Serverless Cloud-based Speed Advisory Application for Connected Vehicles – Case Study

Hsien-Wen Deng
Clemson University
M Sabbir Salek (msalek@clemson.edu)
Clemson University
Mizanur Rahman
University of Alabama
Mashrur Chowdhury
Clemson University
Mitch Shue
Clemson University
Amy W. Apon
Clemson University

Research Article

Keywords: Amazon web services, connected vehicles, internet of things, serverless architecture, transportation cyber-physical systems

Posted Date: February 28th, 2022

DOI: https://doi.org/10.21203/rs.3.rs-1377325/v1

Serverless Cloud-based Speed Advisory Application for Connected Vehicles – Case Study

Hsien-Wen Deng[1], M Sabbir Salek[2]*, Mizanur Rahman[3], Mashrur Chowdhury[4], Mitch Shue[5], and Amy W. Apon[6]

[1]M.S., School of Computing, Clemson University, Clemson, SC 29634, USA; Email: hsienwd@clemson.edu

[2]M.S., Glenn Department of Civil Engineering, Clemson University, Clemson, SC 29634, USA; Email: msalek@clemson.edu

[3]Ph.D., Department of Civil, Construction & Environmental Engineering, The University of Alabama, Tuscaloosa, AL 35487, USA; Email: mizan.rahman@ua.edu

[4]Ph.D., Glenn Department of Civil Engineering, Clemson University, 216 Lowry Hall, Clemson, SC 29634, USA; Email: mac@clemson.edu

[5]M.S., School of Computing, Clemson University, Clemson, SC 29634, USA; Email: mshue@clemson.edu

[6]Ph.D., School of Computing, Clemson University, Clemson, SC 29634, USA; Email: aapon@clemson.edu

*Corresponding author

ABSTRACT

In this study, we develop a real-time connected vehicle (CV) speed advisory application, which we refer to as "Serverless CloSA", using commercial cloud services and present case studies for a signalized corridor for different roadway traffic conditions. First, we develop a highly scalable serverless cloud computing architecture using Amazon Web Services (AWS) to support the requirements of a real-time CV application. Second, we develop an optimization-based real-time CV speed advisory algorithm that is deployable in the cloud. Third, we develop a cloud-in-the-loop simulation testbed using AWS and an open-source microscopic roadway traffic simulator called Simulation of Urban Mobility (SUMO). Then, we conduct three case studies for three different roadway traffic conditions, i.e., low, medium, and high-density traffic. Our analyses show that Serverless CloSA can reduce the average stopped delays at signalized intersections in a corridor by 77% while reducing the aggregated risk of collision by 21% compared to the baseline scenario, i.e., no speed advisory for the CVs. Our experiments show an average end-to-end delay of 452 ms, which is well under the 1000 ms delay threshold of real-time CV mobility applications. Thus, this study also demonstrates the feasibility of deploying a real-time CV mobility application using commercial cloud services.

Keywords: Amazon web services, connected vehicles, internet of things, serverless architecture, transportation cyber-physical systems

1 INTRODUCTION

In transportation cyber-physical systems (TCPS), the interaction between cyber and physical systems makes it possible to develop real-time CV applications [1]. However, to develop a real-time feedback-based interaction between cyber systems and physical systems, high-performance computing infrastructure is required to process the heterogeneous data from different sources. While edge or fog computing offers a viable solution to deploy real-time applications in a TCPS environment [2], there are issues related to edge computing-based CV application deployments, such as wireless communication range [3] and maintenance costs. The recent evolution of commercial cloud computing services has made it possible to support real-time TCPS applications in the cloud [4]. Additionally, most commercial cloud service providers now offer serverless solutions (e.g., Lambda [5] offered by Amazon Web Services (AWS), Azure Functions [6] offered by Microsoft Azure) that remove the burden of establishing server instances and enable developers to focus primarily on application development, such as CV and Internet of Things (IoT) applications.

In a server-based cloud application, the application developers are required to establish server instances (e.g., AWS EC2 [7]) and configure coding platforms in the cloud that will support the application. On the other hand, in a serverless cloud-based application, the application developers do not need to establish the server instances as the computational resources are managed by the cloud itself based on the computing requirement of an application. Thus, serverless cloud is an attractive option to develop highly scalable real-time CV applications [4]. However, deploying a real-time CV application in a serverless cloud requires developing a feasible serverless cloud architecture utilizing the available cloud services as well as developing an algorithm for the CV application that is deployable through the serverless cloud architecture while meeting the latency requirements of a real-time CV application.

In this paper, we develop a CV speed advisory application (i.e., an application that provides each CV with an advised speed that changes dynamically based on various factors, such as the CV's location, surrounding traffic condition and signal phase and timing of the traffic signal at the intersection that the CV is approaching) using serverless cloud infrastructure with a goal to minimize the stopped delay experienced by CVs while passing through a signalized corridor, i.e., a roadway with traffic signals deployed at its intersections. In this TCPS environment, commercial serverless cloud infrastructure (as cyber systems) interacts with CVs and connected traffic signals (both as parts of physical systems), as shown in Fig. 1. The serverless cloud infrastructure has three types of service: (i) Function as a Service

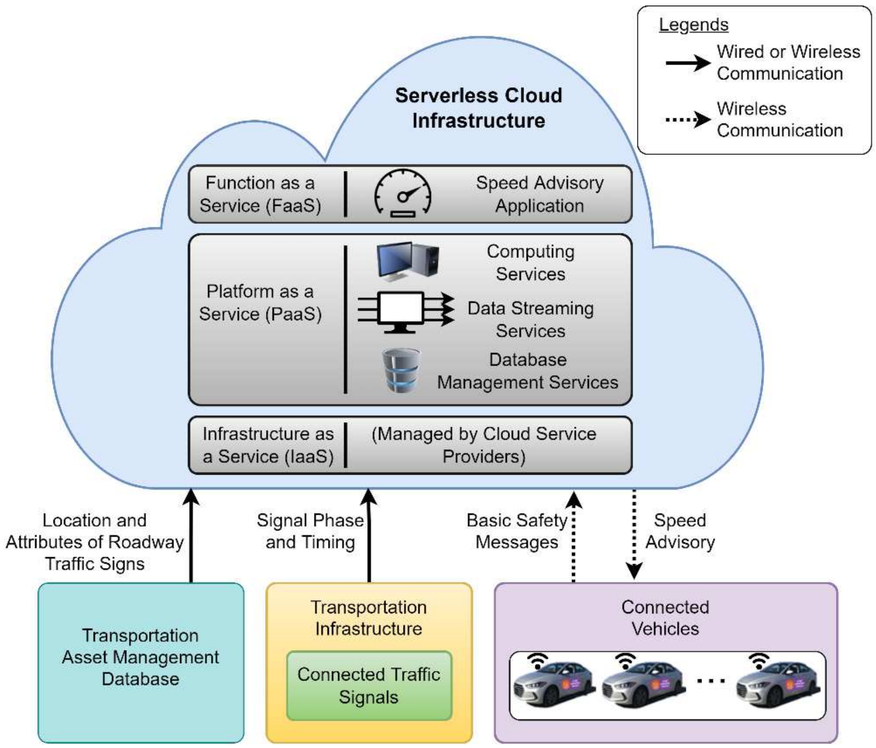


Fig. 1. Serverless cloud architecture of CV speed advisory in TCPS.

(FaaS), where a CV application, such as a speed advisory algorithm, can run, (ii) Platform as a Service (PaaS), where computing, data streaming, and database management services operate, and (iii) Infrastructure as a Service (IaaS), which is managed by the cloud service providers in a serverless architecture. The serverless architecture features a pay-as-you-go model without having to manage the underlying computing infrastructure. It is defined as Function as a Service (FaaS), which are serverless functions triggered by events as required by the application [8, 9].

The primary contributions of this study are (i) to develop a serverless cloud computing architecture using AWS for a CV speed advisory application in a TCPS environment, which we refer to as "Serverless CloSA" in this paper, and (ii) to develop an optimization-based real-time CV speed advisory application which is deployable in the cloud in terms of end-to-end latency requirement. Similar serverless architectures can also be used to develop other types of CV mobility applications, such as queue warnings and eco-driving advisories, where the maximum end-to-end latency threshold is considered to be 1000 ms [10]. In our Serverless CloSA, CVs and connected traffic signals send their state information, i.e., basic safety messages of CVs, and traffic signal phase and timing information of connected traffic signals, into the serverless cloud computing infrastructure. These messages automatically trigger the execution of the serverless functions that support the speed advisory application. Our Serverless CloSA is more scalable in terms of communication coverage area (as CVs directly communicate with the cloud to receive real-time speed advisories) and number of CVs compared to an application supported by traditional edge computing. We also develop a cloud-in-the-loop simulation testbed using AWS and Simulation of Urban Mobility (SUMO) [11], which is a widely used open-source microscopic roadway traffic simulator. Finally, we evaluate the feasibility of Serverless CloSA through a cloud-in-the-loop simulation.

