IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 28 May 2024, accepted 20 June 2024, date of publication 26 June 2024, date of current version 3 July 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3419156

RESEARCH ARTICLE

Edge-Server Workload Characterization in Vehicular Computation Offloading: Semantics and Empirical Analysis

BAEKGYU KIM[1], (Member, IEEE), AND DEEPAK GANGADHARAN[2], (Member, IEEE)
[1]Department of Electrical Engineering and Computer Science, DGIST, Daegu 42988, South Korea
[2]Computer Systems Group, IIIT Hyderabad, Hyderabad 500032, India

Corresponding author: Baekgyu Kim (bkim@dgist.ac.kr)

This work was supported in part by the Institute of Information and Communications Technology Planning and Evaluation (IITP) Grant funded by the Korea Government [Ministry of Science and ICT (MSIT)] (Development of Network Load Balancing Techniques Based on Multiple Communication, Computing, Storage Resources) under Grant 2022-0-01053; and in part by the National Research Foundation of Korea (NRF) under Grant 2022R1C1C1003123.

ABSTRACT Edge server-assisted computation offloading enables vehicles to leverage server compute resources to deliver connected services, overcoming the limitations of onboard resources. Understanding the compute workloads of edge servers is crucial for effective resource management and scheduling, yet this task is challenging due to the complex interplay of factors such as vehicle mobility and computation offloading patterns. To address this, we propose an empirical analysis framework that systematically characterizes the compute workloads of edge servers. We begin by formalizing the relationships among three key aspects: local load (generated by vehicles), composite load (imposed on edge servers), and traffic flow (vehicle mobility patterns). Our framework then uses models of the local load and traffic flow as inputs to generate the composite loads on edge servers. Experiments were conducted by injecting between 600 and 5,000 vehicles per hour in two distinct geographical areas, New York City and Tampa. We provide a quantitative analysis demonstrating how the composite loads on edge servers vary with changes in traffic flows, geographical areas, and offloading patterns.

INDEX TERMS Connected vehicles, computing workload, edge servers, computation offloading.

I. INTRODUCTION

An edge server is designed to process computing workloads generated in a designated geographical region. Compared to an enterprise cloud server, the edge server is equipped with relatively smaller compute resource capacity, such as CPU/GPU and storage (e.g., memory), enough to process a regional workload. However, its geographical proximity with end devices (e.g., vehicles, drones, smartphones) provides several benefits, such as lower network latency or localizing privacy-sensitive data within the region. In the automotive domain, the edge server is becoming popular as an infrastructure component that supports V2I (Vehicle-to-Infrastructure) communication-based services [41].

One usage of the edge server in the V2I service is computation offloading. To overcome the vehicle's limited compute resources, some compute workloads are offloaded to proximate edge servers with larger compute capabilities. For example, a vehicle may periodically send camera images to an edge server that can update the distributable local map using the server's computing capability instead of using the vehicle's. Diverse V2I services that utilize the edge server are proposed to increase road safety or driving convenience by industrial consortium [49], [50], [55] and government agency [51], [52].

As the applicability of the edge server-assisted computation offloading expands, a higher volume of compute workload offloaded from vehicles will be imposed on the edge servers. Given that the edge servers' compute capability is comparatively smaller than that of data centers or cloud

The associate editor coordinating the review of this manuscript and approving it for publication was Ivan Wang-Hei Ho.

servers, the resources of the edge servers need to be intelligently managed; for example, one can distribute some offloading requests to other under-utilized servers, or one can extend the capacity of CPU or memory to meet a sudden surging workload at a particular time. Resource management can be more effectively designed once we understand the unique aspects of the target workloads.

There has been much workload characterization research and their applications in other domains such as cloud services or data centers [34]. Such workload characterization was used for performance evaluation, capacity planning, resource provisioning in conventional web workloads, online social networks, video services, mobile apps, and cloud computing infrastructures [26]. Based on the characterized workload, computation offloading research has been conducted to execute time-sensitive applications such as augmented reality (AR) and map-based navigations [28]. Those computation offloading works targeted various optimization objectives, such as minimizing execution delay [16], power consumption [13], reduction of financial cost [29].

However, characterizing the edge server's workload in the domain of the connected vehicle services is non-trivial and unique since it is collectively formed by many vehicle's offload requests that are intertwined in both temporal and spatial domains in a complex fashion. That is, each vehicle may have different patterns of offloading timings and sizes; the vehicles' mobility is also coupled with such offloading patterns, making it much harder to know the exact timing and size of the edge servers' compute workloads that are geographically distributed.

By extending our previous preliminary research [20], we propose the empirical analysis framework that can systematically characterize the edge server's workload, which is collectively formed by many computation offloads from vehicles. Firstly, we present the semantics of the two separate workload types that comprise V2I services. The *local load* (denoted as $R_c(t)$) is generated by a vehicle independently of its offloading decision, and some of the local loads are subject to the offloading to the edge servers; the *composite load* (denoted as $R_s(t)$) that is collectively formed by some local loads offloaded from vehicles within the edge server's coverage. Then, we provide the analytic workload characterization showing that a naïve composition of multiple local loads without considering the mobility patterns of the vehicles results in highly over-approximated composite loads.

Secondly, to abstract the vehicles' mobility patterns (denoted as $E(t)$), we introduce the three traffic flow variables, the *volume*, *density*, and *speed*, which are widely used to characterize macroscopic traffic in the intelligent transportation domain [56], [57], [58]. Then, we show the relationship between these traffic flow variables and the aforementioned workload types (*i.e.,* the local load and the composite load). This relationship is implemented as the synthetic data generator that can produce the edge server's composite loads using varying combinations of traffic flow variables and local loads.

Thirdly, we present the quantitative analysis based on the generated data to characterize the edge server's composite load, which is comprised of the local loads and the mobility patterns. Our findings show that, while the edge server's composite load takes a highly non-linear relationship with the factors above, there exist uniform patterns of the workloads among different edge servers that hold across different parameter combinations in terms of the aggregate load, the peak loads, and the load similarity. The analysis validates the generality of the findings by injecting 600 to 5,000 vehicles per hour in two very distinctive geographical areas, New York City and Tampa, where the connected pilot project of the U.S Department of Transportation (USDOT) is conducted [51].

We make the following contributions:

- Formalization of the edge server's composite workload based on the traffic flow variables and the computation offload patterns;
- RTC (Real-Time Calculus) curve-based workload characterization to show the gaps between the over-approximated and the actual composite loads;
- Implementation of the synthetic data generator that systematically generates a large realistic edge server's composite workload;
- Quantitative analysis to show the relationship among the traffic flows, the local workloads, and the edge servers' composite workloads.

Section II gives the background and the analysis objective. Section III presents the semantics of the workload and analytic characterization. The system model with the mobility pattern is given in Section IV. We show the analysis design in Section V, and present the empirical analysis result in Section VI. In Section VII, we discuss how our findings can be used for the ongoing V2I infrastructure design and deployment activities, then conclude in Section VIII.

II. OBJECTIVE OF EMPIRICAL ANALYSIS

A. BACKGROUND AND MOTIVATION

When it comes to the computing concept in designing automotive systems, the traditional engineering decision-making mainly focused on the capacity of in-vehicle computing resources (*e.g.,* ECU power), and their placement minimizing the side effects, such as the vehicle dynamics and heating. However, the computing landscape has significantly changed by many automakers that started to equip vehicles with connectivity. Such connectivity opens a new way of performing computation, especially allowing the vehicles to *offload* some part of the computation to the external infrastructures (*e.g.,* cloud servers) with higher computing capacity. For example, USDOT (United States Department of Transportation) awarded more than $45 million for the pilot study of using the infrastructure with connected vehicle technologies to enhance road safety in three sites in the

United States [51], [52]. In addition, multiple industrial consortiums, such as AECC [49], ETSI [50], 5GAA [55], proposed a way to architect the computing infrastructure to support new revenue sources created by the connected vehicle services.

However, it is challenging to design such a computation offloading concept due to the sheer number of connected vehicles; for instance, it is estimated that the number of connected vehicles will grow to about 100 million globally [49]. There is a consensus that one centralized server, such as a centralized data center, cannot meet the diverse requirements of all computation requests generated from such a large number of vehicles [41]. Instead, multiple regional servers, called edge servers in our work, need to distribute the regional workloads of vehicles so that some requests are processed in a particular edge server while other requests are forwarded to other servers when the former server's resources are being over-utilized. To make such a decision, it is essential to precisely characterize the workloads engaged in the computation offloading, such as how much the workloads are generated from vehicles and how their patterns differ across the geographical regions where different servers are located.

B. OBJECTIVE

Fig. 1 illustrates the computation offloading concept. Suppose a vehicle generates a series of workloads according to some patterns that need to be processed by some compute resources as it drives (*e.g.,* generating road images to synthesize the local HD map); we call it the *local load* (illustrated as $R_c^1, \ldots, R_c^4$ in Fig. 1). Such compute resources may reside inside the vehicle or in remote edge servers (servers), and a vehicle can selectively determine those resources to process the workload. When no servers are available in its vicinity, the local load should be processed by the vehicle's local compute resources (*e.g.,*, in-vehicle ECUs). On the other hand, when a vehicle drives within a geographic area where a server is installed, the vehicle is eligible to offload its local load to the server. Since multiple vehicles may drive in the area simultaneously, the server needs to process the workloads that are collectively formed from those vehicles; we call it the *composite load* (illustrated as $R_s^1(t), \ldots, R_s^3(t)$ in Fig. 1).

We use empirical analysis to characterize the server's composite load in various traffic flow scenarios. More precisely, the analysis objective is given as follows.

$$R_s^i(t) = f(R_c(t), E^i(t)), \tag{1}$$

where $R_c(t)$ is the local load generated by a set of vehicles $\{R_c^1(t), R_c^2(t), \ldots, R_c^m(t)\}$; $E^i(t)$ is the traffic flow occurring in the area covered by a server i; $R_s^i(t)$ is the composite load imposed on a server i. The analysis objective is to find insight into the complex function f that determines the composite load from the local load and the traffic flows.

However, such characterization is challenging due to its unique aspect originating from the (vehicular) traffic flow.

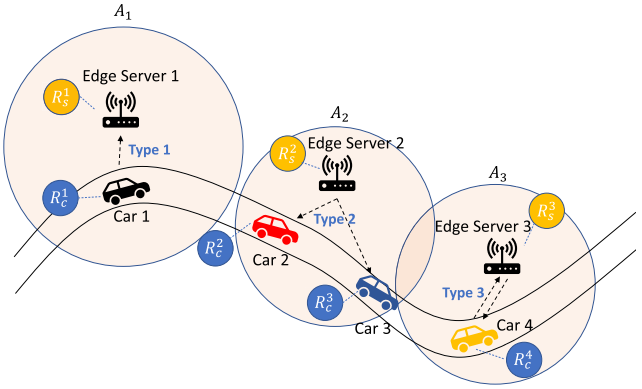


FIGURE 1. Illustration of the local workload and the composite workload.

For example, as a vehicle drives faster, the road images need to be sent more frequently to the servers in order to maintain the required coverage of HD maps; conversely, as a vehicle significantly decelerates, more data needs to be sent to the servers to indicate potential road safety hazards such as accidents or construction sites. Furthermore, when multiple vehicles generate their local loads together, the composite load's magnitude and growing rate are formed differently depending on the traffic flow characteristics.

