat, Storage on wheels: Offloading popular contents through a vehicular cloud, in: 2016 IEEE 17th International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM), IEEE, 2016, pp. 1–9.
- [20] N.S. Safa, C. Maple, M. Haghparast, T. Watson, M. Dianati, An opportunistic resource management model to overcome resource-constraint in the Internet of Things, *Concurr. Comput.: Pract. Exper.* 31 (8) (2019) e5014.

- [21] N. Fernando, S.W. Loke, I. Avazpour, F.-F. Chen, A.B. Abkenar, A. Ibrahim, Opportunistic fog for IoT: Challenges and opportunities, *IEEE Internet Things J.* 6 (5) (2019) 8897–8910.
- [22] R.-I. Ciobanu, C. Dobre, M. Bălănescu, G. Suciuc, Data and task offloading in collaborative mobile fog-based networks, *IEEE Access* 7 (2019) 104405–104422.
- [23] A. Shakarami, M. Ghobaei-Arani, A. Shahidinejad, A survey on the computation offloading approaches in mobile edge computing: A machine learning-based perspective, *Comput. Netw.* (2020) 107496.
- [24] A. Shakarami, A. Shahidinejad, M. Ghobaei-Arani, A review on the computation offloading approaches in mobile edge computing: A game-theoretic perspective, *Softw. - Pract. Exp.* 50 (9) (2020) 1719–1759.
- [25] A. Shakarami, M. Ghobaei-Arani, M. Masdari, M. Hosseinzadeh, A survey on the computation offloading approaches in mobile edge/cloud computing environment: a stochastic-based perspective, *J. Grid Comput.* 18 (4) (2020) 639–671.
- [26] A. Shahidinejad, M. Ghobaei-Arani, Joint computation offloading and resource provisioning for edge-cloud computing environment: A machine learning-based approach, *Softw. - Pract. Exp.* 50 (12) (2020) 2212–2230.
- [27] A. Shakarami, A. Shahidinejad, M. Ghobaei-Arani, An autonomous computation offloading strategy in Mobile Edge Computing: A deep learning-based hybrid approach, *J. Netw. Comput. Appl.* 178 (2021) 102974.
- [28] F. Jazayeri, A. Shahidinejad, M. Ghobaei-Arani, Autonomous computation offloading and auto-scaling the in the mobile fog computing: a deep reinforcement learning-based approach, *J. Ambient Intell. Humaniz. Comput.* 12 (8) (2021) 8265–8284.
- [29] K. Zhang, Y. Mao, S. Leng, A. Vinel, Y. Zhang, Delay constrained offloading for mobile edge computing in cloud-enabled vehicular networks, in: 2016 8th International Workshop on Resilient Networks Design and Modeling (RNDM), IEEE, 2016, pp. 288–294.
- [30] C. Zhu, G. Pastor, Y. Xiao, Y. Li, A. Ylae-Jaeaeski, Fog following me: Latency and quality balanced task allocation in vehicular fog computing, in: 2018 15th Annual IEEE International Conference on Sensing, Communication, and Networking (SECON), IEEE, 2018, pp. 1–9.
- [31] L. Xu, R. Collier, G.M. O'Hare, A survey of clustering techniques in WSNs and consideration of the challenges of applying such to 5G IoT scenarios, *IEEE Internet Things J.* 4 (5) (2017) 1229–1249.
- [32] Y. Chen, P. Martins, L. Decreusefond, F. Yan, X. Lagrange, Stochastic analysis of a cellular network with mobile relays, in: 2014 IEEE Global Communications Conference, IEEE, 2014, pp. 4758–4763.
- [33] M. Charaf, H. Artail, Y. Nasser, Mobile relay node in public transportation for serving outside LTE cell edge users, in: 2015 IEEE 11th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob), IEEE, 2015, pp. 59–66.
- [34] K. Sariaeddine, M. Charaf, M. Ayad, H. Artail, A framework for mobile relay node selection for serving outdoor cell edge users, *Comput. Netw.* 178 (2020) 107359.
- [35] nsnam, Ns-3 simulator, 2019, <https://nsnam.org>. URL <https://nsnam.org>.
- [36] Eclipse, Sumo traffic simulator, 2019, <https://www.eclipse.org/sumo>. URL <https://www.eclipse.org/sumo>.
- [37] J. Benin, M. Nowatowski, H. Owen, Vehicular network simulation propagation loss model parameter standardization in ns-3 and beyond, in: 2012 Proceedings of IEEE Southeastcon, IEEE, 2012, pp. 1–5.
- [38] Z. Tong, H. Lu, M. Haenggi, C. Poellabauer, A stochastic geometry approach to the modeling of DSRC for vehicular safety communication, *IEEE Trans. Intell. Transp. Syst.* 17 (5) (2016) 1448–1458.
- [39] P. Li, J. Li, L. Nie, B. Wang, Research and application of zigbee protocol stack, in: 2010 International Conference on Measuring Technology and Mechatronics Automation, Vol. 2, IEEE, 2010, pp. 1031–1034.
- [40] X. Jiang, X. Cao, D.H. Du, Multihop transmission and retransmission measurement of real-time video streaming over DSRC devices, in: Proceeding of IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks 2014, IEEE, 2014, pp. 1–9.
- [41] R. Boutaba, M.A. Salahuddin, N. Limam, S. Ayoubi, N. Shahriar, F. Estrada-Solano, O.M. Caicedo, A comprehensive survey on machine learning for networking: evolution, applications and research opportunities, *J. Internet Serv. Appl.* 9 (1) (2018) 16.
- [42] J.A. Hatem, A.R. Dhaini, S. Elbassuoni, Deep learning-based dynamic bandwidth allocation for future optical access networks, *IEEE Access* 7 (2019) 97307–97318.
- [43] T.R. Andel, A. Yasinsac, On the credibility of manet simulations, *Computer* 39 (7) (2006) 48–54.
- [44] H. Artail, H. Safa, K. Mershad, Z. Abou-Atme, N. Sulieman, COACS: A cooperative and adaptive caching system for MANETs, *IEEE Trans. Mob. Comput.* 7 (8) (2008) 961–977.
- [45] R. Akhras, W. El-Hajj, M. Majdalani, H. Hajj, R. Jabr, K. Shaban, Securing smart grid communication using ethereum smart contracts, in: 2020 International Wireless Communications and Mobile Computing (IWCMC), IEEE, 2020, pp. 1672–1678.