2 RELATED WORK

Cloud infrastructures can effectively communicate with CVs and transportation infrastructures through vehicle-to-infrastructure (V2I) and infrastructure-to-infrastructure (I2I) communication, respectively, using wireless communication technologies, such as Cellular Vehicle-to-Everything (C-V2X), Long-Term Evolution (LTE), and 5G, or wired communication technologies, such as optical fiber-based communication technology. Services in the cloud then aggregate and analyze these data and generate appropriate information corresponding to the cloud applications. For instance, Ning et al. [12] utilized a cloud-based Fog Computing architecture to implement real-time roadway traffic management. Li et. al. [13] provided a maximum value density-based heuristic algorithm through vehicular edge cloud computing to achieve energy usage efficiency for roadway traffic. Jin et. al. [14] presented a method of constructing cloud-based mobility services for connected and automated vehicle (CAV) highway systems. All these studies used a traditional server-based architecture to develop real-time CV applications. More recently, Deng et al. [4] utilized AWS serverless infrastructure to develop a traffic surveillance application to compute the average speed of CVs in a TCPS environment. However, to our knowledge, no study has used a serverless architecture in a commercial cloud for a real-time CV application that requires the cloud infrastructure to perform computation using data coming from both CVs and transportation infrastructure in real-time while meeting the strict latency requirement of the CV mobility applications.

On the other hand, optimal speed advisory algorithms, that help CVs navigate through a signalized corridor efficiently in terms of reduced stopped delay, fuel consumption, and CO_2 emission, have been studied extensively in the literature. Many studies referred to this type of algorithm as the Green Light Optimal Speed Advisory (GLOSA) algorithm [15–20]. For instance, Suzuki and Marumo [18] developed a GLOSA system that projects a green rectangle on the roadway through the head-up display of a GLOSA-enabled vehicle. Stebbins et al. [16] combined model predictive control (MPC) with state-space reduction and GLOSA to yield efficient trajectories for the CVs. However, few studies considered platoon formation in GLOSA. Among them, Stebbins et al. [17] developed a platoon-based optimization technique for GLOSA. The authors included a safety constraint in their optimization model considering that the human drivers may not follow an advised speed if they feel that they will not be able to stop if needed while approaching an intersection. Zhao et al. [21] developed a platoon-based MPC to optimize fuel consumption which enables a platoon of vehicles to pass an intersection within a traffic signal system's green interval, where the model's efficacy was evaluated for different CV penetration rates. However, none of these studies considered a real-time implementation of the platoon-based GLOSA system for speed advisories in a signalized corridor that is "deployable" in a commercial cloud-based TCPS environment. In this study, we develop a platoon-based real-time CV speed advisory application to minimize the stopped delay experienced by the CVs that is deployable in the commercial clouds in terms of the strict latency requirement of the CV mobility applications.

3 CLOUD-BASED SERVERLESS ARCHITECTURE

AWS maintains a vast cloud infrastructure and services catalog, which makes it secure, scalable, and highly available for developing real-time CV mobility applications [4]. Besides, AWS offers various serverless services, such as AWS Lambda [5], that can be used to develop applications without being concerned about establishing or maintaining any server instances. Such serverless services generally follow pay-as-you-go billing models that make the serverless architectures cost-effective as we mentioned before [22]. Thus, in this study, we develop a serverless cloud-based CV application utilizing the serverless services offered by AWS, such as AWS Lambda. In Fig. 2, we present a serverless cloud architecture showing the computing resources, databases, and streaming services integrated to support a real-time speed advisory application for CVs using AWS. The serverless architecture removes the need for developers to manage traditional server infrastructure. Thus, we only need to focus on developing the application using relevant AWS services.

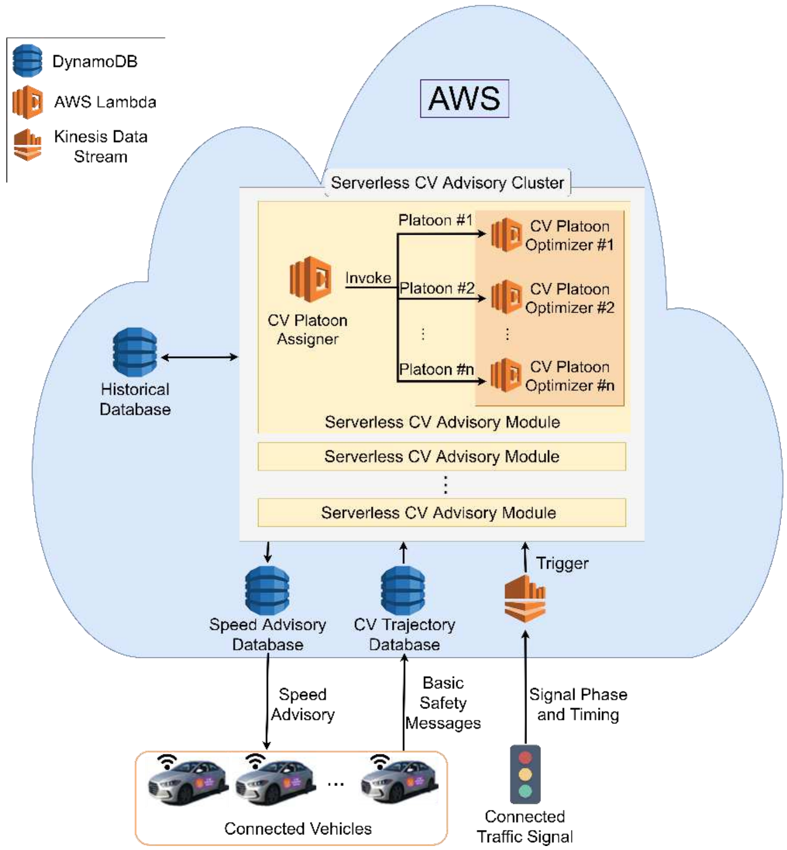


Fig. 2. Details of the Serverless CloSA architecture using AWS services.

The serverless architecture (shown in Fig. 2) employs the following AWS services: 1) DynamoDB, 2) Kinesis Data Stream (KDS), and 3) AWS Lambda. We use DynamoDB, i.e., a NoSQL database service with a key-value structure [23], for creating our databases. We create a CV trajectory database to update the CVs' trajectory information, and a speed advisory database to store speed advisory results from which the CVs can download their corresponding speed advisories in real-time. For each traffic signal, we create a historical database to save and update the distances between CVs and the traffic signal in real-time. We utilize Kinesis Data Stream (KDS), a real-time data stream service [24] in AWS, to send a message from each traffic signal to the cloud every second to trigger (i.e., launch the target program automatically) the serverless functions in CV advisory cluster. AWS Lambda [5] is the serverless compute service at the core of this serverless architecture. We design a group of AWS Lambda functions to form a serverless CV advisory cluster that gets triggered by KDS for each traffic signal. Each cluster contains multiple serverless CV advisory modules that process information from the CVs. To meet the latency requirement of a real-time CV mobility application, i.e., less than or equal to 1000 ms [10, 25], we define the capacity of each serverless CV advisory module in terms of the maximum number of CVs to be processed, which is 50 CVs per module in our AWS implementation, and run all the CV advisory modules in parallel. The usage of parallel computing in a cluster makes our Serverless CloSA fast and scalable.

There are two types of programs in each serverless CV advisory module: 1) a CV platoon assigner, and 2) a set of CV platoon optimizers. A CV platoon assigner is an AWS Lambda function that has the necessary information related to its corresponding traffic signal and intersection, such as the physical location, signal phase duration of the traffic signal, and the posted speed limit on the roadway approaching that intersection. Once the cluster is triggered, the CV platoon assigner performs the following tasks: 1) collect information from both traffic signals and CVs, 2) split the CVs into platoons based on the CVs' gap information (based on the method discussed in subsection A of section 4), 3) compute a speed advisory for only the leader CV of each platoon (based on the method discussed in subsection 4.2), and 4) save the speed advisory for the leader CV of each platoon into the speed advisory database. Then, for each platoon, the CV platoon assigner invokes a CV platoon optimizer. A CV platoon optimizer is also a serverless process, i.e., an AWS Lambda, that is responsible for its corresponding CV platoon. It computes speed advisories for the follower CVs in that platoon to help them pass the intersection while maintaining the minimum safety distances and operating within the roadway speed limit. The results, i.e., the speed advisories for the follower CVs, generated from the CV platoon optimizer are then stored in the speed advisory database.

In the real world, each CV generates Basic Safety Messages (BSMs) and each traffic signal generates signal phase and timing messages. In the Serverless CloSA, each CV uploads a filtered BSM including the CV's ID, location, speed, and the gap with its immediate leading CV into the CV trajectory database. Each traffic signal sends a filtered signal phase and timing message every second containing the current traffic signal phase and the remaining time of that phase through KDS. Our optimization-based speed advisory algorithm deployed in each serverless CV advisory cluster utilizes these BSMs and signal phase and timing messages to generate speed advisories for the CVs in real-time.

4 CLOUD-BASED SPEED ADVISORY APPLICATION

In this section, we present an optimization-based speed advisory application (i.e., Serverless CloSA) running in AWS to minimize the stopped delay for CVs at signalized intersections. The Serverless CloSA consists of three parts: 1) CV platoon identification, 2) optimization-based speed advisory algorithm for the leader CVs of the platoons, and 3) optimization-based speed advisory algorithm for the follower CVs of the platoons. Fig. 3 and Table 1 present all the relevant symbols that we use to develop the application.