C. RELATED WORK

Workload characterization has been extensively studied in other highly dynamic environments such as cloud data centers [34]. For example, it has been used for conventional web workloads, online social networks, video services, mobile apps, and cloud computing infrastructures [26]. Despite the extensive literature on workload characterization, it has yet to be studied in the context of vehicular networks, where the complex relationship of workload with traffic flow parameters needs to be considered.

Computation offloading has also been significantly studied to provide resource-constrained devices with computing-intensive services by distributing the computational burdens to other resource-rich devices via communication [33]. The applied domain includes augmented reality (AR), map-based navigation [28]; code partitioning in mobile cloud computing [18]. Various types of optimization objectives in computation offloading were considered, such as execution delay [3], [6], [16], [17], [19], [40], power or energy consumption [6], [13], [31], [32]. While these works consider computation offloading techniques in the context of a mobile device, they lack consideration of a unique aspect of the vehicle flow model, which is essential to understanding the composite load on a server. These works also do not consider the modeling of local computation workloads and the derivation of composite workloads based on traffic flow parameters.

RSU (Road-Side-Unit) placement is a problem domain that determines the distance among RSUs and their locations to enhance the communication coverages and distribute their

compute workloads. In [36], the authors propose an RSU deployment strategy considering a 2-D road network and finite RSU capacity. A density-based RSU deployment strategy is explored in [24], where the RSUs are deployed using an inverse proportion to the expected density of vehicles. The capital and operating expenditures are minimized for RSU deployment [37]. Although these works on RSU placement consider coverage, road topology, and cost, the dependency on their effect on the workload models has yet to be explored.

Analytic Analysis of Workloads: There has been extensive work to characterize the workloads in the real-time systems domain. In particular, the Real-Time Calculus (RTC) [4], [60] abstracts an event stream (*i.e.,* a workload) using the arrival curve that expresses the maximum and minimum number of event occurrences within any interval, which can then be used to analyze the delay and backlogs. The RTC works have been extended into many forms. Moy studied a way to compute tighter arrival curves, proposing a definition of causality [35] Kunzil et al. proposed a hybrid approach to gain the benefits of a simulation approach (*e.g.,* sliding window) and an analytic approach (*e.g.,* RTC) [7]. Salem et al. proposed a variation of arrival curves, called inter-arrival curves, that count the number of event occurrences within an interval since the occurrence of a particular event of interest [14]. Carvajal et al. proposed constructing arrival curves empirically to find the upper and lower bounds of the event occurrences in intervals [27]. Li et al. proposed a way to characterize the delay upper bound better when a system consists of multiple asynchronous flows by introducing offsets to the Network Calculus [9]. Beyond these works, there are other RTC works that try to compute tighter bounds of the input and output curves [10], [15], and to improve the analysis efficiency [5]. In addition, to overcome the limited expressiveness of RTC curves, there were several works to interface RTC with other formalisms such as Lustre [2], Timed Automata [8] and AADL [39].

III. SEMANTICS OF COMPUTATION OFFLOADING

A. COMPUTATION OFFLOADING SEMANTICS

When a vehicle requests a series of computations to be offloaded to the servers on the move, it should meet the requirements of the connected services (*e.g.,* latency requirements). We present the two representative offload patterns inspired by the service requirements of the use cases extensively studied in the literature [41], [52].

1) PERIODIC OFFLOADING

Fig. 2 and Fig. 3 illustrate how two clients' workloads produce a server's composite workload in two different offloading patterns (Type-A and Type-B). The figures have three timelines that indicate the event occurrence timings (upward arrows) for Client 1 (first row), Client 2 (second row), and Server A (third row). When multiple events occur simultaneously, we use multiple overlapped arrows to explicit

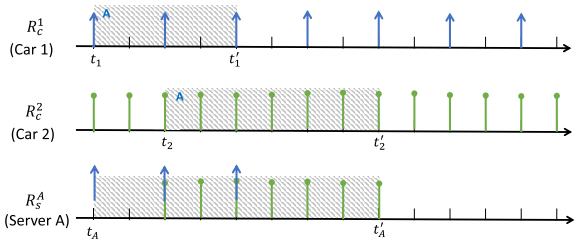


FIGURE 2. Example of Clients' and Server's Workloads (Type A). The active periods are highlighted in gray.

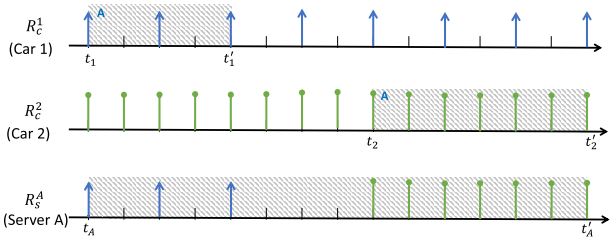


FIGURE 3. Example of Clients' and Server's Workloads (Type B).

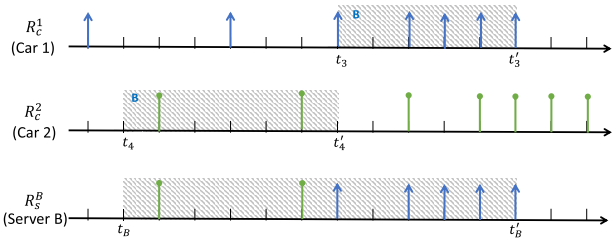


FIGURE 4. Example of Clients' and Server's Workloads (Speed-Dependent Workload).

the number of event occurrences; for example, Server A has two simultaneous event occurrences at times 3 and 5 in Fig. 2.

Examples of client events are requests for processing a graphical image transformation or invoking a machine learning training task for object detection. In this example, we assume Server A processes such events offloaded by Client 1 and 2 only but does not generate its own event. Some of those client events (first and second rows) can be offloaded to Server A (third row), that is deemed to be more appropriate to process them.[1] To scope the offloading period, we use the gray boxes with an annotation of the server identifier. For example, the first three events of Client 1 within the gray box in Fig. 2 and 3 are to be offloaded to Server A.

Even though Client 1 and 2 generate the event streams with exactly the same periods (period 2 for Client 1 and period 1 for Client 2) in both Fig. 2 and 3, the offloading timing of Client 2 is different; Client 2 starts offloading at time 3 and finish at time 9 in Fig. 2, while the offloading starts at time 9 and finishes at time 15 in Fig. 3. We call such an interval an *active period*. Consequently, Server A experiences different

[1]Note that such offloading decisions are made based on application dependent criteria such as monetary cost, energy consumption or latency, but we do not discuss them here as it is out of the scope of this paper.

patterns of the composite workloads in the two cases; that is, the composite workload in Fig. 2 is denser than that of Fig. 3, and one reason is that the clients' offloading periods in the former case overlapped and closer each other, while the latter is not.

2) SPEED-DEPENDENT OFFLOADING

Fig. 4 illustrates the speed-dependent offload pattern. A vehicle produces different sizes of compute loads at different frequencies in proportion to its speed. As a vehicle moves faster, larger (or smaller) local loads are generated more (or less) frequently. In the upper two timelines, for example, as vehicles 1 and 2 accelerate, their local loads are generated more frequently, gradually reducing their interval from four time-unit to one time-unit. Vehicle 1 enters the Server-B area when its local load is generated more frequently, while vehicle 2 enters the Server-B area when its local load is generated less frequently; the resulting Server-B composite load is shown in the bottom timeline.

B. ANALYTICAL GAPS BETWEEN OVER-APPROXIMATED AND ACTUAL COMPOSITE WORKLOADS

One easy way to approximate the upper-bound of the server's composite workload ($R_s(t)$) is to add up the maximum local workload of each vehicle ($R_c(t)$), ignoring their arrival/departure times at/from the server (*e.g.,* t_1 and t_1' in Fig. 2). That is, all vehicles are assumed to stay within the server's region all the time, expecting all local workloads to be offloaded to the server. However, such a naïve assumption tends to significantly overapproximate the composite workload. In this section, we analytically characterize the overapproximated composite load with the arrival curves used in Real-Time Calculus (RTC).

Real-Time Calculus (RTC) is one promising formal framework to characterize various types of workloads and analyze their impact on system performances [4], [60]. Extending the concept of Network Calculus [25], RTC abstracts an event stream (*i.e.,* a workload) using the arrival curve that expresses the maximum and minimum number of event occurrences within any interval Δ; the arrival curve can then be integrated with a service model expressing the amount of resources available and a processing model expressing the semantics of processing input streams into output streams; several performance criteria can then be formally analyzed such as delays or backlogs.

We use the following definition of the arrival curve typically used in RTC works [5], [15], [60].

Definition 1 (Arrival Curve): Let $R[s, t)$ denote the total number of events that arrived in the interval $[s, t)$. The corresponding upper and lower arrival curves are denoted as α^l and α^u, respectively, and satisfy the following condition:

$$\forall\, s < t, \alpha^l(t-s) \leq R[s,t) \leq \alpha^u(t-s),$$

where $\alpha^l(0) = \alpha^u(0) = 0$.

We give examples of the arrival curves using the event streams of Fig. 2 and Fig. 3 as follows:

Example 1 (Arrival Curves of Client 1 and 2): Consider two periodic event streams from Client 1 (Period 2) and Client 2 (Period 1) in Fig. 2 and Fig. 3. Then, according to Definition 1, the dotted green-lines in Fig. 5a and Fig. 5b express the arrival curves of Client 1 and 2, respectively. Note that we plotted the interval only up to 12 in the x-axis due to the space limit, but the periodic patterns of the curves can be further stretched out to the infinite interval.

Given two arrival curves that represent the local loads, one can compute the upper/lower bounds of the composite load using Equation 2 [60].

$$\begin{aligned} \alpha^u_{OR} &= \alpha^u_1 + \alpha^u_2 \\ \alpha^l_{OR} &= \alpha^l_1 + \alpha^l_2 \end{aligned} \tag{2}$$

Now we show why this approach—summing up the upper and lower bound of the local loads without considering when each vehicle arrives at or departs from the server—is insufficient to characterize the server's composite load in the offloading scenarios.

Consider the offloading scenario in Fig. 3. In Fig. 5c, the orange solid lines are the composite arrival curve of Server A computed from the two client arrival curves in Fig. 5a and Fig. 5b via Equation 2. On the other hand, the green-dotted lines are the composite arrival curve computed directly over the server's event stream R^A_s using a sliding-window approach; note that we assume such server's *real* composite event streams are *unknown* a prior (also, this is our target to characterize), and their composite arrival curves are unknown.

Here are two issues in the bounds (*i.e.,* the orange-solid lines) computed by Equation 2:

- *The upper bound is too pessimistic*; in Fig. 3, the two offloadings from Client 1 and 2 never overlap. Hence, a naïve sum-up in Equation 2 results in an overly conservative upper-bound, as it is originally designed assuming that *all* events of the input streams are to appear in the composite workload, which is not the case in offloading where *partial* events are to appear.
- *The lower bound is inappropriate*; in Fig. 3, there is a period $[t_1', t_2)$ where no event occurs in the composite workload since no offloading is made either from Client 1 or 2. Hence, a naïve sum-up over the lower bounds in Equation 2 also results in an inappropriate lower bound.

