Transportation Research Record

Development and Performance Evaluation of a Connected Vehicle Application Development Platform (CVDeP)

--Manuscript Draft--

Full Title:	Development and Performance Evaluation of a Connected Vehicle Application Development Platform (CVDeP)
Abstract:	Connected vehicle (CV) application developers need a development platform to build, test and debug real-world CV applications, such as safety, mobility, and environmental applications, in edge-centric cyber-physical systems. Our study objective is to develop and evaluate a scalable and secure CV application development platform (CVDeP) that enables application developers to build, test and debug CV applications in real-time. CVDeP ensures that the functional requirements of the CV applications meet the corresponding requirements imposed by the specific applications. We evaluated the efficacy of CVDeP using two CV applications (one safety and one mobility application) and validated them through a field experiment at the Clemson University Connected Vehicle Testbed (CU-CVT). Analyses prove the efficacy of CVDeP, which satisfies the functional requirements (i.e., latency and throughput) of a CV application while maintaining scalability and security of the platform and applications.
Manuscript Classifications:	Operations and Traffic Management; Intelligent Transportation Systems AHB15; Vehicles and Equipment; Intelligent Transportation Systems AHB15
Manuscript Number:	20-05605
Article Type:	Presentation and Publication
Order of Authors:	Mhafuzul Islam
	Mizanur Rahman, PhD
	Sakib Mahmud Khan, PhD
	Mashrur Chowdhury, PhD
	Lipika Deka, PhD

Development and Performance Evaluation of a Connected Vehicle Application Development Platform (CVDeP)

Mhafuzul Islam*
Ph.D. Student
Glenn Department of Civil Engineering, Clemson University
351 Fluor Daniel Engineering Innovation Building, Clemson, SC 29634
Tel: (864) 986-5446, Fax: (864) 656-2670
Email: mdmhafi@clemson.edu

Mizanur Rahman, Ph.D.
Postdoctoral Fellow
Center for Connected Multimodal Mobility (C^2M^2)
Glenn Department of Civil Engineering, Clemson University
127 Lowry Hall, Clemson, SC 29634
Tel: (864) 650-2926; Email: mdr@clemson.edu

Sakib Mahmud Khan, Ph.D.
Glenn Department of Civil Engineering, Clemson University
351 Fluor Daniel Engineering Innovation Building, Clemson, SC 29634
Tel: (864) 569-1082, Fax: (864) 656-2670
Email: sakibk@g.clemson.edu

Mashrur Chowdhury, Ph.D., P.E., F. ASCE
Eugene Douglas Mays Professor of Transportation
Clemson University
Glenn Department of Civil Engineering
216 Lowry Hall, Clemson, South Carolina 29634
Tel: (864) 656-3313   Fax: (864) 656-2670
Email: mac@clemson.edu

Lipika Deka, Ph.D.
Assistant Professor
School of Computer Science and Informatics,
De Montfort University, Leicester, UK
Tel: +441162506051
Email: lipika.deka@dmu.ac.uk

*Corresponding author
Word count: 6964 words text + 2 table x 250 words (each) = 7464 words
Submission date: August 1, 2019

ABSTRACT

Connected vehicle (CV) application developers need a development platform to build, test and debug real-world CV applications, such as safety, mobility, and environmental applications, in edge-centric cyber-physical systems. Our study objective is to develop and evaluate a scalable and secure CV application development platform (CVDeP) that enables application developers to build, test and debug CV applications in real-time. CVDeP ensures that the functional requirements of the CV applications meet the corresponding requirements imposed by the specific applications. We evaluated the efficacy of CVDeP using two CV applications (one safety and one mobility application) and validated them through a field experiment at the Clemson University Connected Vehicle Testbed (CU-CVT). Analyses prove the efficacy of CVDeP, which satisfies the functional requirements (i.e., latency and throughput) of a CV application while maintaining scalability and security of the platform and applications.

Keywords: Connected vehicle, Connected vehicle applications, Development platform, Testbed, Cyber-physical systems

INTRODUCTION

The emerging connected vehicle (CV) environment consists of different components, such as vehicle onboard units (OBUs), and roadside units (RSUs) which are capable of exchanging data with each other as well as communicating with personal devices (e.g., cell phone), sensors (e.g., camera sensors), and traffic management centers (TMCs). With integrated computing and/or control capabilities, these connected physical components communicate with each other to form a cyber-physical system (CPS). Considering a large-scale deployment of connected vehicle CPS, the concept of edge computing is introduced as the underlying computing approach. Edge computing has the potential benefits for enabling reduced communicational latency and increased scalability. Such benefits are a result of bringing resources such as storage, and computational resources closer to the edge and consumers (*1*)(*2*). In an edge-centric CPS, the resources for communication, computation, control, and storage are placed at different edge layers (e.g., mobile edge as a vehicle, fixed edge as a roadside infrastructure, system edge as a backend server or TMC) in a CV environment (*3*). Therefore, a CV application can be divided into sub-applications where different sub-applications run in different edge layers depending on the requirements of the application.