4.1 CV Platoon Identification

We form CV platoons based on whether they can pass a signalized intersection within the available time of the current green time or the next green time, i.e., $t_{avail}(k)$, measured at the k^{th} timestamp. Therefore, to be identified as a platoon of N number of CVs, the last or N^{th} CV of the platoon must be able to pass the intersection within the available time, i.e., meet the following criterion:

$$\min t_{N,int}(k) \leq t_{avail}(k) \tag{1}$$

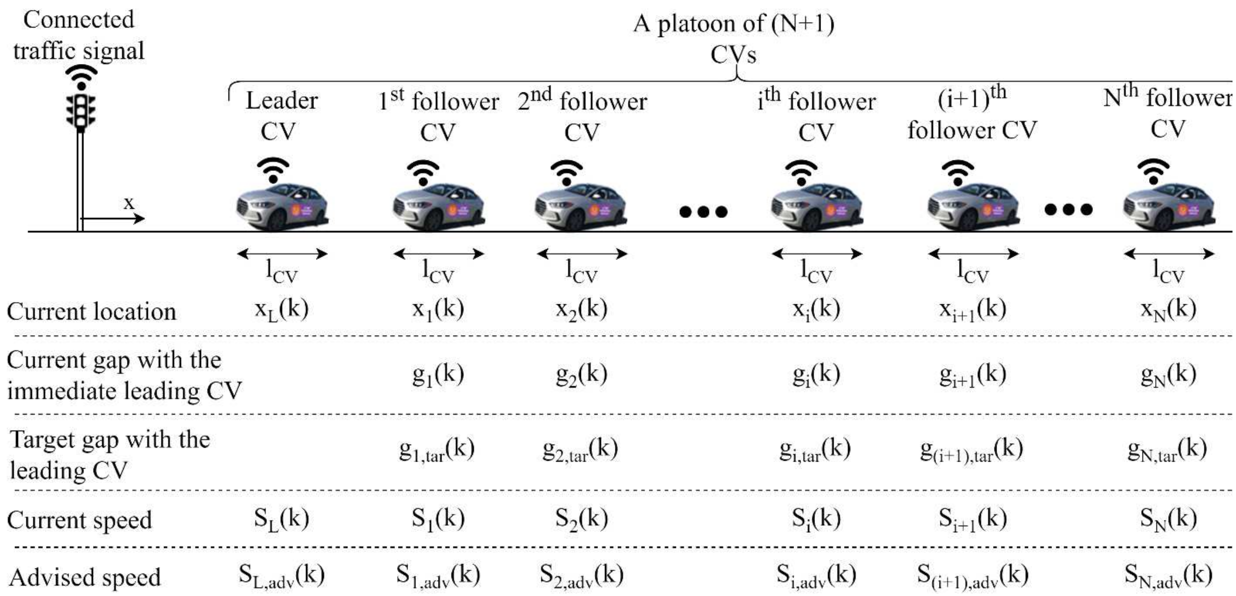


Fig. 3. Relevant symbols related to the speed advisory algorithm.

where, $t_{N,int}(k)$ denotes the estimated time taken by the N^{th} CV to reach the intersection from its location at the k^{th} timestamp. To estimate the minimum of $t_{N,int}(k)$, we consider the total time required by the N^{th} CV to accelerate from its current speed ($S_N(k)$) to the maximum speed based on the roadway speed limit (S_{max}) using its maximum acceleration (a_{Acc}) and then continue to travel at S_{max} until it reaches the intersection, which is given by the following equation (according to Newton's equations of motion),

$$\min t_{N,int}(k) = \frac{S_{max} - S_N(k)}{a_{Acc}} + \frac{1}{S_{max}}\left[d_{N,int}(k) - \frac{(S_{max})^2 - S_N^2(k)}{2a_{Acc}}\right] \quad (2)$$

First part of the above equation gives the minimum time required by the N^{th} CV to accelerate from $S_N(k)$ to S_{max}, and the second part of the equation gives the time required by the N^{th} CV to reach the intersection at a constant speed (i.e., S_{max}) after it achieves S_{max}. Thus, (2) estimates the minimum time required by the N^{th} CV of the platoon to reach the intersection. We explain how the time spent in constant speed is obtained for the leader CV of a platoon in the next subsection.

Table 1. Symbols used in the speed advisory algorithm

Symbol	Meaning	Symbol	Meaning
L	Subscript L refers to the leader CV of a platoon	$d_{i,int}(k)$	Distance from the i^{th} follower CV in a platoon to the target intersection at the k^{th} timestamp
i	Subscript i refers to the i^{th} follower CV in a platoon, i.e., $i \in \{1,2,3, \dots \dots N\}$ for a platoon of $(N+1)$ CVs consisting one leader CV and N follower CVs	$d_{L,int}(k)$	Distance from the leader CV of a platoon to the target intersection calculated at the k^{th} timestamp
$x_i(k)$	Location of the i^{th} follower CV at the k^{th} timestamp with respect to the target intersection	$d_{L,constAcc}(k)$	Estimated (at the k^{th} timestamp) distance covered by the leader CV of a platoon while accelerating from $S_L(k)$ to achieve a target speed ($S_{L,tar}(k)$)
$x_L(k)$	Location of the leader CV at the k^{th} timestamp with respect to the target intersection	$d_{L,constSpd}(k)$	Estimated (at the k^{th} timestamp) distance covered by the leader CV of a platoon while operating at a target speed ($S_{L,tar}(k)$) from the moment it achieves $S_{L,tar}(k)$
l_{CV}	Length of the CV	$t_{N,int}(k)$	Estimated total time required (from the k^{th} timestamp) by the last (i.e., N^{th}) follower CV of a platoon to reach the intersection from its location ($x_N(k)$)
$g_i(k)$	Gap between the i^{th} follower CV at the k^{th} timestamp with its immediate leading CV	$t_{L,constAcc}(k)$	Estimated time required by the leader CV of a platoon from the k^{th} timestamp to accelerate from $S_L(k)$ to a target speed ($S_{L,tar}(k)$)
$g_{i,tar}(k)$	Target gap of the i^{th} follower CV at the k^{th} timestamp		
g_{stand}	Constant standstill gap		
T_g	Constant time gap	$t_{L,constSpd}(k)$	Time required (estimated at the k^{th} timestamp) by the leader CV of a platoon to reach the intersection while operating at a target speed ($S_{L,tar}(k)$) from the moment it achieves $S_{L,tar}(k)$
$S_i(k)$	Speed of the i^{th} follower CV at the k^{th} timestamp		
$S_L(k)$	Speed of the leader CV of a platoon at the k^{th} timestamp		
$S_{i,adv}(k)$	Speed advisory for the i^{th} follower CV at the k^{th} timestamp	$t_{remain}(k)$	Remaining time of the current green interval calculated at the k^{th} timestamp
$S_{L,adv}(k)$	Speed advisory for the leader CV of a platoon at the k^{th} timestamp	$t_{avail}(k)$	Available time to pass an intersection calculated at the k^{th} timestamp
S_{max}	Maximum speed, which is same as the roadway speed limit		
a_{Acc}	Maximum acceleration	t_G	(Minimum) green interval
a_{Brk}	Maximum braking deceleration	t_{AR}	All red interval
a_{const}	Constant acceleration; $a_{const} = a_{Acc}$ if the CV is accelerating, and $a_{const} = a_{Brk}$ if the CV is decelerating	t_Y	Yellow interval
$delay_L(k)$	Additional estimated delay calculated from the k^{th} timestamp experienced by the leader CV of a platoon while following $S_{L,adv}(k)$ compared to following S_{max}		

We assume 100% CV penetration on the signalized corridor considered in this study. There are two cases to consider based on the current phase of the traffic signal at the target intersection that the CVs are approaching; case I: the platoon can pass the intersection within the current green interval, and case II: the platoon can pass the intersection in the next green interval. For case I, the available time to reach the intersection before the signal turns red is,

$$t_{avail}(k) = t_{remain}(k) \tag{3}$$

where, $t_{remain}(k)$ is the remaining green interval, whereas, for case II, this available time is an aggregate of the remaining green interval and the other intervals till the next green interval, i.e., sum of the minimum green intervals ($\sum t_G$) and yellow intervals ($\sum t_Y$) for the other approaches in the intersection, and sum of the all-red intervals ($\sum t_{AR}$);

$$t_{avail}(k) = t_{remain}(k) + \sum t_G + \sum t_Y + \sum t_{AR} \tag{4}$$

4.2 Speed Advisory for the Leader CVs of the Platoons

For the leader CV of a platoon, the speed advisory is determined based on whether the platoon is a case I platoon or a case II platoon. For the case I platoons, the speed advisory algorithm attempts to assist the CVs to cross the intersection as fast as possible while operating within the roadway speed limit, S_{max}. Therefore, for case I, the leader CVs are simply advised with the roadway speed limit, S_{max}, as the speed advisory. For the case II platoons, the speed advisories for the leader CVs are found through an optimization with an objective to reduce the estimated delay to pass the intersection.