Necessity of the Empirical Analysis: The analytic analysis shows that the composition of the local loads ($R_c(t)$) without explicitly considering their active periods significantly over-approximates the actual composite load ($R_s(t)$). Such an over-approximated composite load may allocate unnecessary compute resources, resulting in higher consumption of the monetary cost, energy, or storage of CPU or memory. Therefore, we need to explicitly consider (i) the length of the active period and (ii) the overlapped patterns of multiple active periods to characterize the composite load precisely.

A tighter approximation of the composite load can be obtained by explicitly modeling the active periods using

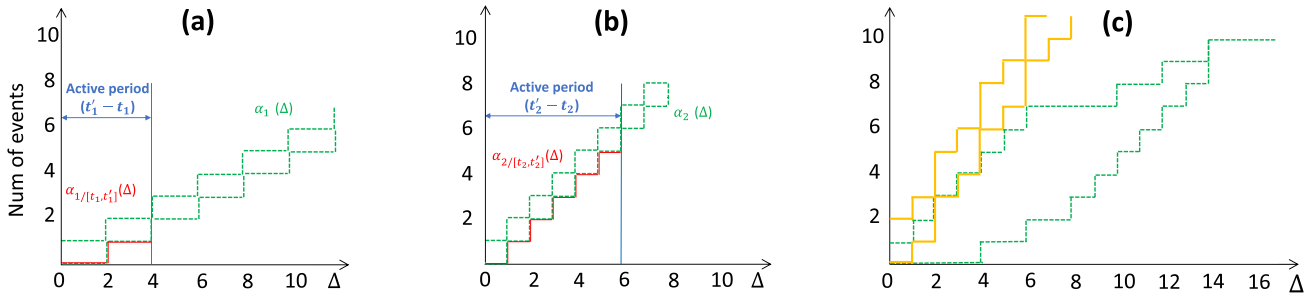


FIGURE 5. (a) Arrival curves of Client 1 in Fig. 2 (Green-dotted line: Original arrival curve, Red-solid line: Active arrival curve) (b) Arrival curves of Client 2 in Fig. 2 (c) Arrival curve of the composite workload of Server A in Fig. 3: the orange-solid lines are the *analytic* upper/lower bounds computed by Equation 2, while the green-dotted lines are the *real* upper/lower bounds computed directly over the server's event stream R_s^A.

stateful information. For example, Linh et al. extended the original RTC to accommodate state information more explicitly, and other techniques, such as the Timed Automata [23], LUSTRE [30], can construct such a stateful model and verify the property of the composite load. However, such active periods also depend on the traffic flow ($E^i(t)$); in general, this is difficult to represent and analyze in an analytic form, especially when a large number of vehicles go through complex traffic conditions. In the next section, we design the empirical analysis framework to better characterize the composite load by abstraction of the traffic flow and draw insights into the spatio-temporal aspects encoded in Equation 1.

IV. SYSTEM MODEL FOR TRAFFIC FLOWS

The macroscopic traffic flow variables are one of the widely used models that quantify the demand and the quality of the traffic service [47], [56], [57], [58]. In particular, we introduce the three traffic flow variables to characterize $E^i(t)$ in our empirical analysis: *Volume*, *Density*, and *Speed*.

A. VOLUME (V)

The volume is the variable that quantifies the number of vehicles passing through a specific area during a time interval, which is defined as:

$$V_i = \frac{n_i}{T}, \tag{3}$$

where i is the area identification (area ID), n_i is the number of vehicles passing the area i, T is the length of time period. The customary unit is [veh/hour] and [veh/min].

The volume observed in a geographical area covered by a server implies the collective demand of the offload requests generated by vehicles that pass through the area. Assuming every vehicle generates a certain amount of a local load, an area that experiences a higher volume would also experience a larger size of the server's composite load.

B. DENSITY (K)

The density is the variable that quantifies the number of vehicles occupying a certain area size at a particular moment, which is defined as:

$$K_i = \frac{m_i}{L}, \tag{4}$$

where L is the unit distance, m_i is the number of vehicles present in the area i. The customary unit is [veh/km] and [veh/m]. Unlike the volume whose definition relies on the time interval, the density measures the number of vehicles per distance. This subtle difference in their definitions sometimes results in a notable difference in the server's composite load.

Example 2 Impact of Volume and Density on the Composite Load: Consider two areas A and B that have road lengths of 50 meters and 100 meters, respectively; further, suppose that, both areas have the same volume of vehicles as 4 [veh/hour]; for every 15 minutes, there exists a vehicle that exits areas A and B; this means that each vehicle that passes through the areas have 15 minutes of time gap between each other in both areas. On the other hand, at any point in time, since there exist 4 vehicles present in both areas, A and B have different densities as 0.08 [veh/meters] and 0.04 [veh/meters], respectively. However, to achieve the volume and density above, the vehicles in B need to drive at a higher speed than those in A due to the difference in the driving distances (50 meters v.s. 100 meters). If the frequency and size of some offload requests increase proportionally to the vehicle speeds, more compute offload requests would be generated from the vehicles in B than A, leading to two different server's composite loads in A and B.

C. AVERAGE SPEED (S)

The average speed is the variable that measures the quality of the traffic services [42]. There are several definitions of the average speed, and one typically used definition is the space-mean speed, defined as follows:

$$S_i = \frac{L_i}{\frac{1}{n_i}\sum_j t_i^j}, \tag{5}$$

where n_i is the number of vehicles, L_i is the length of the area the vehicle travels, and t_i^j is the travel time for vehicle j to cross the area i. The customary unit is [km/hour].

Intuitively, this variable is the average speed of vehicles traveling a particular road segment during a specific period.

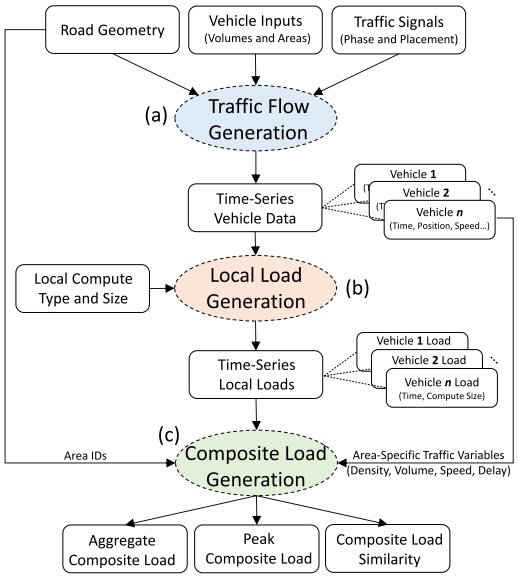


FIGURE 6. The implementation design of the data generator for the analysis of the composite load.

This definition is slightly different from the conventional time-mean speed that averages the speed of individual vehicles at a specific point. Despite such subtle differences, the space-mean speed is better matched to the property of the composite load occurring in a given road segment (not at a specific time). The average speed also plays a role in changing the server's composite load. For example, a higher average speed implies that the vehicles tend to pass through the area faster, making them stay there for a shorter period; this would sometimes reduce the server's composite load. In addition, if a specific offload request depends on the vehicle speed, the speed affects the server's composite load positively or negatively.

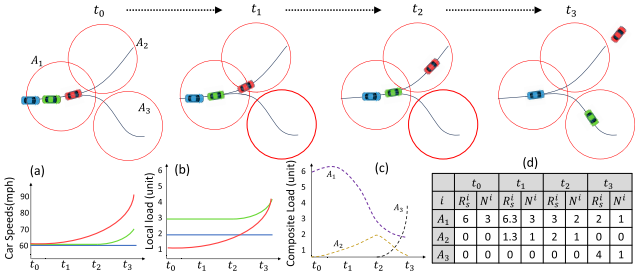
V. DESIGN OF COMPOSITE LOAD GENERATOR

The most challenging part in characterizing the function f in Equation 1 is that the two inputs, the local load ($R_c(t)$) and the traffic flow ($E^i(t)$), affect the server's composite load ($R_s^i(t)$) together at very different abstraction levels, and their interplay is complex. While many traffic flow analysis tools, such as VISSIM [22] or SUMO [11] support modeling $E^i(t)$, the interface with the computing load, such as $R_c(t)$ and ($E^i(t)$) is lacking. We introduce a layered analysis design in Fig. 6 that systematically constructs various types of $R_s^i(t)$ by superimposing $R_c(t)$ and $E^i(t)$.

A. DATA GENERATOR IMPLEMENTATION

In Fig. 6, the three colored oval boxes —(a), (b), (c) —represent the processes of characterizing the three time-series variables —$E^i(t)$, $R_c(t)$, $R_s^i(t)$—used in Equation 1.

In (a), we use the state-of-the-art tool, called VISSIM, to generate realistic traffic flows [22], [38] based on complex road geometries such as intersections, highways, merging



	t_0		t_1		t_2		t_3	
i	R_s^i	N^i	R_s^i	N^i	R_s^i	N^i	R_s^i	N^i
A_1	6	3	6.3	3	3	2	2	1
A_2	0	0	1.3	1	2	1	0	0
A_3	0	0	0	0	0	0	4	1

FIGURE 7. Illustration of the Load Composition Scheme; N^i indicates the number of vehicles present within the coverage of server i.

or exit; the example is shown in Fig. 8. For each selected road geometry, several road segments are chosen where a volume of vehicles is to be injected, and we call it the vehicle input. In addition, the traffic signals are placed at several locations to produce various patterns of stop-and-go traffic, which would result in a range of traffic congestion scenarios. The artifact produced by the traffic flow generation process is the time-series vehicle data that characterizes the changes in each vehicle's position, speed, or direction over time.

In (b), the local load of each vehicle is generated according to a particular offload pattern, such as the periodic or the speed-dependent offload, defined in Section III. Then, a collection of multiple local loads becomes the artifact produced by the local load generation. Note that the produced local loads (*i.e.,* $R_c(t)$) from (b) do not have the spatial semantics of which geographical area they are generated. In other words, from the perspective of the server's composite load (*i.e.,* $R_s^i(t)$), only a subset of each local load occurring within its coverage area is meaningful, so there exists a gap between $R_c(t)$ and $R_s^i(t)$.

In (c), the composite load generation process assigns such a server's spatial semantics to the temporal semantics of the local load. More specifically, the local load is categorized according to the geographical areas covered by each server where it occurs; each category of the local load is combined with other local loads in the same area to collectively form the server's composite load within a particular geographical region. Finally, the server's composite load is analyzed according to the metrics of the aggregate load, the peak loads, and the load similarity among different geographical areas.

We do not give the details of the computation complexity required to generate the composite load; the complexity grows proportionally to the size of the simulation and sampling interval, the number of vehicles, their states, and the offloading frequency of the local load.

B. LOAD COMPOSITION SCHEME

We explain the load composition scheme as to how (a), (b) are used to generate (c) in Fig. 6. Fig. 7 illustrates a driving scenario where three vehicles (red, green, blue) move at different speeds performing offloading in the areas (A_1, A_2, A_3) covered by three servers from time t_0 to t_3.