Khaled Sariaeddine is a Ph.D. student at Concordia University. He was an assistant instructor in the Department of Computer Science at the American University of Beirut (AUB), where he also obtained his bachelor's and master's degree. His research interests include Vehicular Ad hoc Networks (VANETs), Fog, Edge, and Cloud Computing. Moreover, his most recent research interests extend to malware and ransomware detection/prevention, along with security of the electric vehicle ecosystem.



Hassan Artail is a Professor in the ECE department at the American University of Beirut (AUB), doing research in mobile computing, vehicular networking, and IoT. Prior to joining AUB, Dr. Artail developed systems for vehicle testing at Chrysler Corp., Michigan. He obtained a B.S. with high distinction and an M.S. in ECE from the University of Detroit in 1985 and 1986 respectively, and a Ph.D. from Wayne State University in 1999. Since joining AUB, Dr. Artail published over 230 articles in reputable journals and top conferences. He obtained several awards, including the Research Excellence Award from the Lebanese National Council for Scientific Research in July 2012, and the Career Excellence in Scientific Research Award from the Lebanese Association for the Advancement of Science in April 2017. He is an Associate Editor of the IEEE Transactions on Mobile Computing, and an Area Editor of the Elsevier Computer Networks journals.



Prof. Safa received a B.Sc. in Applied Mathematics and Computer Science in 1991 from Lebanese university, Beirut, Lebanon, M.Sc. in Computer Science in 1996 from University of Quebec at Montreal (UQAM) and a Ph.D. in Electrical and Computer Engineering in 2001 from Ecole Polytechnique de Montreal, Quebec, Canada. In 2000, he joined ADC Newnet in Shelton, Connecticut where he worked on designing and developing networking and system software. In 2003, Prof. Safa joined the American University of Beirut as where he is currently a professor and Chair of Computer Science. Prof. Safa also associated with the Mobile Computing and Networking Research Laboratory (LARIM), Ecole Polytechnique de Montreal, Montreal, Canada. His research interests are mainly in networking and mobile computing. Prof. Safa also served or is currently serving on the editorial board of many journals and organizing body and technical program committees of many international conferences.