For a case II platoon, our objective function of the optimization for determining the advisory speed for the leader CV is the estimated delay experienced by the leader CV while traveling from its current state till it reaches the target intersection. In this context, "delay" is estimated as the additional time required by the leader CV to reach the intersection using the advised speed, $S_{L,adv}$, compared to the lowest possible time to reach the intersection using the maximum speed, i.e., S_{max}, which is set to be the same as the speed limit. Thus, the objective function for this optimization is considered as this additional estimated delay for the leader CV, which is given by the following expressions,

$$\min_{S_{L,adv}} delay_L(k) \tag{5}$$

$$\text{where, } delay_L(k) = \left(t_{L,constAcc}(k) + t_{L,constSpd}(k)\right)_{for\ S_{L,adv}} - \left(t_{L,constAcc}(k) + t_{L,constSpd}(k)\right)_{for\ S_{max}} \tag{6}$$

$\left(t_{L,constAcc}(k) + t_{L,constSpd}(k)\right)_{for\ S_{L,adv}}$ and $\left(t_{L,constAcc}(k) + t_{L,constSpd}(k)\right)_{for\ S_{max}}$ both consist of two periods:

1) acceleration period, $t_{L,constAcc}(k)$: the time required to accelerate from the leader CV's current speed, $S_L(k)$, to $S_{L,adv}(k)$ or S_{max}; and
2) constant speed period, $t_{L,constSpd}(k)$: the time required to reach the intersection at a constant speed, $S_{L,adv}(k)$ or S_{max}, after achieving $S_{L,adv}(k)$ or S_{max}.

Here, we only discuss how to estimate the above two periods for $S_{L,adv}(k)$ as the same steps are followed to estimate the two periods for S_{max}. The required time to accelerate from $S_L(k)$ to $S_{L,adv}(k)$ is given by,

$$t_{L,constAcc}(k) = \frac{S_{L,adv} - S_L(k)}{a_{const}} \tag{7}$$

where, $a_{const} = a_{Acc}$ if $S_{L,adv}(k) > S_L(k)$, and $a_{const} = a_{Brk}$ if $S_{L,adv}(k) < S_L(k)$). Then, we estimate the distance covered during the acceleration period. Distance covered while accelerating from $S_L(k)$ to $S_{L,adv}(k)$,

$$d_{L,constAcc}(k) = \frac{\left(S_{L,adv}(k)\right)^2 - S_L^2(k)}{2a_{const}} \tag{8}$$

To determine $t_{L,constSpd}(k)$, first, we need to estimate the distance covered (i.e., $d_{L,constSpd}(k)$) while operating at a constant speed, $S_{L,adv}(k)$, which can be obtained by subtracting $d_{L,constAcc}(k)$ from the distance of the leader CV from the target intersection (i.e., $d_{L,int}(k)$),

$$d_{L,constSpd}(k) = d_{L,int}(k) - d_{L,constAcc}(k) = d_{L,int}(k) - \frac{\left(S_{L,adv}(k)\right)^2 - S_L^2(k)}{2a_{const}} \tag{9}$$

Now, we can estimate $t_{L,constSpd}(k)$ for $S_{L,adv}(k)$ as follows,

$$t_{L,constSpd}(k) = \frac{d_{L,constSpd}(k)}{S_{L,adv}(k)} = \frac{1}{S_{L,adv}(k)}\left[d_{L,int}(k) - \frac{\left(S_{L,adv}(k)\right)^2 - S_L^2(k)}{2a_{const}}\right] \tag{10}$$

Similarly, $t_{L,constAcc}(k)$ and $t_{L,constSpd}(k)$ for S_{max} can be written as follows,

$$t_{L,constAcc}(k) = \frac{S_{max} - S_L(k)}{a_{const}} \tag{11}$$

$$t_{L,constSpd}(k) = \frac{d_{L,constSpd}(k)}{S_{max}} = \frac{1}{S_{max}}\left[d_{L,int}(k) - \frac{(S_{max})^2 - S_L^2(k)}{2a_{const}}\right] \tag{12}$$

Therefore, we can now estimate the delay experienced by the leader CV while traveling from its current state until it reaches the target intersection by substituting the terms derived in (7), (10), (11) and (12) into (6),

$$delay_L(k) = \left(d_{L,int}(k) + \frac{S_L^2(k)}{2a_{const}}\right)\left[\frac{1}{S_{L,adv}(k)} - \frac{1}{S_{max}}\right] - \frac{S_{L,adv}(k) - S_{max}}{2a_{const}} \tag{13}$$

For this speed advisory optimization for the case II platoons' leader CV, we consider the following constraint,

$$S_{max} - 10\ mph \le S_{L,adv} \le UB \tag{14}$$

$$\text{where, } UB = \begin{cases} \min\left(S_{max}, \frac{d_{L,int}(k)}{t_{avail}(k)}\right)\ if\ \frac{d_{L,int}(k)}{t_{avail}(k)} \ge (S_{max} - 10\ mph) \\ (S_{max} - 10\ mph)\ if\ \frac{d_{L,int}(k)}{t_{avail}(k)} < (S_{max} - 10\ mph) \end{cases}$$

This constraint sets lower and upper bounds to the speed advisory for the case II platoons' leader CVs. The lower bound makes sure that the case II platoons' leader CVs are not advised speeds that are too low compared to the roadway speed limit. To ensure this, the lower bound is set to 10 miles per hour (mph) below the roadway speed limit, S_{max}. We chose this threshold to be 10 mph because a threshold less than 10 mph, for example, 5 mph below the speed limit) would leave a small window to select the advisory speeds, and a threshold greater than 10 mph, for example, 15 mph below the speed limit, might cause selecting advisory speeds that are too low compared to the roadway speed limit. On the other hand, the upper bound ensures that 1) the advised speeds do not exceed the roadway speed limit, S_{max}, and 2) the leader CVs do not arrive at the intersection early before the signal turns green again.

Note that, if $\frac{d_{L,int}(k)}{t_{avail}(k)} < (S_{max} - 10\ mph)$, then the leader CV needs to slow down to a speed that is lower than $(S_{max} - 10\ mph)$ to reach the intersection before the signal turns green again. However, as we mentioned above, an advised speed lower than $(S_{max} - 10\ mph)$ may seem too low considering the speed of other vehicles on the roadway and the drivers may not want to or able to follow that. In that case, the leader CV is advised a speed equal to $(S_{max} - 10\ mph)$, as this will be the only solution that meets the constraint (14). On the other hand, if $\frac{d_{L,int}(k)}{t_{avail}(k)} \geq (S_{max} - 10\ mph)$, then we set the minimum value between S_{max} and $\frac{d_{L,int}(k)}{t_{avail}(k)}$ as the upper bound, which leads the optimization to pick a solution that would minimize the delay defined in (13) by allowing the leader CV to operate at a speed within the speed limit so that it can arrive at the intersection when it would turn green again. Thus, the constraint defined in (14) helps to find speed advisory solutions for the leader CVs that would minimize the stopped delay by slowing the CVs down. On the other hand, the objective function defined in (13) pushes the advisory speed solutions toward the speed limit, S_{max} (note that, $S_{L,adv} = S_{max}$ yields $delay_L(k) = 0$ in (13)) and the optimization determines $S_{L,adv}$ that is optimum in terms of the above two opposing conditions.

As we mentioned before, this part of the algorithm (i.e., subsection 4.2 in this paper) runs in the "CV Platoon Assigner", a serverless process, i.e., AWS Lambda, as shown in Fig. 2. Once the CV Platoon Assigner assigns the CVs into platoons and determines the advisory speeds for the corresponding leader CVs, it saves the results into the Speed Advisory Database. Then, it invokes CV Platoon Optimizers (i.e., one CV Platoon Optimizer for one CV platoon) to run another algorithm of speed advisory optimization for the follower CVs in the platoons, which we explain in the following subsection.

4.3 Speed Advisory for the Follower CVs in the Platoons

While the leader CVs of the platoons are advised speeds to help the CVs quickly pass the intersection (for case I) or to reduce the stopped delay as much as possible (for case II), the follower CVs are advised speeds simply to reduce the gap among the follower CVs as much as possible without causing any safety issues, such as increased collision risks compared to the case when the CVs run without any advisory speeds. We do this using a discrete-time linear model predictive control (MPC)-based optimization algorithm that is solved globally to determine the speed advisories for all the follower CVs in each platoon at each time step. In this subsection, we discuss the detailed formulation of the MPC-based optimization for the follower CVs' speed advisories. Table 1 and Fig. 3 explain the relevant symbols that are used in this formulation.

First, we assume the advised speeds are achievable by the follower CVs in a platoon within a short period of time Δt based on the CVs' maximum acceleration, a_{Acc}, or deceleration, a_{Brk}, capabilities. Then, assuming constant acceleration or deceleration within this short period of time, Δt, we can write the following equations of motion for the i^{th} and the $(i+1)^{th}$ follower CVs in a platoon,

$$x_i(k+1) = x_i(k) + \left(\frac{S_i(k)+S_{i,adv}(k)}{2}\right)\Delta t = x_i(k) + u_i(k)\Delta t \tag{15}$$

$$\text{where, } u_i(k) = \left(\frac{S_i(k)+S_{i,adv}(k)}{2}\right) \tag{16}$$

$$\text{similarly, } x_{i+1}(k+1) = x_{i+1}(k) + u_{i+1}(k)\Delta t$$

Now, we estimate the gap $(g_{i+1}(k+1))$ for the $(i+1)^{th}$ follower CV with its immediate leading follower CV, i.e., the i^{th} follower CV, as,

$$g_{i+1}(k+1) = x_{i+1}(k+1) - x_i(k+1) - l_{CV} = g_{i+1}(k) + [u_{i+1}(k) - u_i(k)]\,\Delta t \tag{17}$$