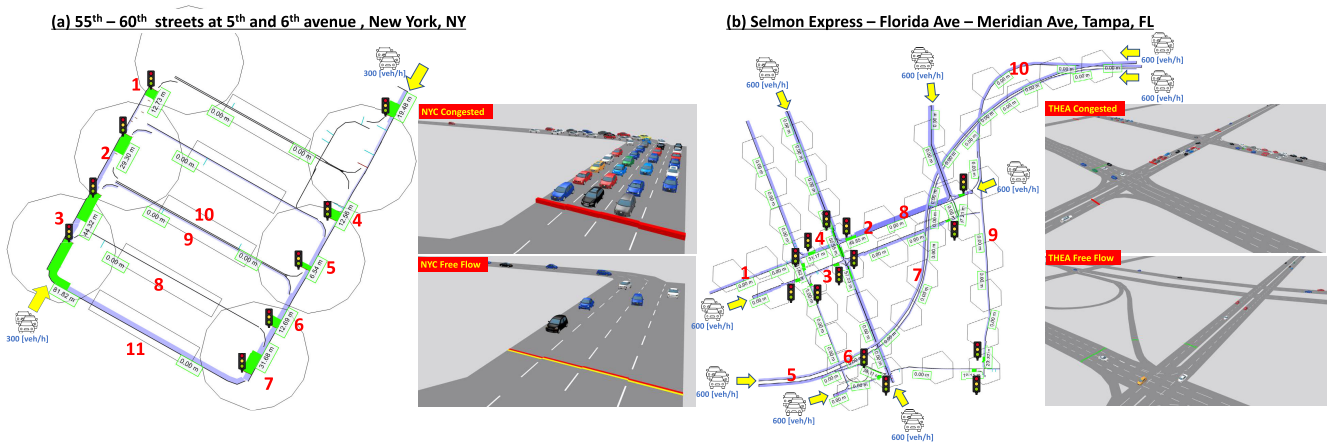


FIGURE 8. Selected geographical areas with the traffic flow visualizations: (a) New York City (NYC), (b) Tampa (THEA). Maximum queue length (Green), Traffic volume (Purple: The darker the purple color becomes, the higher traffic volume present); the selected locations of the traffic signals are indicated using the traffic signal icons; the locations of the vehicle input are indicated as the yellow arrows; the areas surrounded by the black lines are where the vehicle data is collected; the examples of the congested and free flow traffics for each area are given.

Suppose the red and green vehicles utilize speed-dependent offloading while the blue vehicle utilizes periodic offloading according to the offloading semantics in Section III. Fig. 7.a shows the speed changes of the three vehicles over time; the red and green vehicles drive gradually faster at different rates, while the blue vehicle drive at a constant speed. Fig. 7.b shows the size of the local loads generated by the three vehicles over time. Initially, for each offloading, the red, green, and blue send 1 unit, 2 units, and 3 units of computation loads to the servers, respectively. The local load of the blue vehicle remains constant in between t_0 and t_3 as the size does not change by the speed factor; the periodic load is characterized by Equation 6.

$$R_c^j(t) = \begin{cases} w_{prd} & \text{(Case 1-a)} \\ 0 & \text{(Case 1-b)}, \end{cases} \tag{6}$$

where j is the vehicle identification, and w_{prd} indicates the unit compute size of the periodic load. That is, the size of w_{prd} is generated for t such that $t = n \times T_j$ where $n \geq 1$ and T_j is the size of the period of vehicle j (Case 1-a). Otherwise, no load is generated for all t such that $t \neq n \times T_j$ (Case 1-b). As the size changes due to the speed factors, the local loads of the red and green also change. This relationship is the basis for the speed-dependent load, which is characterized by Equation 7.

$$R_c^j(t) = \begin{cases} w_{spd} \times v^j(t) & \text{(Case 2-a)} \\ \dfrac{w_{spd}}{v^j(t)} & \text{(Case 2-b)}, \end{cases} \tag{7}$$

where w_{spd} indicates the unit compute size of the speed-dependent load and $v^j(t)$ is the speed of the vehicle j at time t.

To explain the implication of Equation 7, consider two types of offloading scenarios related to the vehicle speed. One scenario is that, a higher volume of offloading requests should be generated as a vehicle moves *faster*. Consider a scenario in which a vehicle continuously takes road images using a camera that can be sent to servers for server-assisted route-planning (*i.e.,* a server decides a vehicle route based on the collected data). As a vehicle drives faster, more amount of data would be collected and sent to a server within a fixed time since the vehicle would experience more variety of situations within the same interval than it drives slower, in order to make the server-assisted route-planning decision.

Another scenario is that, a higher volume of offloading requests should be generated as a vehicle moves *slower*. Consider a scenario when a vehicle suddenly encounters a road accident; it may share additional data (*e.g.,* camera images of the accident scenes) for situational awareness with a server in proximity, which can instruct other following vehicles to reduce their speeds to avoid another collision; that is, such an unusual event caused by slow-down would require to trigger a larger amount of compute offloading. Hence, as a vehicle experiences a sudden deceleration (slow-down), the size of situation awareness data should be proportionally increased.

(Case 2-a) and (Case 2-b) characterize that the load increases and decreases proportionally to the vehicle speed to capture the above scenarios. Note that the offloading frequency changes depending on the speed is another orthogonal factor illustrated in Fig. 4; it is not explicitly shown in Equation 7.

Finally, the composite loads of the three servers ($R_s^1(t)$, $R_s^2(t)$, $R_s^3(t)$) are determined by the combination of (i) the unit sizes of the local loads, (ii) the position and speed of the vehicles, (iii) the servers where each local load; the composite load is characterized by Equation 8.

$$R_s^i(t) = \sum_j R_c^j(t), \tag{8}$$

where a vehicle j stays within the coverage area of a server j at time t. That is, the composite load of server j is the sum of all offloaded local loads generated by the vehicles within

the server's coverage. Fig. 7.c and d show how the composite loads change over time using Equation 8. Note that the red vehicle is located in two different coverage areas, A_1 and $A2$ at time t_1. This example assumes that the red vehicle offloads its computation to both servers. More offloading strategies exist when a vehicle stays in the coverage areas of more than one server, but we only discuss some strategies since they are out of the scope.

C. EXPERIMENT PARAMETER SELECTION

Many environmental parameters contribute to the traffic flow, which are summarized in Table 1 and 2. We introduce the factors focused on our analysis and the associated parameters.

Factor 1: Different traffic flow characteristics ($E^i(t)$) affect the server's composite loads differently.

A traffic flow determines the timing when a vehicle enters an area covered by a server and the duration of time the vehicle stays within it. Hence, such traffic flow characteristics would determine the timings of each offload request, which would consequently affect the server's composite load. Table 1 and 2 intend to capture the characteristics above in the form of parameters, and here is the intuition of the parameter choice.

The road geometry is one important factor that determines the traffic flow characteristics, and we selected two very different geographical areas, New York City (NYC) and Tampa (THEA), where the USDOT launched the connected pilot projects to support various connected services [51]. In particular, the two areas have very different geometric characteristics, as shown in Fig. 8, which results in diverse traffic flow variations, such as the number of intersections, traffic signals, highway entry or exit points, merging, one-way or two-way roads, possible vehicle routes.

The level of traffic congestion is another orthogonal factor that determines the traffic flow characteristics. For example, as the congestion level increases in a particular area, a group of vehicles tends to stay in the area longer, and the volume of the vehicles would be higher than in other non-congested areas. One way to control the congestion level is to place the traffic signals and adjust their signal phase timings. For example, for every 4-way intersection, four traffic signals can be installed that operate with different signal phases and cycle times, which would make congested or free flows.

Table 1 shows the selected parameters to reflect the aforementioned traffic congestion factors. For example, in NYC, the cycle time is 60 seconds, which implies the time it takes a traffic signal (*e.g.,* green, yellow, red) to display one complete sequence). The phase number is 4, which implies that four different traffic lights are grouped together to synchronize each other; for example, in the four-way intersections, four traffic lights are installed on each road segment of the intersection; their displays need to be synchronized to avoid the conflicts of the traffic flows in the intersections (*e.g.,* the north light should display green while the east, west, and south lights should display red). In addition, the phase interval is the time interval where each signal is continuously displayed before changing into another light.

On the other hand, Table 2 shows the effect of choosing different parameter combinations from Table 1. For example, by choosing different sizes of vehicle input, numbers of signalized intersections or signal phases of green lights, NYC and THEA have very different queue lengths, vehicle delays, stop delays as illustrated in the columns of *congested* and *free flow*. We refer to those two groups of parameter combinations as the *congested flow* and the *free flow* in the rest of the paper.

Vehicles are injected using the parameter of the vehicle input volume. For example, 300 and 600 vehicles are injected per hour in NYC and THEA. Note that this does not mean that there are exactly 300 and 600 vehicles only in each scenario since the parameters represent the rate of the vehicle injection but not the number of total vehicles; that is, a much larger number of vehicles may present in each simulation when sufficient simulation time passes. Initially, the number of vehicles in the scenario gradually increases according to the injection rate; after some time, some vehicles egress from the simulation, and some appear new. This implies that there is a point where a similar number of vehicles are present in the simulation scene, and our experiment is started when such a stabilized condition occurs.

Factor 2: Different computation offload patterns ($R_c(t)$) adopted by individual vehicles affect the server's composite loads differently.

Each vehicle may use a different offload pattern that would generate a different style of the local load. We consider the two types of offload patterns, the periodic offload (Fig. 2 and Fig. 3) or the speed-dependent offload (Fig. 4) in Section III. When coupled with the traffic flows explained in Factor 1, the local loads generated by the two offload patterns contribute to the server's composite load differently depending on how many vehicles are located, how fast they are moving, and how long each vehicle stays within the area. We selected the experiment parameters to show the impacts via case-by-case analysis.

VI. ANALYSIS RESULT

A. RAW DATA DISTRIBUTION

To illustrate how complex the function f in Equation 1 is, we take a partial dataset generated from the data generator in Fig. 6, and apply the curve-fitting to visualize the relationship of its input and output.

Fig. 9 shows how the server's composite load (output) changes under different traffic speeds (input), volumes (input), and road geometries (input), provided that all vehicles periodically offload their computations to the servers (input). As shown from the shapes of the four curves (a) - (d), their relationship is (i) highly non-linear, and their patterns are significantly different for (ii) each congestion level and (iii) each road geometry. Furthermore, when more other variables are considered together, their relationship will take a more complex form.

TABLE 1. The parameters for the traffic flow and the compute load generation in two different cities: new york city (NYC) and tampa (THEA).

Traffic/Load Parameter Name	NYC	THEA
Vehicle Input Volume	300 veh/hour	600 veh/hour
No. of Vehicle Input Places	2	9
No. of Servers	22	61
Server coverage range	[25, 300] meters	[30, 1000] meters
No. of Signalized Intersections	10	18
Traffic Signal Phase	Cycle Time (60 sec), Phase Numbers (4), Phase Interval ([10, 60] sec)	Cycle Time (60 sec), Phase Numbers ([2, 4]), Phase Interval ([10, 60] sec)
Desired vehicle speed	60 km/h	60 km/h
Simulation Travel Time	1000 sec	1000 sec
Compute load size per request	7 units	7 units
Period length (Periodic Request)	5 secs (fixed)	5 secs (fixed)
Period length (Speed-Dependent Request)	[0.1 * current speed] sec	[0.1 * current speed] sec
Speed-dependent load size	[0.1 * current speed] units	[0.1 * current speed] units

TABLE 2. The variables that represent the levels of the traffic congestion within the RSU coverage.