Architecture reference for cooperative and intelligent transportation (ARC-IT), which has been developed by the U.S. Department of Transportation (USDOT), has listed and provided guidelines for planning and implementation of over a hundred CV applications for safety, mobility and environmental benefits (*4*). For example, 'Traffic data collection for traffic operations' is a CV application, which uses CV data obtained from OBUs to support traffic operations. To develop such CV applications for such an edge-centric CPS, developers need a dedicated platform where they can build, test and debug CV applications. The operational data environment (ODE) system, which is being developed by Intelligent Transportation Systems Joint Program Office (*5*), is a real-time data collection and distribution software system that collects, processes and distributes data to different components of the CV environment, such as CVs themselves, personal mobile devices, infrastructure components (e.g., traffic signal) and sensors (e.g., camera, environmental sensor). According to the architecture of the ODE, CV application developers can stream data using ODE in real-time. However, this system does not provide application developers an opportunity for building, testing and debugging CV applications. Thus, it is critical to develop an application development platform and evaluate the platform in terms of latency and throughput to satisfy the temporal and spatial requirements of CV applications (*6*).

Major challenges for developing a CV application development platform for an edge-centric CPS are to (a) enable developers to collect, process and distribute data, while running multiple CV applications concurrently in real-time in different edge layers; and (b) ensure security of the platform and application while maintaining the scalability of the platform. In fact, the same challenges are true for a deployed edge-centric CPS for CV applications. Hence, the objective of this study is to develop and evaluate a scalable and secure CV application development platform that handles real-time data from CVs in an edge-centric CPS and can satisfy the requirements imposed by CV applications. This platform, which we call 'Connected vehicle application development platform (CVDeP)' has been designed to hide the underlying low-level software, hardware, and associated details by providing access via an abstraction layer. An application development graphical user interface (GUI) layer will provide developers an easy and secure access to the edge devices. Security of the platform is guaranteed by securing access of the developers to the platform, in addition to maintaining application security. However, developing security policies for detecting cyberattacks and identifying related countermeasures are not the focus of this study.

A case study has been conducted to evaluate the efficacy of CVDeP using a safety application (i.e., Forward collision warning) and a mobility application (i.e., Traffic data collection for traffic operation) (*4*). These applications were developed and evaluated on the CVDeP emulated environment and later validated in a real-world edge-centric Clemson University Connected Vehicle Testbed (CU-CVT), which is located at Clemson, South Carolina. ‘Forward collision warning’ application has been selected as it is a fundamental application for vehicle-to-vehicle (V2V) safety. Similarly, ‘Traffic data collection for traffic operation’ application has been selected for the case study, because this application supports many other vehicle-to-infrastructure (V2I) safety and mobility applications, such as cooperative adaptive cruise control, incident detection and implementation of localized operational strategies (e.g., altering signal timing based on traffic flows, freeway speed harmonization and optimization of ramp metering rates). The efficacy of the CVDeP was evaluated using two communication-related measures of effectiveness (i.e., latency and throughput).

RELATED WORK

In order to develop the CVDeP that uses real-time CV data, we reviewed existing work related to the CV applications development criteria, and developer access control and application security.

CV Application Development Requirements

CV applications are bounded by time and space requirements for providing the desired service (*7*). If CV data are not received within the temporal and spatial threshold as required by specific CV applications, CV data will not have any efficacy for real-time applications. The Michigan connected vehicle testbed ‘Proof of concept test report’ categorized CV data by time and space contexts (*8*). While streaming data, timestamp information and location should be included in the CV data as such data are included in the basic safety message (BSM) sets, and they support data validity checks. In addition, data disseminated by the application development platform must be consistent and error-free (*9*).

Application developers may require two kinds of data depending on the application, namely real-time disaggregated data and aggregated data. For example, applications such as incident detection applications require real-time disaggregated data for running and testing of algorithms (*6*), thus making it necessary for the platform to provide such data. On the other hand, applications such as those that provide queue warning after every 5 minutes (*10*) may not require the raw data, but aggregated data is sufficient. Considering the CV applications that require data from multiple sources (e.g., OBUs, RSUs), a CV environment is considered to be one of the largest distributed networks in the near future (*11*). As the size of the network grows (e.g., number of vehicles, sensors, roadside infrastructure), the demand for data will also increase (*12*). Thus, a platform for CV application development needs to be designed in such a way so that it can handle a high demand of data without compromising the quality of service (in terms of temporal and spatial requirements). Thus, in providing the data to the users, CVDeP needs to meet the application requirement in terms of latency and throughput and must be capable of handling the scalability issues with the increasing number of connected vehicles, sensors, and roadside infrastructures.

Access Control and Application Security

Security is one of the major concerns in deploying CV applications because of the vulnerability and safety-critical aspect of connected transportation systems (*13*)(*14*). The USDOT proposed a security concept ‘security credential management system (SCMS)’ to ensure privacy and integrity in a CV system that includes application security. The data shared between applications and edge devices need to be secured and we need to maintain data confidentiality, integrity, and availability (*4*). One way to protect the data from unwanted user access is to authenticate user information before sharing and streaming data. In SCMS, fixed edges (e.g., RSUs) will provide a certificate to the application, which can be used by the application for message exchange (*15*)(*16*). Registration authority (RA) and certificate authority (CA) were considered for providing the certificate. While RA verifies the user request and checks the digital signature, CA issues a new digital certificate or renews a certificate. In our study, we have adopted a security module to control access, certificate exchange mechanism following SCMS, as well as application security based on policies developed by (*17*). In this study, we have considered the data and application security, however, the network security is not part of this study.