In this algorithm, we assume that the lengths of all the CVs are the same, i.e., l_{CV} is the same for all the CVs. However, individual CV length can be used as well if the information is available. Note that, (16) stands for the control input that we seek from our MPC-based optimization. Once we obtain the control inputs, we can easily determine the speed advisories for the follower CVs from (16). Now, as (17) is applicable for all the follower CVs in a platoon, we can write it in an augmented matrix form as follows,

$$\boldsymbol{G}(k+1) = \boldsymbol{G}(k) + \boldsymbol{B}\boldsymbol{U}(k) \tag{18}$$

where, $\boldsymbol{G}(k+1) = \begin{bmatrix} g_L(k+1) \\ g_1(k+1) \\ g_2(k+1) \\ \dots \\ g_N(k+1) \end{bmatrix}_{(N+1)\times 1}$, $\boldsymbol{G}(k) = \begin{bmatrix} g_L(k) \\ g_1(k) \\ g_2(k) \\ \dots \\ g_N(k) \end{bmatrix}_{(N+1)\times 1}$,

$$\boldsymbol{B} = \begin{bmatrix} 0 & 0 & 0 & & 0 \\ -\Delta t & \Delta t & 0 & \cdots & 0 \\ 0 & -\Delta t & \Delta t & & 0 \\ & \vdots & & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & \Delta t \end{bmatrix}_{(N+1)\times(N+1)}, \quad \boldsymbol{U}(k) = \begin{bmatrix} u_L(k) \\ u_1(k) \\ u_2(k) \\ \dots \\ u_N(k) \end{bmatrix}_{(N+1)\times 1}$$

Note that, although the speed advisory for the leader CV in a platoon is not sought from this MPC-based optimization, we still include the leader CV in (18) because the gap associated with 1st follower CV in a platoon is calculated with respect to the leader CV of that platoon. However, as the leader CV does not have an immediate leading CV, the dynamics of its gap cannot be formulated as in (17). Therefore, all the entries of the first row of $\boldsymbol{B}$ are set to zeros and $g_L(k)$ is set to an arbitrary value. Thus, the gap for the leader CV, $g_L(k)$, will remain unchanged over the prediction horizon irrespective of whatever control inputs are chosen and it will not affect our MPC-based optimization.

To determine the follower CVs' target gap at each timestamp, we adopt the constant time gap (CTG) policy. In a CTG policy, all the follower CVs in a platoon are expected to maintain a constant time gap with their immediate leading CVs. Besides, in a platooning operation, CTG policy can help to reduce the collision risks by varying the target gap requirement based on the speed of the vehicles. In this study, we consider a two-second constant time gap, i.e., $T_g = 2$ seconds, with a two-meter standstill gap, i.e., $g_{stand} = 2$ meters [26, 27]. A standstill gap is a minimum gap to avoid the chance of collisions that all CVs must maintain, even if they come to a complete stop. Therefore, the target gap for the $(i+1)^{th}$ follower CVs in a platoon ($g_{(i+1),tar}(k+1)$) can be written as,

$$g_{(i+1),tar}(k+1) = S_{i+1}(k) \times T_g + g_{stand} \tag{19}$$

where, g_{stand} denotes constant standstill distance. As (19) can be written for all the follower CVs in a platoon, we can write them in an augmented form as follows,

$$\boldsymbol{G_{tar}}(k+1) = \boldsymbol{G_{tar}}(k) \tag{20}$$

where, $\boldsymbol{G_{tar}}(k+1) = \begin{bmatrix} g_{L,tar}(k+1) \\ g_{1,tar}(k+1) \\ g_{2,tar}(k+1) \\ \dots \\ g_{N,tar}(k+1) \end{bmatrix}_{(N+1)\times 1}$, and $\boldsymbol{G_{tar}}(k) = \begin{bmatrix} S_L(k) \times T_g + g_{stand} \\ S_1(k) \times T_g + g_{stand} \\ S_2(k) \times T_g + g_{stand} \\ \dots \\ S_N(k) \times T_g + g_{stand} \end{bmatrix}_{(N+1)\times 1}$

Now, we augment (18) and (20) to get the state dynamics for our MPC-based optimization,

$$\begin{bmatrix} \boldsymbol{G}(k+1) \\ \boldsymbol{G_{tar}}(k+1) \end{bmatrix} = \begin{bmatrix} \boldsymbol{G}(k) \\ \boldsymbol{G_{tar}}(k) \end{bmatrix} + \begin{bmatrix} \boldsymbol{B} \\ \boldsymbol{0}_{(N+1)\times(N+1)} \end{bmatrix} \boldsymbol{U}(k) \tag{21}$$

where, $\boldsymbol{0}_{(N+1)\times(N+1)}$ is an $(N+1)\times(N+1)$ dimensional matrix with all zero entries. We can rewrite (21) as,

$$\boldsymbol{X_a}(k+1) = \boldsymbol{A_a X_a}(k) + \boldsymbol{B_a U}(k) \tag{22}$$

where, $\boldsymbol{X_a}(k+1) = \begin{bmatrix} \boldsymbol{G}(k+1) \\ \boldsymbol{G_{tar}}(k+1) \end{bmatrix}$, $\boldsymbol{X_a}(k) = \begin{bmatrix} \boldsymbol{G}(k) \\ \boldsymbol{G_{tar}}(k) \end{bmatrix}$, $\boldsymbol{A_a} = \boldsymbol{I}_{2(N+1)\times 2(N+1)}$, and

$$\boldsymbol{B_a} = \begin{bmatrix} \boldsymbol{B} \\ \boldsymbol{0}_{(N+1)\times(N+1)} \end{bmatrix}$$

where, $\boldsymbol{I}_{2(N+1)\times 2(N+1)}$ is an $(N+1)\times(N+1)$ dimensional identity matrix. As with this MPC-based optimization, we want to adjust the gap among the follower CVs in a platoon based on the CTG policy, we define our measured variable as follows,

$$\boldsymbol{Y_a}(k) = \boldsymbol{G}(k) - \boldsymbol{G_{tar}}(k) = [\boldsymbol{I}_{(N+1)\times(N+1)} \quad -\boldsymbol{I}_{(N+1)\times(N+1)}] \begin{bmatrix} \boldsymbol{G}(k) \\ \boldsymbol{G_{tar}}(k) \end{bmatrix} \tag{23}$$

which can be rewritten as,

$$\boldsymbol{Y_a}(k) = \boldsymbol{C_a X_a}(k) \tag{24}$$

where, $\boldsymbol{C_a} = [\boldsymbol{I}_{(N+1)\times(N+1)} \quad -\boldsymbol{I}_{(N+1)\times(N+1)}]$

Now, we define our cost function for the optimization. In this case, we prefer a quadratic cost function as our aim is to minimize the difference between the current gap, $\boldsymbol{G}(k)$, and the target gap, $\boldsymbol{G_{tar}}(k)$, based on the CTG policy through the speed advisories. Therefore, the cost function for a single-step prediction horizon (as only one step is required to be predicted based on the state dynamics defined in (22)) can be written as,

$$\boldsymbol{J} = \boldsymbol{Y_a^T}(k)\boldsymbol{Y_a}(k) \tag{25}$$

Substituting $\boldsymbol{Y_a}(k)$ from (24) into (25), we get,

$$\boldsymbol{J} = \boldsymbol{X_a^T}(k)\boldsymbol{C_a^T}\boldsymbol{C_a}\boldsymbol{X_a}(k) = \boldsymbol{X_a^T}(k)\boldsymbol{P}\boldsymbol{X_a}(k) \tag{26}$$

where, $\boldsymbol{P} = \boldsymbol{C_a^T}\boldsymbol{C_a}$

Now, we move on to the constraints for this MPC-based optimization. In this case, we introduce constraints for the control inputs defined in (16) and the measured variables defined in (24). First, the follower CVs should never be advised with speeds that exceed the roadway speed limit, S_{max}, nor should they be advised negative speeds, which leads us to the following constraint,

$$0 \leq S_{i,adv}(k) \leq S_{max} \tag{27}$$

As each control input is defined as the average of each follower CV's current speed, $S_i(k)$, and advised speed, $S_{i,adv}(k)$, in (16), we can rewrite (27) in terms of the control input as follows,

$$\frac{S_i(k)}{2} \leq u_i(k) \leq \frac{1}{2}\ (S_i(k) + S_{max}) \tag{28}$$

Second, as mentioned before, we assume that the advised speeds are achievable by the follower CVs based on their maximum acceleration, a_{Acc}, or deceleration, a_{Brk}, capabilities. Therefore, we also have,

$$S_i(k) + a_{Brk}\Delta t \leq S_{i,adv}(k) \leq S_i(k) + a_{Acc}\Delta t \tag{29}$$

Again, we can rewrite (29) in terms of the control input for the i^{th} follower CV as,

$$S_i(k) + \frac{1}{2}a_{Brk}\Delta t \leq u_i(k) \leq S_i(k) + \frac{1}{2}a_{Acc}\Delta t \tag{30}$$

Then, we combine (28) and (30) to get a single equation of constraint for the control input of the i^{th} follower CV as,

$$\max\left(\frac{S_i(k)}{2},\left(S_i(k)+\frac{1}{2}a_{Brk}\Delta t\right)\right)\le u_i(k)\ \le \min\left(\frac{1}{2}(S_i(k)+S_{max}),\left(S_i(k)+\frac{1}{2}a_{Acc}\Delta t\right)\right) \tag{31}$$