	NYC		THEA	
	Congested	Free Flow	Congested	Free Flow
Avg. queue Length (meters)	17.67	3.21	10.17	1.29
Avg. vehicles	37.9	117.60	153.4	152.96
Avg. vehicle delays (secs)	16.29	0.37	5.37	0.35
Avg. stop delay (secs)	10.57	0.03	3.25	0.01
Avg. stops	0.57	0.01	0.22	0.01

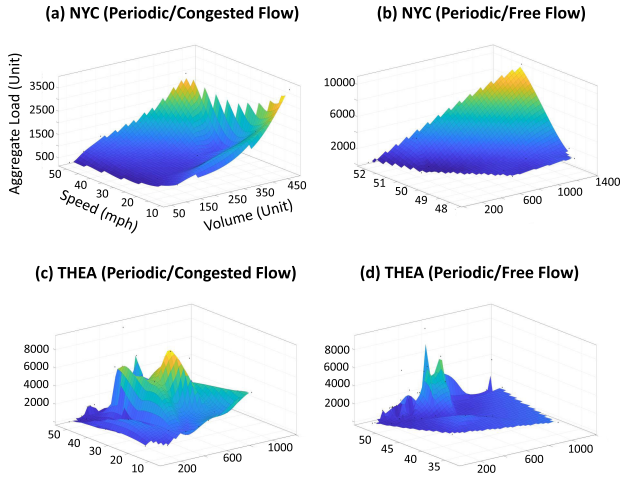


FIGURE 9. The curve fitting result that shows the relationship among the traffic flow variables (speed and volume), the local loads (periodic offload pattern), the server's composite load (aggregate load) in two different cities (NYC and THEA) under two different levels of traffic congestions (congested flow and free flow).

While such a data-driven approach like curve-fitting or machine learning can characterize the server's composite load, they do not provide intuitive explanations of how diverse factors affect the server's composite load. Our empirical analysis aims to extract a set of observations that can explicitly explain the relationship among the three parts of Equation 1 using the two metrics, the aggregate loads and the peak loads.

B. GEOGRAPHIC AGGREGATE COMPUTE LOADS

The geographic aggregate load is the cumulative sum of the local loads over time in a specific geographical region covered by a particular server, and this is defined as:

$$A_j(t) = \sum_{i,t'} R_c^i(t'), \tag{9}$$

where any vehicle i that stays in the server coverage at time t'.

This metric is used to gauge how large amount of offload requests from vehicles are accumulated over time in a particular area; if this value is too high in a particular area, the server's compute resources tend to be excessively utilized. For example, the server's energy consumption will become higher as more requests are centralized; for the offload requests that have to store some data on the server's data storage (*e.g.,* the image transfer of the HD map service), those (*e.g.,* memory or disks) will be filled to the limit early.

Fig. 10 and 11 show the geographic aggregate loads calculated for 11 areas in NYC ((a) - (d)) and 10 areas in THEA ((e) - (h)). The identification of each area is shown in Fig. 8. Each area is covered by one server, and the segment type indicates its coverage size; some servers cover smaller or larger areas than others. Each graph has upper and lower parts that compare the aggregate loads to different traffic flow variables. For (a), (b), (e), (f), the aggregate loads are compared to the traffic density (the upper part) and the traffic volume (the lower part). For (c), (d), (g), (h), the aggregate loads are compared to the traffic delay (the upper part) and the traffic average speed (the lower part). The slash-colored bars represent the normalized magnitude of the traffic flow variables (the green slash: density, the blue slash: volume, the magenta slash: delay, the gray slash: speed). The solid-blue bar represents the aggregate load collected by the local loads generated according to the periodic offload pattern only (periodic aggregate load); the solid-orange bar represents the aggregate load collected by the local loads generated according to the speed-dependent offload pattern only (speed-dependent aggregate load).

(Observation 1) If the sizes of the two areas are similar, the aggregate loads grow as the density, and the volume grow when both are positively co-related.

In many cases, the density and volume are positively co-related; as more vehicles pass through a particular area, the number of vehicles that occupy a unit area also increases. In this case, the aggregate loads increase as well. For example, in NYC case ((a) and (b)), for any two areas where the density and the volume are *positively* co-related and increasing, the amount of both types of aggregate loads (*i.e.,* the periodic and the speed-dependent) increase as well. The same growth pattern also happens in areas 5 - 10 in THEA cases ((e) and (f)) that have the same conditions.

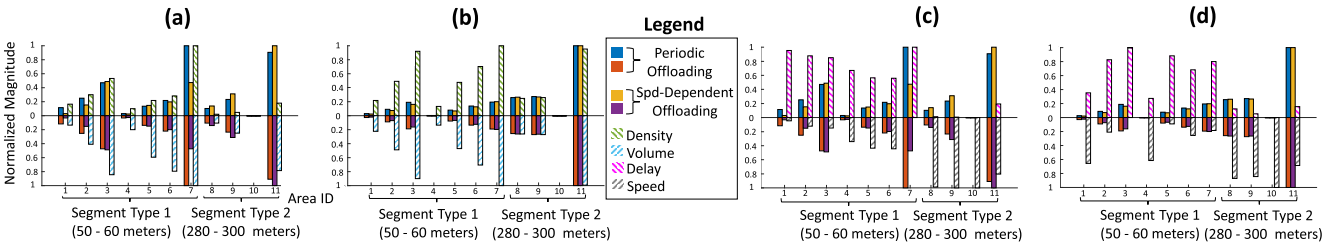


FIGURE 10. Aggregated compute loads in four different traffic flow scenarios in NYC. The comparison between the two types of compute loads (periodic pattern and speed-dependent pattern) and the macro traffic variables in (a) Density (Upper) and Volume (Lower) in Congested Flow of NYC (New York City), (b) Density (Upper) and Volume (Lower) in Free Flow of NYC, (c) Delay (Upper) and Speed (Lower) in Congested Flow of NYC, (d) Delay (Upper) and Speed (Lower) in Free Flow of NYC.

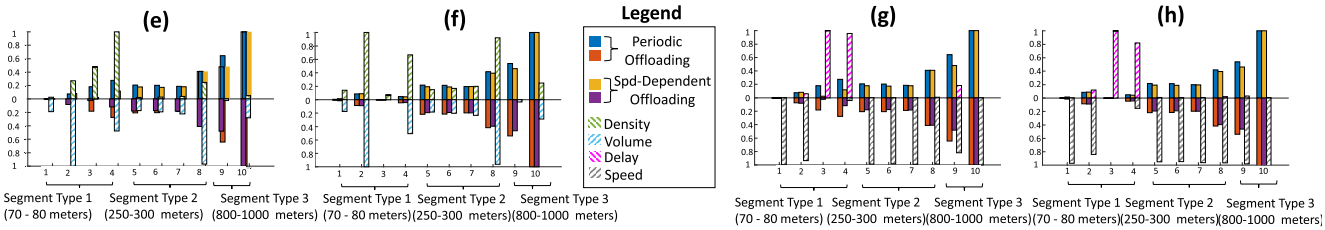


FIGURE 11. Aggregated compute loads in eight traffic flow scenarios in THEA. The meaning of (e), (f), (g), (h) are the same as (a), (b), (c), (d) in Fig. 10, respectively.

On the other hand, there are cases that, for two areas where the density and the volume are *negatively* co-related, the growth of the aggregate loads have different patterns (or even decreases) depending on the offload patterns of the local loads. For example, in THEA case (e), the density of area 3 is higher than area 2, while the volume of area 3 is smaller than area 2. In this case, the periodic aggregate load is positively correlated with the density (but *not* with volume). However, there is no clear relationship between the speed-dependent aggregate load and the density; the aggregate load decreases as the density increases.

This scenario (*i.e.,* the negative co-relationship between the density and the volume) happens when a small number of vehicles are waiting in the queue too frequently (*e.g.,* due to the red signal waiting) in a small area. In such a scenario, the growth of the speed-dependent load is penalized by the two factors simultaneously. Firstly, since there is a high chance for most vehicles to stop on the road, the average speed of the vehicles becomes significantly lower (or the delay of the vehicles is significantly higher) than in other areas, as shown in (g) in the area 2 and 3; this would consequently decrease the speed-dependent aggregate load. In addition, since there are fewer vehicles due to the low volume, the aggregate load becomes smaller than in other areas with a higher volume of vehicles. On the other hand, the periodic aggregate load is not affected by the vehicle speed, so the aggregate load grows as the density grows.

(Observation 2) If the density and the volume of the two areas are similar, the aggregate load grows as the area size becomes larger.

Even though the two areas have similar density and volume, this does not necessarily imply their accumulated loads are also similar, mainly due to the difference in their travel times. That is, vehicles that pass through a larger area tend to have a longer travel time, resulting in a larger number of offload requests, consequently leading to a higher aggregate load. For example, in NYC case (a), the sizes of the area 6 and 11 are different, but their density and volume are similar; the areas 7 and 11 in NYC case (b), and the area 2 and 8 in THEA case (e) and (f) have the similar characteristic. In these area pairs (*i.e.,* different area sizes with similar density and volume), a higher aggregate load is observed in a larger area than in a smaller area.

On the other hand, when the two areas have different sizes, densities, and volumes, it is not easy to draw a general conclusion about their effects on the aggregate load. For example, in NYC case (a), areas 3 and 5 have similar sizes, while area 9 is smaller than areas 3 and 5. In this case, area 3 has a much larger density, volume, and aggregate load than area 9. However, area 5 also has a much larger density and volume than area 9, but its aggregate load is smaller than area 9. That is, the changes in the density, the volume, and the area size altogether form a compound effect on the aggregate load, so depending on which one plays a dominant role, the relative aggregate load becomes different.

(Observation 3) The average speed differently affects the growth rate of the speed-dependent load and the periodic aggregate load.

Intuitively, since the speed-dependent local load is affected by the vehicle speed while the periodic load is not, their aggregate loads also have different patterns depending on the speed factor. For example, when the vehicle speed and the speed-dependent local load are positively co-related (*i.e.,* as the speed increases, the amount of the local load also increases), each vehicle produces a higher local load in those areas where the average speed is high.

However, when it comes to the aggregate load (*i.e.,* a composition of multiple local loads), this is not generally true since the speed factor also interplays with other traffic flow variables (*e.g.,* the density or the volume) affecting the aggregate load in a complex fashion.

In NYC case (a), areas 3 and 7 have positively co-related density and volume with a similar area size. Hence, it is expected to have a higher aggregate load in area 7. However, the periodic aggregate load increases in area 7, while the speed-dependent aggregate load stays the same between areas 3 and 7. This is because the average speed of area 7 is slightly lower than area 3 as shown in (c). This slight difference (*i.e.,* a negative factor to the aggregate load) penalizes the growth of the speed-dependent aggregate load despite a much larger increase in the density (*i.e.,* a positive factor to the aggregate load).