CONNECTED VEHICLE APPLICATION DEVELOPMENT PLATFORM (CVDeP)

The proposed approach is illustrated in **Figure 1**, which includes conceptual development and implementation, and evaluation and validation of CVDeP. In an edge-centric CPS, a CVDeP architecture is developed including application management platform and GUI for application development. Application management platform contains three modules: (i) control platform module; (ii) communication module; and (iii) data warehouse module. Application development GUI contains a graphical interface module. In the implementation phase of CVDeP, all four modules are developed and implemented. However, the control platform module includes three sub-modules: (a) access and credential management; (b) application security management; and (c) data collection and distribution. After that, we evaluate and validate the CVDeP using safety and mobility applications in two stages: i) evaluation in a CVDeP emulated environment and; ii) field validation. The safety application is evaluated using a communication and computation latency metric. On the other hand, the mobility application is evaluated using communication and computation latency, and throughput (to test the scalability of the platform). Later, we explain the experimental set-up, experiment scenarios and CV applications for the evaluation of CVDeP using each CV application. In the following sections, we present the above-mentioned study approach in detail for developing and evaluating the proposed CVDeP.

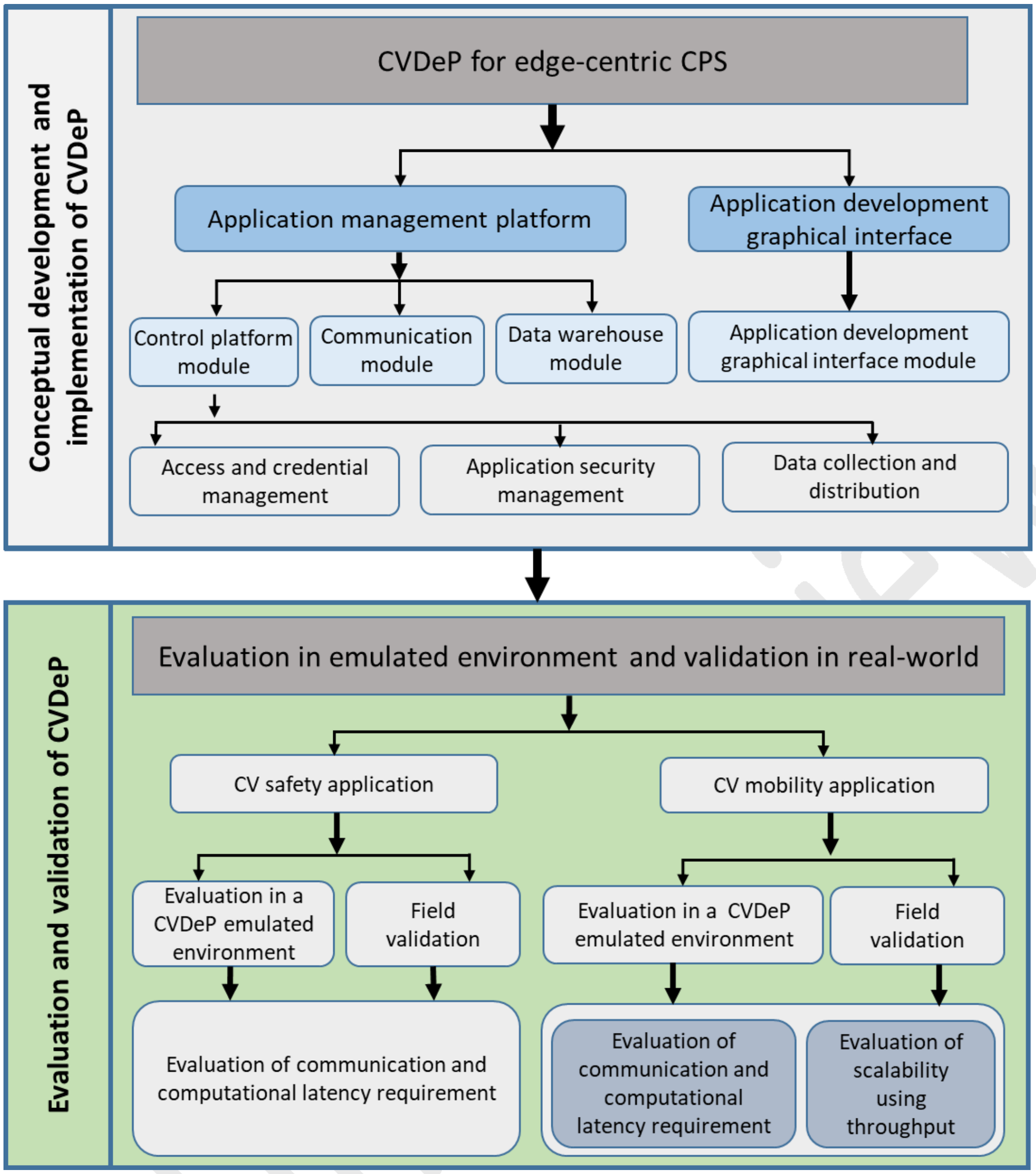


Figure 1 Study Approach for CVDeP Development and Evaluation

Conceptual Development and Implementation of CVDeP

In an edge-centric CPS, the physical proximity of devices to the data source is envisioned to reduce latency and the distributed architecture aims at increased scalability. The edge-centric CPS as shown in **Figure 2** for CV systems consist of three edge layers: i) mobile edge (e.g., on-board sensors); ii) fixed edge (e.g., roadside transportation infrastructure); and iii) system edge (e.g., backend server at TMC) (*3*). This hierarchical cyber-physical system architecture