We can write (31) into an augmented form as,

$$\boldsymbol{U_{low}}(k)\le \boldsymbol{U}(k)\le \boldsymbol{U_{high}}(k) \tag{32}$$

$$\text{where, } \boldsymbol{U_{low}}(k)=\begin{bmatrix}\max\left(\frac{S_L(k)}{2},\left(S_L(k)+\frac{1}{2}a_{Brk}\Delta t\right)\right)\\ \max\left(\frac{S_1(k)}{2},\left(S_1(k)+\frac{1}{2}a_{Brk}\Delta t\right)\right)\\ \max\left(\frac{S_2(k)}{2},\left(S_2(k)+\frac{1}{2}a_{Brk}\Delta t\right)\right)\\ \dots\\ \max\left(\frac{S_N(k)}{2},\left(S_N(k)+\frac{1}{2}a_{Brk}\Delta t\right)\right)\end{bmatrix}, \text{ and}$$

$$\boldsymbol{U_{high}}(k)=\begin{bmatrix}\min\left(\frac{1}{2}\ (S_L(k)+S^{max}),(S_L(k)+\frac{1}{2}a_{Acc}\Delta t)\right)\\ \min\left(\frac{1}{2}\ (S_1(k)+S^{max}),(S_1(k)+\frac{1}{2}a_{Acc}\Delta t)\right)\\ \min\left(\frac{1}{2}\ (S_2(k)+S^{max}),(S_2(k)+\frac{1}{2}a_{Acc}\Delta t)\right)\\ \dots\\ \min\left(\frac{1}{2}\ (S_N(k)+S^{max}),(S_N(k)+\frac{1}{2}a_{Acc}\Delta t)\right)\end{bmatrix}$$

Next, we introduce a lower bound for the measured variable, $\boldsymbol{Y_a}(k)$, due to safety considerations. As the optimized solution should not result in a situation where any of the follower CVs has a lower gap than its corresponding target gap based on the CTG policy, we write,

$$\boldsymbol{Y_a}(k)\ge \boldsymbol{0}_{(N+1)\times 1} \tag{33}$$

Now, we have all the necessary equations formulated that we need for our MPC-based speed advisory optimization for the follower CVs in a platoon. As our linear MPC formulation includes a quadratic cost function (as given in (26)), we utilize a Python-based open-source solver, i.e., CVXOPT [28], for solving quadratic programming problems to run this MPC-based optimization. As mentioned before, this part of the Serverless CloSA, i.e., subsection 4.3, runs in the CV Platoon Optimizer shown in Fig. 2.

5 CASE STUDY

We conduct three case studies for different traffic conditions by developing a cloud-in-the-loop simulation testbed to evaluate the feasibility of the Serverless CloSA at a system level. In addition, we compare the results obtained from the simulation with and without the Serverless CloSA to evaluate the performance improvement in terms of stopped delay of the CVs at the signalized intersections, total travel time of the CVs through the signalized roadway section, and an aggregated collision risk indicator. We also evaluate the communication and processing delays for running the Serverless CloSA application to evaluate the feasibility of our cloud-based speed advisory application in terms of latency requirement of this CV mobility application.

5.1 Cloud-in-the-loop Simulation

We use an open-source microscopic traffic simulator called Simulation Urban Mobility (SUMO) [11] to simulate a section of a roadway including traffic signals and CVs operating in the roadway section. In our cloud-in-the-loop simulation, AWS services (residing in the cloud) are integrated with SUMO (running in a local machine) to evaluate the Serverless CloSA (as shown in Fig. 4). Traffic Control Interface (TraCI) [29] is a Python-based interface compatible with SUMO. As Fig. 4 shows, we use TraCI to extract BSMs (e.g., CVs' location and motion information) and signal phase and timing messages (e.g., current signal interval, remaining green time) from the CVs and the traffic signals, respectively. Data collected from the simulation are packaged and transferred to the AWS cloud through different AWS services, i.e., DynamoDB and KDS, via LTE communication. In the cloud, each KDS triggers a Serverless CV Advisory Cluster, as mentioned before. Each Serverless CV advisory Cluster gets CV trajectory information from the CV Trajectory Database. Each Serverless CV Advisory Cluster also collects and updates the distances of the CVs from its corresponding traffic signal (as shown in

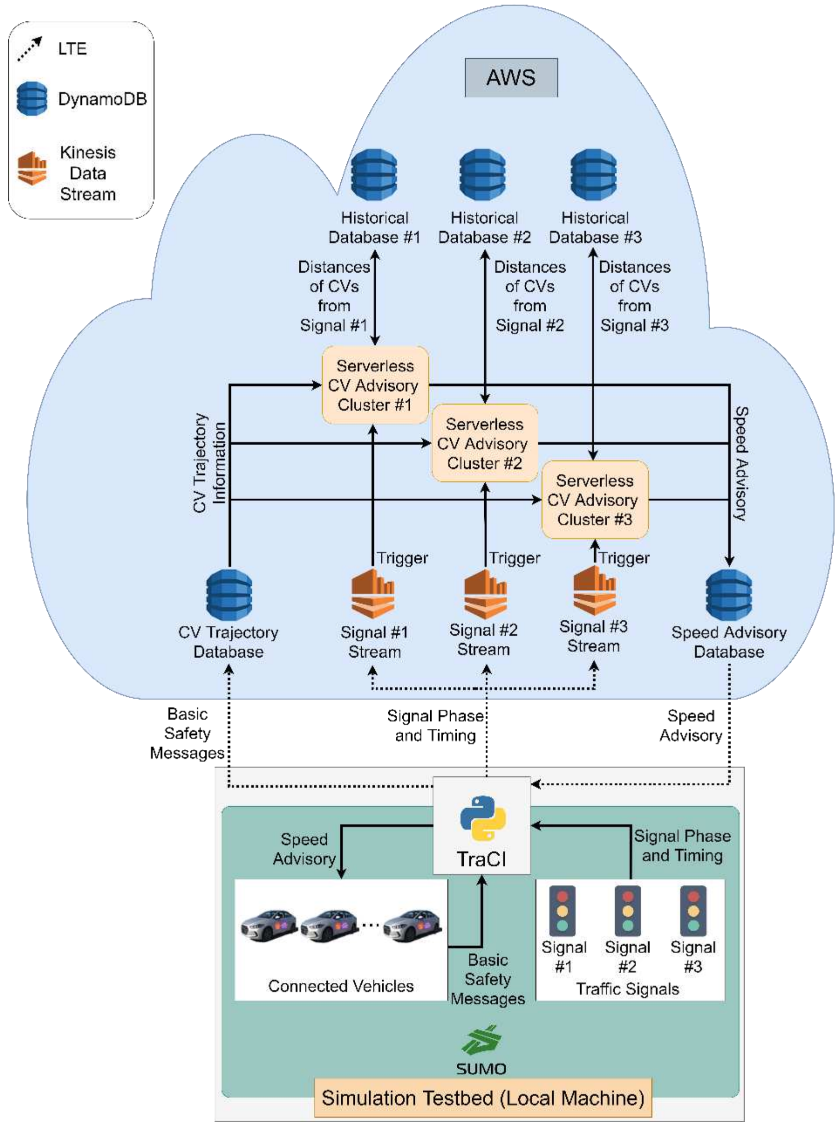


Fig. 4. Dataflow in the cloud-in-the-loop simulation.

Fig. 4). Inside the clusters, CV platoon identification and speed advisory optimization algorithms run using serverless processes (details are mentioned in section 3) and the results of the optimizations, i.e., the speed advisories) are saved in the Speed Advisory Database. Then, SUMO can collect the speed advisories via LTE and assign the speed advisories to the CVs through TraCI.

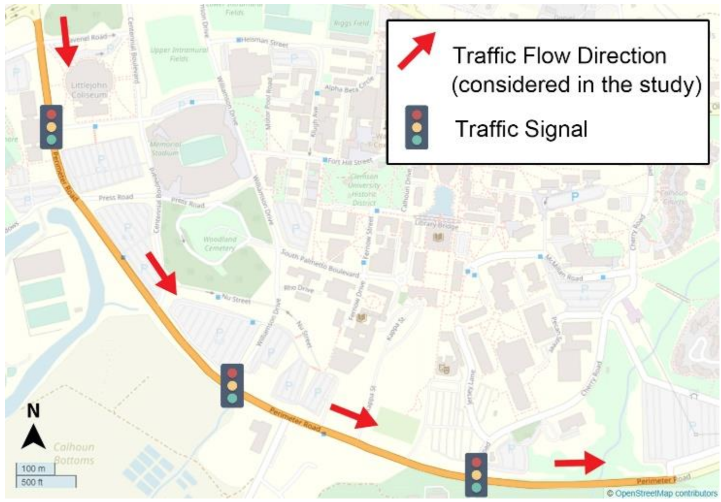


Fig. 5. Route location and layout.

In Fig. 5, the simulated roadway is shown in orange-colored line, which is a 1.5-mile-long 4-lane highway (2 lanes in each direction) with three traffic signals in Clemson, South Carolina, and it is a part of a CV deployment site known as South Carolina Connected Vehicle Testbed (SC-CVT) [30]. By defining the traffic flows in SUMO configuration [31], we generated 50 CVs on the simulated roadway in three different traffic densities, i.e., low, medium, and high traffic densities. SUMO allows controlling the time interval within which a given number of vehicles will be generated, which we use here to create the different traffic densities. Here, low traffic density stands for 633 passenger cars per hour per lane (pc/h/ln), which is 33% of the traffic capacity, i.e., 1900 pc/h/ln, medium traffic density stands for 1267 pc/h/ln, i.e., 66% of traffic capacity, and high traffic density stands for 1900 pc/h/ln, i.e., full traffic capacity, based on the base saturation flowrate defined in [32]. All CVs operate within a roadway speed limit of 35 mph, which is already included in the map data. For each condition, we evaluate two scenarios in the simulation: 1) the baseline scenario, i.e., no speed advisory, and 2) the Serverless CloSA-deployed scenario. For each traffic density defined above, we run the simulation five times with randomly generated CVs.