Similarly, in THEA case (e), the periodic aggregate load grows from area 2 to area 3 as the density increases; however, the speed-dependent aggregate load decreases from area 2 to 3 despite the density increase. On the other hand, the periodic and speed-dependent aggregate loads grow from area 2 to area 4 as the density increases. This is because, even though the average speeds of the area 3 and 4 are much smaller than area 2, their significance in contributing to the speed-dependent aggregate loads is different when coupled with the density factor; that is, the speed factor in the area 4 is less dominant than the area 3.

C. PEAK COMPUTE LOADS

The peak load is a notably large compute load sustained for a relatively shorter period than other normal loads. Handling such a peak load is important but challenging because securing sufficient resources to process it completely may often be overkill due to its irregular arrival timing and short-lived nature [12]. In our context, the peak loads of servers occur at different timings, magnitudes, and patterns that are affected by both the vehicles' local loads and the traffic flows and need to be treated differently.

Fig. 12 compares the time-series composite loads occurring at multiple areas. More specifically, each graph has two different time-series composite loads occurring in a pair of areas, and those area IDs are shown on top of each graph; the blue solid line and the red-dotted line indicate the time-series composite load of the left area ID and the right area ID, respectively. Those lines show different peak load styles, and we analyze those from three perspectives: their peak timings, magnitudes, and similarities.

(Observation 4) The geographically adjacent areas have similar timed patterns of the peak loads.

Informally, we consider two peak patterns to be similar whenever (1) both areas have similar numbers of peaks that are significantly deviated from their average loads and (2) the similar time gaps between successive peaks.

Fig. 12 shows two categories of area pairs and their peak loads; the four geographically adjacent area pairs: (a) NYC (7, 11), (b) NYC (3, 9), (d) THEA (2, 8), (e) THEA (2, 4); the two geographically non-adjacent areas: (c) NYC (1, 8), (f) THEA (1, 7).

Geographically non-adjacent areas have significantly different peak patterns; for example, the peak patterns of (a) NYC (7, 11) and (d) THEA (2, 8) are very different; the peak patterns of (c) NYC (1, 8) and (f) THEA (1, 7) are also very different. On the other hand, two geographically adjacent areas have similar peak patterns; for example, the peak patterns of NYC-7 and NYC-11 in pair (a) are similar, and the similar observation holds in pair (b), (d), (e).

This shows two geographically adjacent areas would experience peak loads with similar frequencies and intervals. From the resource management perspective, each server needs to predict when and how often the peak load would occur to determine the proper timing of the resource rescaling. However, in many scenarios, the timing of the peak load is highly unpredictable, so it is not easy to know it in advance. In this case, an edge server can use the history of the peak load occurrences at the adjacent servers as a good indicator of its own peak loads so that the appropriate timing of the resource rescaling can be more precisely determined.

(Observation 5) The geographically adjacent areas show different peak magnitudes.

The peak magnitude implies how large a load deviates from the average load, and this is an orthogonal issue from its timed pattern. For example, two areas may experience similar intervals of successive peaks, but the degree to which each peak deviates from its average load can be different.

In Fig. 12, (a) NYC (7,11) has a relatively smaller gap of the peak magnitudes than (b) NYC (3, 9), (d) THEA (2, 8), (e) THEA (2, 4). That is, the height difference between the two lines in (b), (d), (e) is much larger than (a).

One reason for such deviation is the traffic volume change from one area to another due to the presence of other alternative routes. Referring to Fig. 8, for example, in NYC (3, 9), the traffic flows in the direction from NYC-3 to NYC-9. However, there is another alternative route from NYC-3 to NYC-2. This means that not all vehicles from NYC-3 flow to NYC-9; only a partial number of vehicles from NYC-3 would eventually flow to NYC-9. For this reason, the magnitude of NYC-9 (blue) is smaller than NYC-3 (red) in (b) due to the smaller number of vehicles. On the other hand, in NYC (7, 11), there is no alternative route between NYC-7 and NYC-11. Hence, all vehicles from NYC-7 flow to NYC-11, which makes relatively similar magnitudes between the two.

However, the traffic volume change is not the sole factor for the magnitude. For example, in THEA(2, 8), the vehicles flow from THEA-8 to THEA-2, and there is no alternative route. Nonetheless, the peak magnitudes of the two areas are substantial. This is because THEA-2 is adjacent to the 4-way intersection, which makes the traffic flow more congested than THEA-8. This difference is also supported by Fig. 11.(h) where the traffic volumes of THEA 2 and 8 are similar, but their speeds differ. Since both cases use the periodic offload patterns in (d), the slower traffic flow at THEA-2 makes the

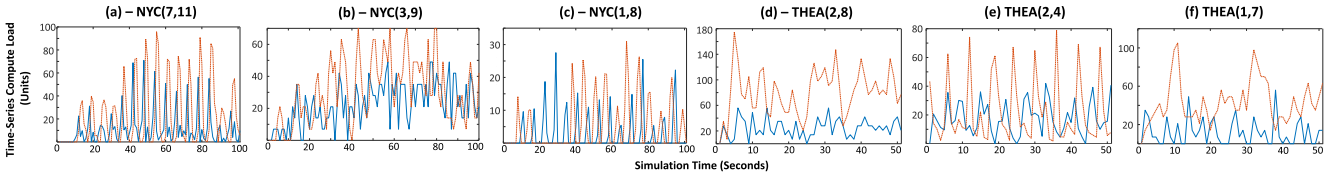


FIGURE 12. Comparison of the peak compute loads in 12 different areas; for each pair of the area ID in the upper title, the left side matches to the red-dotted line, and the right side matches to the blue solid line; (a) NYC-Congested-Speed-Dependent Load, (b) NYC-Free Flow-Speed-Dependent Load, (c) NYC-Congested-Speed-Dependent Load, (d) THEA-Free Flow-Periodic Load, (e) THEA-Congested-Speed-Dependent Load, (f) THEA-Congested-Periodic Load.

vehicles stay in the area longer than THEA-8, resulting in a higher peak magnitude.

(Observation 6) The travel times and driving directions between the two areas affect the absolute peak timings of the geographically adjacent areas.

Even though the two areas have a similar pattern of peak loads, the absolute timing when each peak occurs can be different due to the propagation delay of the peak loads from one area to another. The travel direction and time are major factors affecting such propagation delay.

In NYC (7, 11), the driving direction is one way from NYC-7 to NYC-11, while THEA (2, 8) has the opposite direction from THEA-8 to THEA-2. For this reason, the directions of the propagation of the blue and red lines in (a) and (d) are opposite. The propagation delay in (a) NYC (7, 11) and (d) THEA (2, 8) are also different due to the travel time difference experienced at each source area (*i.e.,* NYC-7 and THEA-8).

Many factors determine the travel time, and one factor is the area size; THEA-8 has a larger area size than NYC-7, so it typically experiences a longer travel time. For this reason, the propagation delay of the peak load in (d) (*i.e.,* the size of the shift between the blue and the red lines) is longer than (a).

From the resource management perspective, this aspect can also be used to determine the timing of the resource scaling. When a peak load occurs in one adjacent area, one can understand which adjacent servers would experience similar peak loads and how long it would take to determine the appropriate timing of the resource scaling.

(Observation 7) The magnitudes of the peak loads in different areas do not always match with the magnitudes of the peak volumes of their traffic flows.

As explained in the aggregate load analysis, the naïve conjecture - *a higher server's composite load would be generated in the area where more vehicles exist* - also does *not* hold generally true for the peak load analysis as well; it depends on which variable plays more dominant factors affecting the magnitudes of the peak loads. Fig. 13 shows one comparison of the composite loads (left) and the traffic volumes (right) in the area pairs in NYC. In the right graph, NYC-11 (blue) has a higher magnitude of the peak volume in its traffic flow than NYC-11 (red), while the composite load in NYC-11 is always lower than NYC-7.

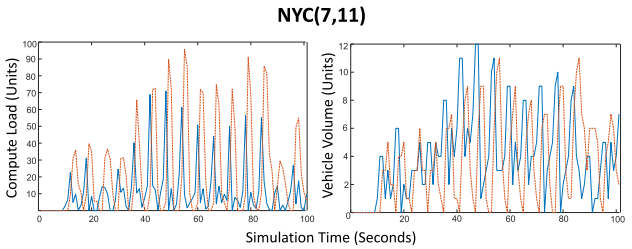


FIGURE 13. The comparison of the compute load (left) and the traffic volume (right) in NYC-Congested-Speed-Dependent Load where the magnitudes of the compute loads and the traffic volumes are in the reverse relationship; NYC-7 is the red-dotted line, and the NYC-11 is the the blue solid line.

This is because, since the speed-dependent offload pattern is used for NYC (7, 11), NYC-7 has a much higher delay (*i.e.,* lower speed) than NYC-11 as shown in Fig. 10.(c). The lower average speed of NYC-7 plays a more dominant negative role than the traffic volume factor, resulting in a lower composite load than NYC-11.

VII. DISCUSSION

While the edge server is the core component to realize many V2X connected vehicle services, deploying them on public roads is challenging since it requires a huge infrastructure cost and an in-depth analysis of its safety and effectiveness. We explain how our findings can be an input to the edge server design and deployment for ongoing standardization activities [53], [59].

A. INTEGRATION WITH NETWORK COMMUNICATION MODELS

While our work focuses on characterizing the compute workloads imposed on edge servers, it is also important to consider the network performance (*e.g.,* network bandwidth, latency, packet drop rates) to reason about the system-level Quality of Service (QoS) requirements. For example, in the intelligent intersection management scenario, the vehicles may share their driving trajectories with the RSU (edge server), which would decide whether an imminent collision among them would occur. In this case, the service's end-to-end latency is defined as the time interval from the moment the vehicles send their trajectories to the RSU to the moment

the vehicles receive the collision warning message. Such an end-to-end latency (*i.e.,* the system-level QoS requirement) is affected by both the compute load and the network performance (*e.g.,* network bandwidth, latency, packet drop rates), and we explain how our work can be integrated with the network research domain [44], [46].

Firstly, the timing when the composite load occurs may vary depending on the choice of network models that enable communication between the vehicle and the servers. When the local loads are generated, the timing when such loads are imposed on the server may be different due to the network latency or packet drop rate. Much research has been done to construct realistic analytic models of such network behavior, which can be integrated with our compute workload characterization. For example, the distance between the vehicles and the servers can be obtained by our data generator; this can be input to the function to compute the average packet delivery ratio (PDR) proposed by Gonzalez-Martín's work [43]; then, the obtained PDR can then be used to determine more realistic timing when the load arrives at the server's side.

Secondly, the research on comparative performance analysis of the V2X communication protocols can be integrated with our work to decide the most appropriate combination of the compute and network model to achieve the system-level QoS requirements. For example, Mannoni et al. studied the performance comparison between ITS-G5 and C-V2X MAC layer protocols [21], and Ali et al. analyzed the performance of different modes of the 3GPP NR V2X protocol [45]. According to that literature, various tradeoff points, such as network bandwidth, latency, energy consumption, and monetary costs, exist when selecting a specific communication protocol to serve certain V2X services; analyzing such tradeoffs requires understanding the compute load generated by the target V2X services, and the workload characterization from our work can be utilized.

B. COMPUTE RESOURCE ALLOCATION

The edge server needs to be equipped with an appropriate size of compute resources (*e.g.,* CPU power or memory sizes), taking into account the number of service requests from vehicles. Allocating too much compute resources would incur unnecessary infrastructure costs, while allocating less resources would not guarantee the service requirements.