can address complexity and scale issues of CV systems. A system edge is a single endpoint for a cluster of fixed edges. A fixed edge includes a general-purpose processor (i.e., application development device) and a dedicated short-range communication (DSRC)-based RSU. A fixed edge can communicate with mobile edges using DSRC and communicate with the system edge using optical fiber/Wi-Fi. Although we are using DSRC, any low latency communication technology, such as 5G can be incorporated in our development platform. Fixed edge can be extended to support a video camera and other sensing devices, such as weather sensors and GPS sensors. CVs participating in our system will be acting as mobile edges and are equipped with a DSRC-based OBU. Fixed edges are connected to a system edge that can effectively serve as a backend resource. Mobile edges (edge layer 1) can exchange data with fixed edges (edge layer 2) and system edges (edge layer 3) using DSRC and LTE/Wi-Fi communication, respectively as shown in **Figure 2**.

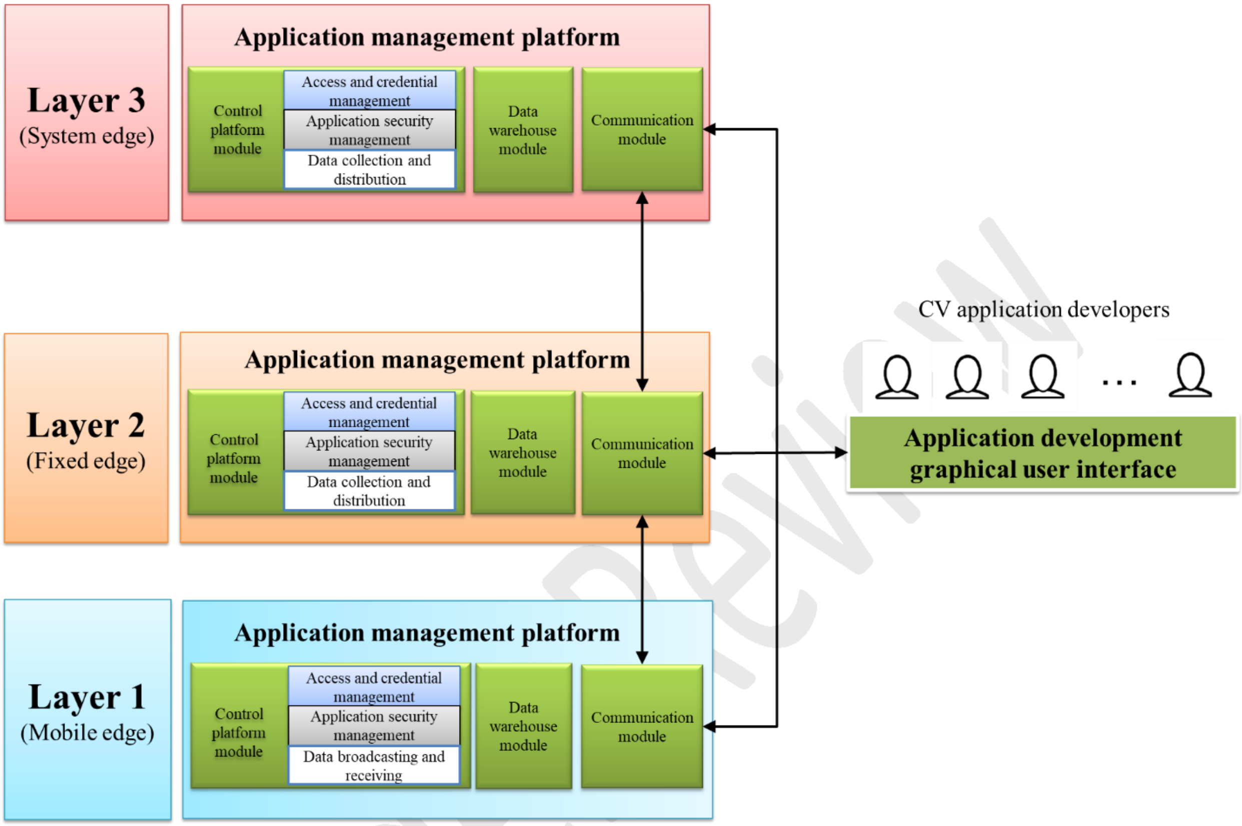


Figure 2 CVDeP architecture for an edge-centric CPS

In an edge-centric CPS for CVs, each component generates different types of data. For example, OBUs installed in a vehicle (i.e., mobile edge) broadcast BSMs, which contain the vehicles' information, such as location, speed, direction, acceleration, and braking status (*18*). The fixed edge (i.e., RSU with an additional edge device that has computational power) collects data from the OBUs and acts a primary gateway to transfer data from CVs to the transportation infrastructures (e.g., system edge, which could represent a TMC). For developing a CV application, developers need to interact with all of the layers mentioned above. Hence, edge layers are accessed through an application development GUI, which provides a way for the CV application developers to interact with the different edge layers. **Figure 2** illustrates the architecture of our CVDeP for an edge-centric CPS which comprises of the following two components: 1) application management platform, and 2) application development graphical user interface.