5.2 Evaluation Results and Discussions

To evaluate Serverless CloSA's performance, we compare three measures of effectiveness (MoEs): 1) stopped delay at the signalized intersections of the simulated roadway, 2) total travel time to pass the simulated roadway section, and 3) time-integrated time-to-collision (TTC) or TIT. Stopped delay and travel time are MoEs related to traffic flow, whereas TIT is a widely used surrogate measure for evaluating collision risks that integrates the TTC profile below a predefined threshold (i.e., TTC threshold, TTC^*) over time for all the CVs under collision risk evaluation. Details for calculating TIT can be found in [33, 34]. For our study, TIT for the i^{th} CV (i.e., TIT_i) can be calculated using the following equation,

$$TIT_i = \sum_t [TTC^* - TTC_i(t)], \;\; \forall 0 \le TTC_i(t) \le TTC^* \tag{34}$$

$$\text{where, } TTC_i(t) = \begin{cases} \frac{g_i(t)}{S_i(t) - S_{i-1}(t)} & if\ S_i(t) > S_{i-1}(t) \\ \infty & if\ S_i(t) \le S_{i-1}(t) \end{cases} \tag{35}$$

Here, t represents a timestamp, $g_i(t)$ represents the gap between the i^{th} CV (i.e., a follower CV) and the $(i-1)^{th}$ CV (i.e., immediate leading CV of the i^{th} CV) at t, and $S_i(t)$ and $S_{i-1}(t)$ represent the speeds of the i^{th} and the $(i-1)^{th}$ CVs at t, respectively. As observed from (34) and (35), the risk of collision is only considered when the follower CV has a higher speed compared to its immediate leading CV. Once TIT for all the CVs is calculated using (35), we sum them up to get the aggregated TIT for all the CVs within the simulation run time. In this paper, we use a TTC^* of 2 seconds based on the time headway requirement in our MPC-based optimization for determining the advisory speeds of the follower CVs. A TTC^* of 2 seconds means that whenever the time gap between any two successive CVs is measured to be less than or equal to 2 seconds, the risk of collision is considered in calculating TIT.

Fig. 6 shows box chart comparisons between our Serverless CloSA and the baseline "no speed advisory" scenario in terms of stopped delay and total travel time for three different traffic conditions, i.e., low, medium, and high-density traffic. As observed from Fig. 6(a), Serverless CloSA reduced the stopped

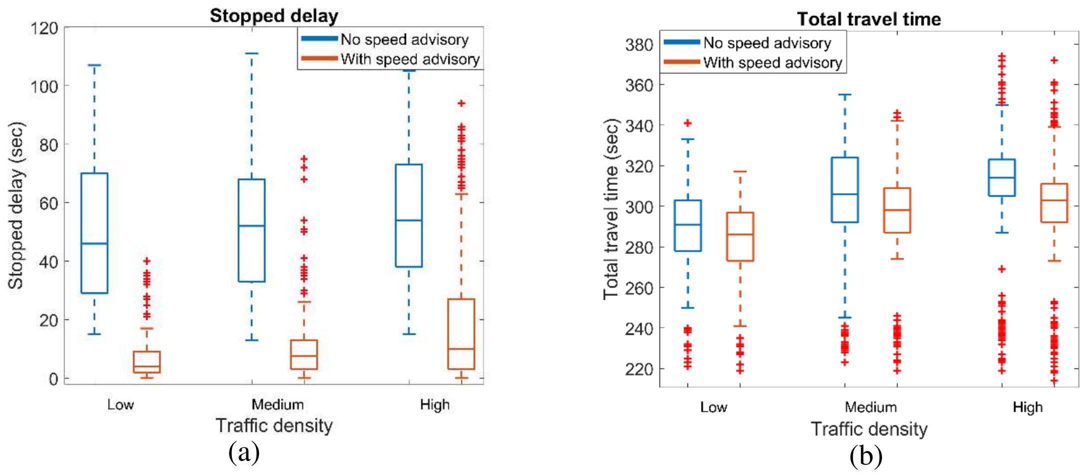


Fig. 6. Box chart comparisons for (a) stopped delay and (b) total travel time.

delay significantly for all three roadway traffic conditions, i.e., low, medium and high-density roadway traffic, compared to the baseline “no speed advisory” scenario. In Fig. 6(b), we observe a small reduction in the total travel time when using Serverless CloSA for providing speed advisories to the CVs as compared to the “no speed advisory” case. This is not unexpected because our speed advisory optimization aims to reduce the stopped delay, not the travel time. While it may seem that a reduction in the stopped delay should cause a reduction in the travel time as well, it may not be the case all the time [35]. For example, note that although the Serverless CloSA reduces the total stopped delay for the CVs significantly, it cannot entirely remove the stopped delay and the CVs may have to stop at the intersections for some time. Then, these CVs would have to start from a stopped condition when the signal turns green again in which case the benefit of having no startup lost time is not achievable. Also, our Serverless CloSA does not advise CVs with speeds considering that they can pass the intersection within the yellow interval. On the other hand, in the “no speed advisory” case, the CVs have no such conditions imposed on them. Thus, reducing the total travel time is not always guaranteed for all the CVs while using Serverless CloSA.

Table 2 presents the effectiveness of Serverless CloSA in terms of percentage reduction of the MoEs on average for each CV in the simulation. We observe that the maximum reduction of the stopped

Table 2. Average (per CV) reduction of the MOEs for Serverless CloSA

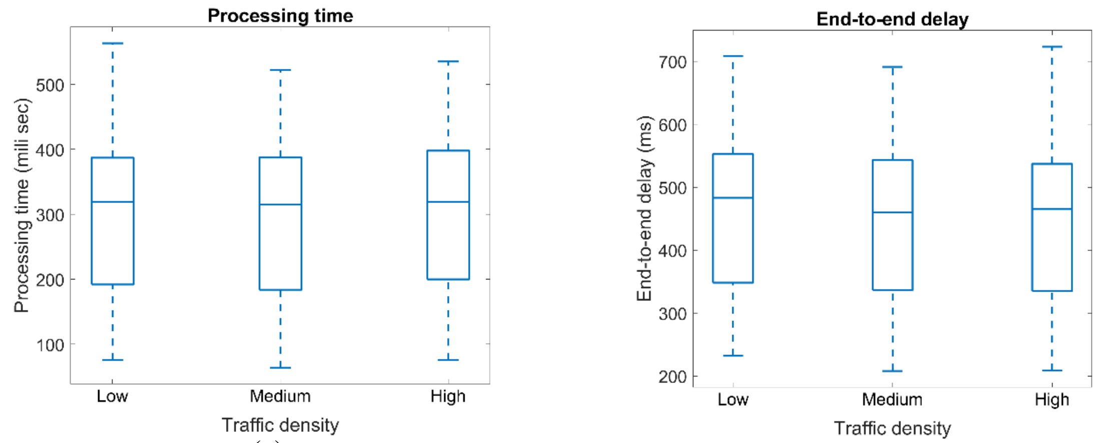
	Traffic Density			Average of Low, Medium, and High Traffic Densities
	Low	Medium	High	
Average reduction in stopped delay	85%	80%	65%	77%
Average reduction in total travel time	2%	3%	4%	3%
Average reduction in TIT	24%	16%	23%	21%

delay, i.e., about 85%, was possible for low traffic density. In terms of reducing the total travel time, Serverless CloSA's performance did not vary much based on the different traffic conditions. We also observe that Serverless CloSA is most effective in reducing the average per CV TIT, i.e., about 24%, for low-density traffic condition.

We also evaluate the end-to-end delay to assess the feasibility of the Serverless CloSA as a real-time CV application. The end-to-end delay is calculated using the following equation,

$$\textit{end-to-end delay} = \textit{upload delay} + \textit{processing delay} + \textit{download delay} \tag{36}$$

Fig. 7 presents the processing time, and the end-to-end delay reported during our experiments using box charts and Table 3 provides the averages of the processing time and the end-to-end delay for each CV in the cloud for the three traffic density conditions. From Table 3, the end-to-end delay is about 452 ms (on average for all three traffic density conditions), which meets the requirement of a real-time CV mobility application, i.e., maximum allowable delay of 1000 ms [10, 25]. Besides, we observe from Fig. 7 and Table 3 that the processing delays and the end-to-end delays do not vary much across the various traffic densities, which indicates the scalability of the Serverless CloSA.



(a) (b)

Fig. 7. Box charts of (a) processing time, and (b) end-to-end delay.