When judging the amount of the edge server's compute resources, the two factors, the traffic flow and the compute offload patterns, shall be taken into account together; considering only one aspect and ignoring the other would not accurately estimate the size of the edge server's composite load, which consequently results in over or under-estimation of the required compute resources.

For example, allocating the compute resource in proportion to the number of vehicles present in the area may not always be the best strategy; if the edge server deployed in the area typically experiences a relatively high average speed while the service request period from each vehicle is relatively long, the edge server would not require the same magnitude of the compute load increase as much as the area where the smaller number of vehicles request offloads much more frequently.

In addition, there is a spatial locality in the pattern of the edge server's composite loads; the closer edge servers tend to have a similar pattern in their peak loads. This means that, in order to estimate the compute resources of the edge server in one area, the history of the composite load of its adjacent edge server can be used as a good indicator.

Despite the similarity of the peak load patterns, however, the timings and the magnitudes associated with the peak loads can be different. This is because the traffic volume from one edge server area is different from its adjacent edge server area; some former vehicles may take alternative routes, or multiple routes may be converged into one edge server area, which may change the timing and magnitudes. If such locality aspects are appropriately analyzed together, the timings and the magnitudes of the adjacent edge server's composite loads can be more accurately estimated, which would help its compute resource scaling as well.

C. COMPUTE OFFLOAD SCHEDULING

Each compute offload request should satisfy timing constraints at various granularities and the spatial constraints where the request needs to be processed. For example, the edge-assisted collision avoidance service needs to collect the positions of the vehicles and the pedestrians at intersections and instantly (*e.g.,* in the order of milliseconds) inform the vehicles of the potential collision hazard. On the other hand, the HD map service may collect road images from vehicles and update the road geometry of the corresponding area at a coarser time granularity since the updated geometry does not need to be immediately reflected (*e.g.,* in the order of seconds or minutes).

Our observations can be used to construct more effective scheduling strategies for offload requests. That is, when a certain edge server experiences too large composite loads at a particular moment, multiple granularities of the timing and spatial constraints provide room for such offload requests to be scheduled in other edge servers.

For example, suppose other offload requests have more relaxed requirements. In that case, a scheduling strategy may decide to process them in other under-utilized edge servers on the vehicles' route later. In order to find such under-utilized edge servers, one may check another route where the traffic flow significantly diverges from the current location, which would distribute the compute load and offload the request to the edge server on the route. Alternatively, one can check the average speed in each route and intentionally take a slower route that would give a longer time to offload the request with a significant processing time, sacrificing the travel time increase.

VIII. CONCLUSION

Our empirical analysis indicates that understanding the edge server's composite load requires accounting for the intricate interaction between traffic flow (e.g., volume, density, speed) and local load (e.g., periodic, speed-dependent offloading). Even when vehicles generate the same type of local loads, the composite loads on edge servers can vary significantly based on the speed, number of vehicles, and the size of the area they cover. Geographically adjacent edge servers often exhibit similar peak load patterns; however, the magnitudes and timings of these peaks can differ depending on how traffic flow diverges or converges between areas. Building on these findings, we aim to develop a theoretical model of the edge server's composite load. This model will mathematically capture its characteristics, enabling formal analyses such as end-to-end latency analysis, resource allocation, and schedulability tests.

ACKNOWLEDGMENT

The authors are grateful to Jihyun Kwon and Sangeun Park for their significant effort to disseminate the research results to the public. They also would like to acknowledge the invaluable discussion with Prof. Hoon Sung Chwa, Jeongho Kwak, and Ji-Woong Choi.

REFERENCES

[1] S. Bucur, V. Ureche, C. Zamfir, and G. Candea, "Parallel symbolic execution for automated real-world software testing," in *Proc. 6th Conf. Comput. Syst.*, Apr. 2011, pp. 183–198, doi: 10.1145/1966445.1966463.

[2] K. Altisen and M. Moy, "Ac2lus: Bringing SMT-solving and abstract interpretation techniques to real-time calculus through the synchronous language Lustre," in *Proc. 22nd Euromicro Conf. Real-Time Syst.*, Jul. 2010, pp. 207–216, doi: 10.1109/ECRTS.2010.11.

[3] A. Ashok, P. Steenkiste, and F. Bai, "Adaptive cloud offloading for vehicular applications," in *Proc. IEEE Veh. Netw. Conf. (VNC)*, Dec. 2016, pp. 1–8, doi: 10.1109/VNC.2016.7835966.

[4] S. Chakraborty, S. Kunzli, and L. Thiele, "A general framework for analysing system properties in platform-based embedded system designs," in *Proc. Design, Autom. Test Eur. Conf. Exhibition*, Mar. 2003, pp. 190–195, doi: 10.1109/DATE.2003.1253607.

[5] N. Guan and W. Yi, "Finitary real-time calculus: Efficient performance analysis of distributed embedded systems," in *Proc. IEEE 34th Real-Time Syst. Symp.*, Dec. 2013, pp. 330–339, doi: 10.1109/RTSS.2013.40.

[6] X. Guo, R. Singh, T. Zhao, and Z. Niu, "An index based task assignment policy for achieving optimal power-delay tradeoff in edge cloud systems," in *Proc. IEEE Int. Conf. Commun. (ICC)*, May 2016, pp. 1–7, doi: 10.1109/ICC.2016.7511147.

[7] S. Kunzli, F. Poletti, L. Benini, and L. Thiele, "Combining simulation and formal methods for system-level performance analysis," in *Proc. Design Autom. Test Eur. Conf.*, vol. 1, Mar. 2006, pp. 1–6, doi: 10.1109/DATE.2006.244109.

[8] K. Lampka, S. Perathoner, and L. Thiele, "Analytic real-time analysis and timed automata: A hybrid method for analyzing embedded real-time systems," in *Proc. 7th ACM Int. Conf. Embedded Softw.*, Oct. 2009, pp. 107–116, doi: 10.1145/1629335.1629351.

[9] X. Li, J.-L. Scharbarg, and C. Fraboul, "Improving end-to-end delay upper bounds on an AFDX network by integrating offsets in worst-case analysis," in *Proc. IEEE 15th Conf. Emerg. Technol. Factory Autom. (ETFA)*, Sep. 2010, pp. 1–8, doi: 10.1109/ETFA.2010.5641178.

[10] C.-W. Lin, M. Di Natale, H. Zeng, L. T. X. Phan, and A. Sangiovanni-Vincentelli, "Timing analysis of process graphs with finite communication buffers," in *Proc. IEEE 19th Real-Time Embedded Technol. Appl. Symp. (RTAS)*, Apr. 2013, pp. 227–236, doi: 10.1109/RTAS.2013.6531095.

[11] P. A. Lopez, M. Behrisch, L. Bieker-Walz, J. Erdmann, Y.-P. Flötteröd, R. Hilbrich, L. Lücken, J. Rummel, P. Wagner, and E. Wiessner, "Microscopic traffic simulation using SUMO," in *Proc. 21st Int. Conf. Intell. Transp. Syst. (ITSC)*, Nov. 2018, pp. 2575–2582, doi: 10.1109/ITSC.2018.8569938.

[12] M. Mattess, C. Vecchiola, and R. Buyya, "Managing peak loads by leasing cloud infrastructure services from a spot market," in *Proc. IEEE 12th Int. Conf. High Perform. Comput. Commun. (HPCC)*, Sep. 2010, pp. 180–188, doi: 10.1109/HPCC.2010.77.

[13] J. Oueis, E. C. Strinati, and S. Barbarossa, "Small cell clustering for efficient distributed cloud computing," in *Proc. 25th IEEE Annu. Int. Symp. Pers., Indoor, Mobile Radio Commun.*, Sep. 2014, pp. 1474–1479, doi: 10.1109/PIMRC.2014.7136401.

[14] M. Salem, M. Crowley, and S. Fischmeister, "Anomaly detection using inter-arrival curves for real-time systems," in *Proc. 28th Euromicro Conf. Real-Time Syst. (ECRTS)*, Jul. 2016, pp. 97–106, doi: 10.1109/ECRTS.2016.22.

[15] Y. Tang, N. Guan, W. Liu, L. T. Xuan Phan, and W. Yi, "Revisiting GPC and AND connector in real-time calculus," in *Proc. IEEE Real-Time Syst. Symp. (RTSS)*, Dec. 2017, pp. 255–265, doi: 10.1109/RTSS.2017.00031.

[16] S. M. Shahrear Tanzil, O. N. Gharehshiran, and V. Krishnamurthy, "Femto-cloud formation: A coalitional game-theoretic approach," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Dec. 2015, pp. 1–6, doi: 10.1109/GLOCOM.2015.7417264.

[17] H. Wang, X. Li, H. Ji, and H. Zhang, "Federated offloading scheme to minimize latency in MEC-enabled vehicular networks," in *Proc. IEEE Globecom Workshops (GC Wkshps)*, Dec. 2018, pp. 1–6, doi: 10.1109/GLOCOMW.2018.8644315.

[18] Y. Zhang, H. Liu, L. Jiao, and X. Fu, "To offload or not to offload: An efficient code partition algorithm for mobile cloud computing," in *Proc. IEEE 1st Int. Conf. Cloud Netw. (CLOUDNET)*, Nov. 2012, pp. 80–86, doi: 10.1109/CLOUDNET.2012.6483660.

[19] T. Zhao, S. Zhou, X. Guo, Y. Zhao, and Z. Niu, "A cooperative scheduling scheme of local cloud and Internet cloud for delay-aware mobile cloud computing," in *Proc. IEEE Globecom Workshops (GC Wkshps)*, Dec. 2015, pp. 1–6, doi: 10.1109/GLOCOMW.2015.7414063.

[20] B. Kim and D. Gangadharan, "Empirical composite workload analysis for RSU-assisted computation offloading in connected vehicle services," in *Proc. IEEE Int. Symp. Workload Characterization (IISWC)*, Oct. 2023, pp. 221–222, doi: 10.1109/IISWC59245.2023.00015.

[21] V. Mannoni, V. Berg, S. Sesia, and E. Perraud, "A comparison of the V2X communication systems: ITS-G5 and C-V2X," in *Proc. IEEE 89th Veh. Technol. Conf.*, Apr. 2019, pp. 1–5, doi: 10.1109/VTCSPRING.2019.8746562.

[22] Q. Chao, H. Bi, W. Li, T. Mao, Z. Wang, M. C. Lin, and Z. Deng, "A survey on visual traffic simulation: Models, evaluations, and applications in autonomous driving," in *Proc. Comput. Graph. Forum*, Jul. 2019, vol. 39, no. 1, pp. 287–308.

[23] R. Alur and D. L. Dill, "A theory of timed automata," *Theor. Comput. Sci.*, vol. 126, no. 2, pp. 183–235, Apr. 1994.

[24] J. Barrachina, P. Garrido, M. Fogue, F. J. Martinez, J.-C. Cano, C. T. Calafate, and P. Manzoni, "Road side unit deployment: A density-based approach," *IEEE Intell. Transp. Syst. Mag.*, vol. 5, no. 3, pp. 30–39, Jul. 2013, doi: 10.1109/MITS.2013.2253159.