Application Management Platform

The application management platform is responsible for the selection of the appropriate communication medium of an application, and data collection, storage, and distribution, while ensuring the security of the platform by providing secured access control and security management of the CV applications. As presented in **Figure 2**, the application management platform resides in between the application development GUI and the underlying CV components (i.e., each edge) of the edge-centric CPS. Application developers interact with the management platform through an application development GUI. The application management platform is made up of the following modules: i) control platform module; ii) data warehouse module; and iii) communication module. Following subsections describe the conceptual development and implementation details of each of the module.

Conceptual development of control platform module

The control platform of fixed and system edges supports three types of operations: i) access control and credential management; ii) application security management; and iii) data collection and distribution. On the other hand, the control platform of mobile edge includes: i) access control and credential management; ii) application security management; and iii) data broadcasting to and receiving from the various mobile and fixed edge devices. Edge devices on an edge-centric CPS continuously exchange data between different edges. The data broadcasting and receiving module in the mobile edges handle the continuous data exchange between mobile edges and other edges (i.e., the system edge and the fixed edge). This module continuously provides BSMs to the application developers that can be used to develop CV applications. On the other hand, the data collection and distribution module in fixed edges and system edges is responsible to gather and distribute data to and from mobile edges, fixed edges, and system edge in real-time. Both the broadcasting-receiving module and collection-distribution module can be used by the developer to develop any type of CV applications. After the access control and credential management component are activated, authenticated application developers can access, gather and visualize real-time streaming data generated from different components of each edge layer. In addition, application security management module is responsible for monitoring the data flow and securing the application using security policies.

Implementation of control platform module

The control platform contains four modules depending on whether the edge device is a mobile, fixed or system edge. Implementation overviews of these modules are as follows:

- *Access control and credential management system.* The access control and credential management module ensures that only authorized users have access to CVDeP services. Developers are authenticated via a login interface before given access to the edge-centric CPS testbed components. Permission-Based access control is implemented by providing access rights to application-specific data and services (e.g., access to the BSMs, access sensors data, access to the data warehouse) like android application system where permission are written in a manifest file prior to developers developing the Android application (*19*). On the other hand, the credential management system (CMS) was implemented based on the public key infrastructure (PKI), which takes care of public key exchange that is needed for encrypting and authenticating data using a digital signature. CMS is built in such a way that the functionalities of SCMS proposed by USDOT has been replicated (*15*)(*20*). We have followed the assumptions by the National Highway Traffic Safety Administration (NHTSA) connected vehicle pilot program where V2V communications are secure, but not encrypted, and V2I communication is both secure and encrypted (*21*).
- *Application security management.* The flow-based control module as proposed in (*22*) is implemented within the data collection and distribution systems to ensure application security. Initially, all the consumers and producers need to be authenticated (action 1 (A1) and action 2

(A2) respectively in **Figure 3**) to produce and consume the message. Then they are allowed to produce (A3) and consume (A6) data from the data collection and distribution module. In the security module, trusted application programming interface (API) and quarantine module checks the flow policies (A4 and A5) and deliver the data (A6) to the appropriate consumers (e.g., the consumers who are authenticated and subscribed to a particular topic). As shown in **Figure 3**, producers and consumers communicate with the data collection and distribution module via a trusted API. This trusted API removes any sensitive information (e.g., drivers identify and vehicle ID). Moreover, this trusted API enforces the flow policies among the applications. Using these flow policies, application security can be ensured. In our study, we have implemented the flow policies using '<*source, sink*>' tracking as described in (*22*) in which source is the producer of the data and sink is the intended recipient of that data.