Table 3. Average (per CV) reduction of the MOEs for Serverless CloSA

Delays	**Traffic Density**			**Average of the Three Traffic Densities**	**Allowable Delay**
	Low	Medium	High		
Average reduction in stopped delay	298	297	303	299	
Average reduction in total travel time	463	447	446	452	<1000 ms

6 CONCLUSION

In this paper, we develop a highly scalable serverless cloud computing architecture using Amazon Web Services (AWS) to support the requirements of real-time CV mobility applications. Then, we develop an optimization-based real-time CV speed advisory algorithm, i.e., Serverless CloSA, which is deployable using the serverless cloud-based architecture that we developed. The Serverless CloSA assists CVs to pass through a signalized corridor with speed advisories that can help reduce the stopped delay experienced by these CVs at the signalized intersections of that corridor. We conduct case studies for a signalized corridor for three different roadway traffic conditions (low, medium, and high-density roadway traffic) with a cloud-in-the-loop simulation testbed using AWS and Simulation of Urban Mobility (SUMO), which is an open-source microscopic roadway traffic simulator, to evaluate the feasibility and performance of the Serverless CloSA at a system level. Based on the evaluation results, we conclude that Serverless CloSA is effective in reducing the average stopped delay at the signalized intersections of a corridor by 77% while reducing the risk of collision and the total travel time for the CVs through that corridor when compared to the baseline "no speed advisory" scenario.

Generally, the state departments of transportation (DOTs) deploy transportation applications based on traditional server infrastructure in their traffic management centers (TMCs), which requires significant investments in computing and human resources. This study shows that cloud infrastructure offers a promising alternative for addressing the computing infrastructure needs for CV mobility applications. The commercial cloud-based CV mobility application strategy could potentially lower the costs associated with computing equipment installation, configuration, operation, and maintenance, without sacrificing any performance and reliability.

Authors Contribution

H.-W. Deng, M.S. Salek, M. Rahman and M. Chowdhury wrote the main manuscript text with the help of A. Apon and M. Shue. H.-W. Deng, M.S. Salek, M. Rahman, M. Chowdhury, A. Apon and M. Shue conceived and planned the simulation experiments. H.-W. Deng and M.S. Salek carried out the simulations. H.-W. Deng, M.S. Salek prepared all the figures. All authors critically reviewed the manuscript and provided feedback to make the manuscript better.

Funding Declaration

None.

Conflict of Interest

None.

REFERENCES

1. Deka L, Chowdhury M (2018) Transportation Cyber-Physical Systems, 1st ed. Elsevier

2. Omoniwa B, Hussain R, Javed MA, et al (2019) Fog/Edge Computing-Based IoT (FECIoT): Architecture, Applications, and Research Issues. IEEE Internet of Things Journal 6:4118–4149. https://doi.org/10.1109/JIOT.2018.2875544

3. Xu Z, Li X, Zhao X, et al (2017) DSRC versus 4G-LTE for Connected Vehicle Applications: A Study on Field Experiments of Vehicular Communication Performance. In: Journal of Advanced Transportation. https://www.hindawi.com/journals/jat/2017/2750452/. Accessed 19 Jan 2021

4. Deng H-W, Rahman M, Chowdhury M, et al (2021) Commercial Cloud Computing for Connected Vehicle Applications in Transportation Cyberphysical Systems: A Case Study. IEEE Intelligent Transportation Systems Magazine 13:6–19. https://doi.org/10.1109/MITS.2020.3037314

5. What is AWS Lambda? - AWS Lambda. https://docs.aws.amazon.com/lambda/latest/dg/welcome.html. Accessed 6 Oct 2020

6. Azure Functions Serverless Compute | Microsoft Azure. https://azure.microsoft.com/en-us/services/functions/. Accessed 6 Oct 2020

7. What is Amazon EC2? - Amazon Elastic Compute Cloud. https://docs.aws.amazon.com/AWSEC2/latest/WindowsGuide/concepts.html. Accessed 4 Mar 2021

8. Rajan RAP (2018) Serverless Architecture - A Revolution in Cloud Computing. In: 2018 Tenth International Conference on Advanced Computing (ICoAC). pp 88–93

9. Shafiei H, Khonsari A, Mousavi P (2020) Serverless Computing: A Survey of Opportunities, Challenges and Applications

10. Islam M, Rahman M, Khan SM, et al (2020) Development and Performance Evaluation of a Connected Vehicle Application Development Platform. Transportation Research Record 2674:537–552. https://doi.org/10.1177/0361198120917146

11. Lopez PA, Behrisch M, Bieker-Walz L, et al (2018) Microscopic Traffic Simulation using SUMO. In: 2018 21st International Conference on Intelligent Transportation Systems (ITSC). pp 2575–2582

12. Ning Z, Huang J, Wang X (2019) Vehicular Fog Computing: Enabling Real-Time Traffic Management for Smart Cities. IEEE Wireless Communications 26:87–93. https://doi.org/10.1109/MWC.2019.1700441

13. Li X, Dang Y, Aazam M, et al (2020) Energy-Efficient Computation Offloading in Vehicular Edge Cloud Computing. IEEE Access 8:37632–37644. https://doi.org/10.1109/ACCESS.2020.2975310

14. Jin J, Ran B, Chen T, et al (2020) Cloud-based technology for connected and automated vehicle highway systems

15. Bradaï B, Garnault A, Picron V, Gougeon P (2016) A Green Light Optimal Speed Advisor for Reduced CO2 Emissions. In: Langheim J (ed) Energy Consumption and Autonomous Driving. Springer International Publishing, Cham, pp 141–151

16. Stebbins S, Kim J, Hickman M, Vu HL (2016) Combining model predictive intersection control with green light optical speed advisory in a connected vehicle environment. In: Australasian Transport Research Forum 2016 Proceedings

17. Stebbins S, Hickman M, Kim J, Vu HL (2017) Characterising Green Light Optimal Speed Advisory trajectories for platoon-based optimisation. Transportation Research Part C: Emerging Technologies 82:43–62. https://doi.org/10.1016/j.trc.2017.06.014

18. Suzuki H, Marumo Y (2018) A New Approach to Green Light Optimal Speed Advisory (GLOSA) Systems for High-Density Traffic Flowe. In: 2018 21st International Conference on Intelligent Transportation Systems (ITSC). pp 362–367

19. Pariota L, Costanzo LD, Coppola A, et al (2019) Green Light Optimal Speed Advisory: a C-ITS to improve mobility and pollution. In: 2019 IEEE International Conference on Environment and Electrical Engineering and 2019 IEEE Industrial and Commercial Power Systems Europe (EEEIC / I CPS Europe). pp 1–6

20. Zhang Z, Zou Y, Zhang X, Zhang T (2020) Green Light Optimal Speed Advisory System Designed for Electric Vehicles Considering Queuing Effect and Driver’s Speed Tracking Error. IEEE Access 8:208796–208808. https://doi.org/10.1109/ACCESS.2020.3037105

21. Zhao W, Ngoduy D, Shepherd S, et al (2018) A platoon based cooperative eco-driving model for mixed automated and human-driven vehicles at a signalised intersection. Transportation Research Part C: Emerging Technologies 95:802–821. https://doi.org/10.1016/j.trc.2018.05.025

22. Serverless Computing – Amazon Web Services. In: Amazon Web Services, Inc. https://aws.amazon.com/products/databases/. Accessed 29 Mar 2021

23. What Is Amazon DynamoDB? - Amazon DynamoDB. https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/Introduction.html. Accessed 6 Oct 2020

24. Amazon Kinesis Data Streams - Data Streaming Service - Amazon Web Services. In: Amazon Web Services, Inc. https://aws.amazon.com/kinesis/data-streams/. Accessed 26 Oct 2020

25. Fehr W (2014) Southeast Michigan Test Bed: 2014 Concept of Operations

26. Park B (Brian), Schneeberger JD (2003) Microscopic Simulation Model Calibration and Validation: Case Study of VISSIM Simulation Model for a Coordinated Actuated Signal System. Transportation Research Record 1856:185–192. https://doi.org/10.3141/1856-20

27. Mahmod M, Arem B van, Pueboobpaphan R, Lange R de (2013) Reducing local traffic emissions at urban intersection using ITS countermeasures. IET Intelligent Transport Systems 7:78–86. https://doi.org/10.1049/iet-its.2011.0222

28. CVXOPT. https://cvxopt.org/. Accessed 19 Jan 2021

29. Wegener A, Piórkowski M, Raya M, et al (2008) TraCI: an interface for coupling road traffic and network simulators. In: Proceedings of the 11th communications and networking simulation symposium. Association for Computing Machinery, New York, NY, USA, pp 155–163

30. Chowdhury M, Rahman M, Rayamajhi A, et al (2018) Lessons Learned from the Real-World Deployment of a Connected Vehicle Testbed. Transportation Research Record 2672:10–23. https://doi.org/10.1177/0361198118799034

31. Flow Definitions. In: Shortest or Optimal Path Routing - SUMO Documentation. https://sumo.dlr.de/docs/Demand/Shortest_or_Optimal_Path_Routing.html#flow_definitions. Accessed 4 Mar 2021

32. Manual HC (2010) HCM2010. Transportation Research Board, National Research Council, Washington, DC 1207

33. Shi X, Wong YD, Li MZF, Chai C (2018) Key risk indicators for accident assessment conditioned on pre-crash vehicle trajectory. Accident Analysis & Prevention 117:346–356. https://doi.org/10.1016/j.aap.2018.05.007

34. Minderhoud MM, Bovy PHL (2001) Extended time-to-collision measures for road traffic safety assessment. Accident Analysis & Prevention 33:89–97. https://doi.org/10.1016/S0001-4575(00)00019-1

35. Eckhoff D, Halmos B, German R (2013) Potentials and limitations of Green Light Optimal Speed Advisory systems. In: 2013 IEEE Vehicular Networking Conference. pp 103–110