[25] J. -Y. Le Boudec and P. Thiran, *Network Calculus*. Cham, Switzerland: Springer, 2021.

[26] M. C. Calzarossa, L. Massari, and D. Tessera, "Workload characterization: A survey revisited," *ACM Comput. Surveys*, vol. 48, no. 3, pp. 1–43, Feb. 2016, doi: 10.1145/2856127.

[27] G. Carvajal, M. Salem, N. Benann, and S. Fischmeister, "Enabling rapid construction of arrival curves from execution traces," *IEEE Des. Test*, vol. 35, no. 4, pp. 23–30, Aug. 2018, doi: 10.1109/MDAT.2017.2771210.

[28] A. B. De Souza, P. A. L. Rego, T. Carneiro, J. D. C. Rodrigues, P. P. R. Filho, J. N. De Souza, V. Chamola, V. H. C. De Albuquerque, and B. Sikdar, "Computation offloading for vehicular environments: A survey," *IEEE Access*, vol. 8, pp. 198214–198243, 2020, doi: 10.1109/ACCESS.2020.3033828.

[29] J. Du, F. R. Yu, X. Chu, J. Feng, and G. Lu, "Computation offloading and resource allocation in vehicular networks based on dual-side cost minimization," *IEEE Trans. Veh. Technol.*, vol. 68, no. 2, pp. 1079–1092, Feb. 2019, doi: 10.1109/TVT.2018.2883156.

[30] N. Halbwachs, F. Lagnier, and C. Ratel, "Programming and verifying real-time systems by means of the synchronous data-flow language LUSTRE," *IEEE Trans. Softw. Eng.*, vol. 18, no. 9, pp. 785–793, Sep. 1992, doi: 10.1109/32.159839.

[31] K. Kumar, J. Liu, Y.-H. Lu, and B. Bhargava, "A survey of computation offloading for mobile systems," *Mobile Netw. Appl.*, vol. 18, no. 1, pp. 129–140, Feb. 2013, doi: 10.1007/S11036-012-0368-0.

[32] X. Li, Y. Dang, M. Aazam, X. Peng, T. Chen, and C. Chen, "Energy-efficient computation offloading in vehicular edge cloud computing," *IEEE Access*, vol. 8, pp. 37632–37644, 2020, doi: 10.1109/ACCESS.2020.2975310.

[33] P. Mach and Z. Becvar, "Mobile edge computing: A survey on architecture and computation offloading," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 3, pp. 1628–1656, 3rd Quart., 2017, doi: 10.1109/COMST.2017.2682318.

[34] D. Magalhães, R. N. Calheiros, R. Buyya, and D. G. Gomes, "Workload modeling for resource usage analysis and simulation in cloud computing," *Comput. Electr. Eng.*, vol. 47, pp. 69–81, Oct. 2015, doi: 10.1016/J.COMPELECENG.2015.08.016.

[35] M. Moy and K. Altisen, "Arrival curves for real-time calculus: The causality problem and its solutions," in *Tools and Algorithms for the Construction and Analysis of Systems*. Springer, 2010, pp. 358–372.

[36] Y. Ni, J. He, L. Cai, J. Pan, and Y. Bo, "Joint roadside unit deployment and service task assignment for Internet of Vehicles (IoV)," *IEEE Internet Things J.*, vol. 6, no. 2, pp. 3271–3283, Apr. 2019, doi: 10.1109/JIOT.2018.2882436.

[37] N. Nikookaran, G. Karakostas, and T. D. Todd, "Combining capital and operating expenditure costs in vehicular roadside unit placement," *IEEE Trans. Veh. Technol.*, vol. 66, no. 8, pp. 7317–7331, Aug. 2017, doi: 10.1109/TVT.2017.2665480.

[38] B. Park and J. D. Schneeberger, "Microscopic simulation model calibration and validation: Case study of VISSIM simulation model for a coordinated actuated signal system," *Transp. Res. Record, J. Transp. Res. Board*, vol. 1856, no. 1, pp. 185–192, Jan. 2003.

[39] O. Sokolsky and A. Chernoguzov, "Performance analysis of AADL models using real-time calculus," in *Foundations of Computer Software Future Trends and Techniques for Development*. Cham, Switzerland: Springer, 2010, pp. 227–249.

[40] Y. Sun, X. Guo, J. Song, S. Zhou, Z. Jiang, X. Liu, and Z. Niu, "Adaptive learning-based task offloading for vehicular edge computing systems," *IEEE Trans. Veh. Technol.*, vol. 68, no. 4, pp. 3061–3074, Apr. 2019, doi: 10.1109/TVT.2019.2895593.

[41] H. Wang, T. Liu, B. Kim, C.-W. Lin, S. Shiraishi, J. Xie, and Z. Han, "Architectural design alternatives based on cloud/edge/fog computing for connected vehicles," *IEEE Commun. Surveys Tuts.*, vol. 22, no. 4, pp. 2349–2377, 4th Quart., 2020, doi: 10.1109/COMST.2020.3020854.

[42] H. Yang, M. G. H. Bell, and Q. Meng, "Modeling the capacity and level of service of urban transportation networks," *Transp. Res. Part B, Methodol.*, vol. 34, no. 4, pp. 255–275, May 2000.

[43] M. Gonzalez-Martín, M. Sepulcre, R. Molina-Masegosa, and J. Gozalvez, "Analytical models of the performance of C-V2X mode 4 vehicular communications," *IEEE Trans. Veh. Technol.*, vol. 68, no. 2, pp. 1155–1166, Feb. 2019, doi: 10.1109/TVT.2018.2888704.

[44] S. Gyawali, S. Xu, Y. Qian, and R. Q. Hu, "Challenges and solutions for cellular based V2X communications," *IEEE Commun. Surveys Tuts.*, vol. 23, no. 1, pp. 222–255, 1st Quart., 2021, doi: 10.1109/COMST.2020.3029723.

[45] Z. Ali, S. Lagén, L. Giupponi, and R. Rouil, "3GPP NR V2X mode 2: Overview, models and system-level evaluation," *IEEE Access*, vol. 9, pp. 89554–89579, 2021, doi: 10.1109/ACCESS.2021.3090855.

[46] M. H. C. Garcia, A. Molina-Galan, M. Boban, J. Gozalvez, B. Coll-Perales, T. Sahin, and A. Kousaridas, "A tutorial on 5G NR V2X communications," *IEEE Commun. Surveys Tuts.*, vol. 23, no. 3, pp. 1972–2026, 3rd Quart., 2021, doi: 10.1109/COMST.2021.3057017.

[47] M. Kezia and K. V. Anusuya, "Mobility models for Internet of Vehicles: A survey," *Wireless Pers. Commun.*, vol. 125, no. 2, pp. 1857–1881, Jul. 2022, doi: 10.1007/s11277-022-09637-7.

[48] (2022). *ASAM Open Simulation Interface (OSI)*. [Online]. Available: https://www.asam.net/standards/detail/osi/

[49] (2023). *Automotive Edge Computing Consortium—General Principle and Vision*. [Online]. Available: https://aecc.org/resources/publications/

[50] (2020). *Harmonizing Standards for Edge Computing—A Synergized Architecture Leveraging ETSI ISG MEC and 3GPP Specifications*. [Online]. Available: https://www.etsi.org/images/files/ETSIWhitePapers/ETSI_wp36_Harmonizing-standards-for-edge-computing.pdf

[51] *Using Connected Vehicle Technologies to Solve Real-World Operational Problems*. Accessed: Apr. 1, 2024. [Online]. Available: https://www.its.dot.gov/pilots/

[52] (2024). *Architecture Reference for Cooperative and Intelligent Transportation*. [Online]. Available: https://www.arc-it.net/

[53] (2022). *Institute of Transportation Engineers*. [Online]. Available: https://www.ite.org/technical-resources/standards/rsu-standardization/

[54] *Intelligent Transport Systems—Lane Keeping Assistance Systems (LKAS)—Performance Requirements and Test Procedures*, Standard ISO 11270:2014, 2014.

[55] 5GAA-Automotive Assoc. (2023). *United States Vehicle-to-Infrastructure Communications; Day One Deployment Guide*. [Online]. Available: https://5gaa.org/content/uploads/2023/10/5gaa-wi-usdploy-231667-technical-report-guidance-day-1.pdf

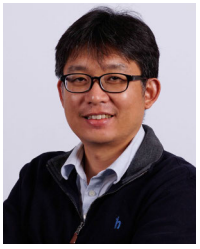
[56] *Transportation Research Board, Engineering National Academies of Sciences, and Medicine*, 7th ed., Nat. Academies Press, Washington, DC, USA, 2022.

[57] F. L. Hall, "Traffic stream characteristics," in *Traffic Flow Theory*. Washington, DC, USA: U.S. Federal Highway Administration, 1996.

[58] A. D. May, *Traffic Flow Fundamentals*. Upper Saddle River, NJ, USA: Prentice-Hall, 1990.

[59] F. Perry, R. Kelli, E. Leslie, Z. Huang, and V. D. Duren. (2017). *Dedicated Short-Range Communications Roadside Unit Specifications*. U.S Dept. Transp. [Online]. Available: https://rosap.ntl.bts.gov/view/dot/3600

[60] E. Wandeler, *Modular Performance Analysis and Interface-Based Design for Embedded Real-Time Systems*. Zurich, Switzerland: Swiss Federal Institute of Technology, 2006.



BAEKGYU KIM (Member, IEEE) received the B.S. and M.S. degrees in electrical engineering and computer science from Kyungpook National University, South Korea, and the Ph.D. degree in computer science from the University of Pennsylvania. He is currently an Assistant Professor with the Department of Electrical Engineering and Computer Science, DGIST. From 2015 to 2021, he was a Principal Researcher with Toyota Motor North America and InfoTech Laboratories, where he has conducted industrial research on connected car software platforms. He has published more than 60 peer-reviewed academic papers and 60 international patents filed/granted. In particular, he has published several peer-reviewed articles and international patents based on the research on analyzing the safety of the autonomous driving software in virtual environments. His primary research interests include verification, validation, and optimization techniques to guarantee assurances for the Internet of Things and cyber physical systems. He received the Best Poster Award from the Korean Computer Scientists and Engineers Association in America (KOCSEA); the Top Inventor Award from Toyota Info Technology Center, U.S.A.; and his collaborative work received the SAE Vincent Bendix Automotive Electronics Engineering Award (Best Paper Award). He serves on technical committee in various academic conferences, including IEEE ICCPS, IEEE CLOUD, and ASP-DAC. He is a Technical Committee Member on IEEE Internet of Things in Intelligent Transportation Systems.

DEEPAK GANGADHARAN (Member, IEEE) received the Ph.D. degree in computer science from the National University of Singapore, in 2012. Subsequently, he was a Postdoctoral Researcher with the Technical University of Denmark, the University of Erlangen-Nuremberg, and the University of Pennsylvania. He is currently an Assistant Professor with the Computer Systems Group, IIIT Hyderabad. His main research interests include scalable design and performance analysis of edge-based IoT/cyber-physical systems, design methodologies for embedded systems, and intelligent transportation systems. His research has won Best Paper Awards in the conferences ISORC 2018 and MECO 2013. He has served in the Program Committee of IEEE conferences, such as HiPC, ANTS, and MASS.