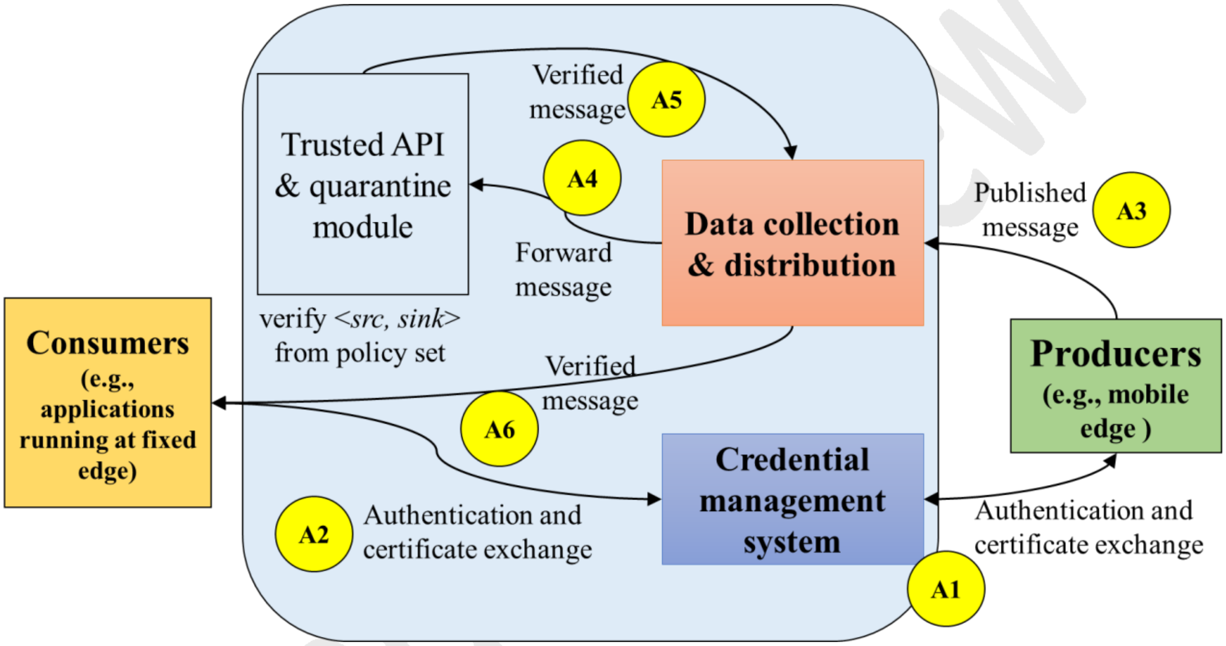


Figure 3 Implementation of application security module with data collection and distribution systems

- *Data collection and distribution.* Data collection and distribution system is the core part for fixed and system edges of CVDeP. We have selected Kafka as a broker-based system data collection and distribution systems because of the following efficacies (*23*): 1) high throughput; 2) low latency; 3) reliability of data delivery, and 4) scalability. In a publish-subscribe based broker-system, data producers (e.g., mobile edges, applications) produce and publish data to the broker, whereas the data consumers (e.g., fixed edge, applications) subscribe and consume the data available at the broker. By tagging individual data elements with labels/topics, producers can produce data for a particular topic and consumers can subscribe to data of that topic. Brokers receive data from producers and immediately make the data available for consumers to consume. As a result, producers and consumers can generate and consume data in an asynchronous and independent manner.
- *Data broadcasting and receiving.* The data broadcasting and receiving module is developed for mobile edge devices, where it is responsible for generating BSMs and receiving the BSMs from other mobile edges. In our implementation, each mobile edge is broadcasting BSMs at a default

rate of 10Hz and each BSM contains necessary attributes for safety applications (e.g., position, speed, and direction) of corresponding mobile edge (*18*)(*24*). In addition, each mobile edge is receiving BSMs from all other mobile edges within their communication range.

Conceptual development of data warehouse module

The data warehouse stores the data generated from different edge devices, sensors, and applications deployed in the edge layers. It is a distributed storage system which resides in the fixed edge and the system edge. The purpose of the data warehouse is to store and provide necessary information that is needed by the developers and/or edge layers for any application's needs. As a mobile edge is limited by computation power and storage size, we do not include a data warehouse in mobile edges. In fixed edges and system edges, the structure of the data warehouse is such that it can support and store both structured (e.g., GPS data) and unstructured data (e.g., text and images). A structured data has a strict tabular format whose column size and attributes of each entity are defined. Examples of structured data include any data that can be stored in delimited formats, spreadsheets, or SQL tables, whose columns are defined. A semi-structured data includes data whose fields are defined but organized in a hierarchical manner. Examples include data stored in extensible markup language (XML) or JavaScript object notation (JSON) formats. Unstructured data, such as pictures, videos, and textual data, do not have any structural organization associated with the data itself.

Implementation of data warehouse module

In our implementation, to support structured, semi-structured, as well as unstructured data, we have used MySQL for structured data and NoSQL for semi-structured and unstructured data. With the structured, semi-structured and unstructured data together produces a huge amount of data in terms of volume. Realistically, we do not need to store all the raw data in their original format. As a result, a lambda infrastructure (e.g., Amazon web service), which is designed to handle data in massive quantities using batch processing, can help to reduce and compress historical data for subsequent batch processes.

Conceptual development of communication module

Communication module decides the best available communication medium suitable for use for the particular application. Developers will provide temporal and spatial requirements of an application to the communication module through application development GUI, and then communication module creates an abstraction layer for the developers on top of the internal communication networks. For example, communication module selects DSRC, which is a low latency communication medium, from the available communication options to satisfy the requirement of safety applications. While the application is running in edge devices, CVDeP will provide communication metadata (e.g., available communication mediums such as DSRC, LTE, and Wi-Fi, and their average, maximum, and minimum transmission latency) for evaluating the performance of the application.

Implementation of communication module

In our communication module implementation, the discovery or searching of communication mediums and their network statistics are measured in the background asynchronously. An application is agnostic of the communication mediums and the decision of the medium to use for transmitting and receiving data is decided by the communication module while control platform functionalities are involved throughout the process as described in the previous sections. We have added the metadata support layer in the communication module to provide metadata to the developers that can support them to develop their applications. Through this metadata layer, developers will be able to observe the communication attributes, such as signal strength, bandwidth utilization, and data loss. A script running in CVDeP provides this information to the developers, and developers can evaluate the effect of communication medium on the performance of an application.

Application Development Graphical User Interface

Application developers can access the underlying edge devices of the edge-centric CPS using a GUI and can develop and deploy any CV application directly on the edge-centric CPS. Based on the requirements of a CV application, interface access rights and available services (e.g., communication services, data storage service) of the platform, application developers can access to the different types of data (e.g., real-time and historical) through an application development GUI in each layer. Using this application development GUI for each layer, application developers can also request any specific data for a specific application purpose. For example, developers can request data from the data warehouse to predict the future roadway traffic condition. Application development GUI will provide an interactive platform to the developers to build their own applications and test these applications by requesting both real-time data from CVs and other sensors, and historical data from the data warehouse from fixed and system edges.

The application development GUI is developed as a desktop application in C# (C sharp) as illustrated in **Figure 4**. Currently, the software has only been developed for the Windows operating systems as a proof-of-concept.

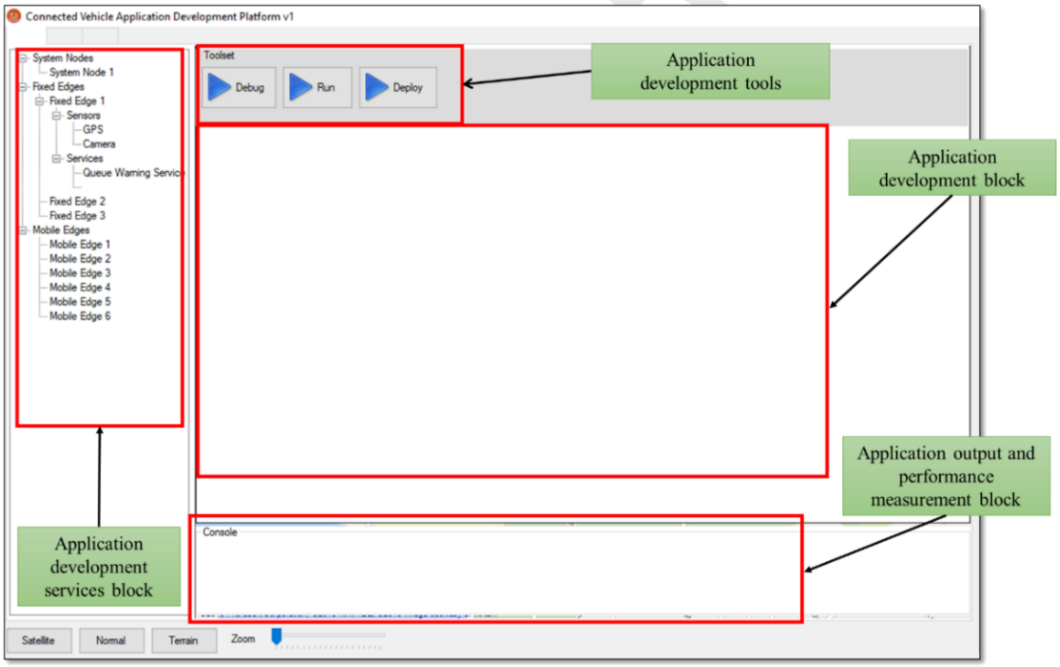


Figure 4 Implementation of application development graphical interface

EXPERIMENTAL SETUP

This section provides a description of the experimental set-up in an emulated environment and real-world environment for a safety and a mobility application to evaluate the efficacy of CVDeP.

Experimental Setup in Emulated Environment

A developer can develop and evaluate the performance of the developed CV applications in the emulated environment. In this environment, the developer will have dedicated hardware to emulate the real-world edge-centric CPS for CVs. As shown in **Figure 5**, a developer can emulate mobile edges using hardware setup #1 and #2 and fixed edges using hardware setup #3, where system edges have been set-up in a dedicated server in Clemson University. Each hardware setup (#1, #2, and #3) consists of one DSRC unit to send and receive the DSRC messages, and computing device for computation as well as communication purpose. Hardware setup #1 is used for developing the safety application whereas hardware setup #2 is used for emulating other mobile edges for safety application. For mobility and environmental application, only hardware setup #2 can be used for emulating mobile edges. Hardware setup #3 is used for creating any number of fixed edges where the location of fixed edges are defined by developers through CVDeP interface. A dedicated server located in Clemson University is intended for creating system edge instances. In this emulated edge-centric CPS, mobile edges and fixed edges communicate with each other using DSRC, and fixed edge and system edge communicate using the Clemson University communication network, which includes optical fiber and Wi-Fi connections. In addition, developers can configure the number of edges in each layer as required by the application. To generate the movement data of mobile edges, the movement of the mobile edges are exported from the 'Simulation of urban mobility (SUMO) (*26*)', which is a microscopic traffic simulator software, using a SUMO trace file. Using this SUMO trace file, developers can create any roadway environment, and generate any number of emulated vehicles and their corresponding BSMs. A program running in mobile edges read that trace file and generate BSMs for each vehicle. Then, these BSMs are broadcasted using DSRC for each vehicle. Fixed edges will receive BSMs only within its communication range, which is defined by the developers. Developers can access the edges through CVDeP Interface in order to develop and evaluate the performance of the developed application.

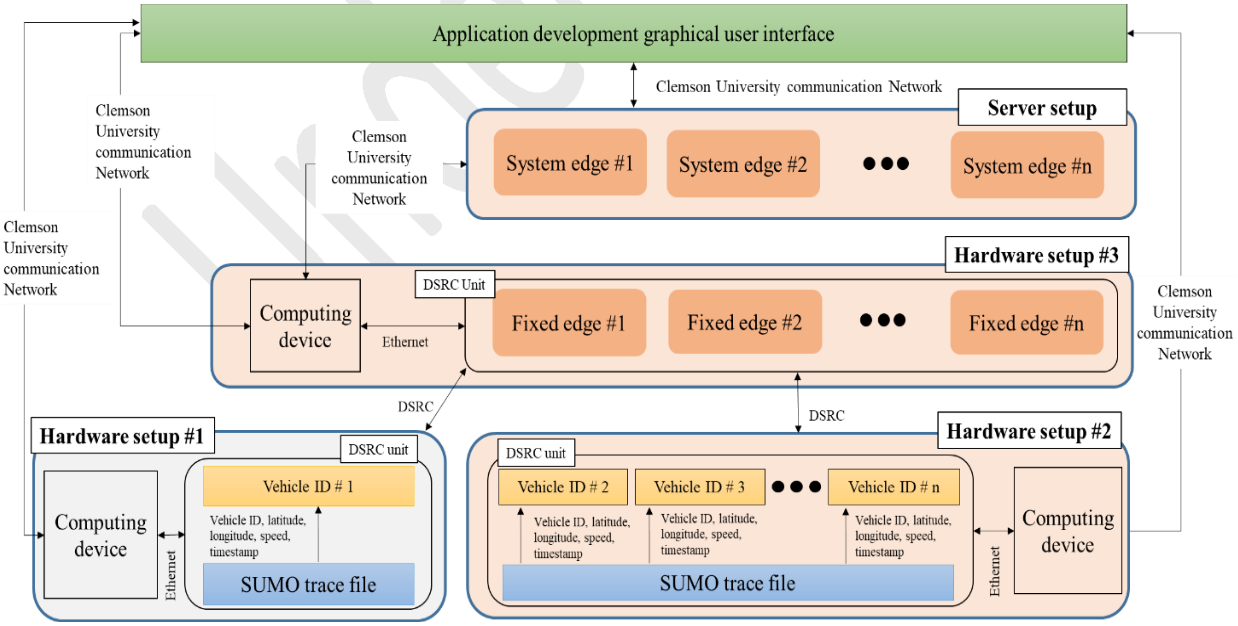


Figure 5 CVDeP setup in an emulated edge-centric CPS

Experimental Setup in CU-CVT

The CU-CVT has three fixed edges, which are deployed along the Perimeter Road in Clemson, South Carolina, and one system edge is deployed as the backend server (*25*). The backend server is located at Clemson University and connected to the Clemson University network. Two of the fixed edges are connected to the Clemson University Network via optical fiber link and one fixed edge is connected to Clemson University network using Wi-Fi link. Each fixed edge has its own DSRC radio to communicate with mobile edges. Each mobile edge (primarily OBUs on vehicles) is equipped with wireless communication devices such as DSRC, LTE and Wi-Fi. In addition, the communication module is available in the CU-CVT for mobile and fixed edges.

EVALUATION AND VALIDATION OF CVDeP

For our case study, we have developed 'Forward collision warning (FCW)' as a safety application and 'Traffic data collection for traffic operations' as a mobility application (*4*) using CVDeP. Then, to prove the efficacy of CVDeP, FCW and Traffic data applications are evaluated in an emulated environment and real-world CU-CVT (*25*).

Safety Application

We developed a FCW application based on the study by (*27*), where FCW application uses a vehicle kinematics (VK) model for generating collision warnings using DSRC communication. Based on the VK model, FCW application generates rear-end collision warnings when two vehicles are closer than a defined safe distance. **Equation (1)** is a modified version of the FCW application used in our study:

$$D_{w=} \frac{(V_o - V_t)^2}{2 * a_{moderate}} + d \qquad (1)$$

Where D_w is the distance threshold for collision warning is; V_o is the preceding vehicle's speed; and V_t is the follower/target vehicle's speed. The follower/target vehicle is the vehicle where the FCW application is intended to run in reality; d is the average length of the preceding and following vehicles, and $a_{moderate}$ is set to 11 ft/s^2 following the SUMO configuration.

Evaluation Scenarios

We create two evaluation scenarios for evaluating the CVDeP as a safety application development platform: i) scenario 1: the preceding vehicles (hardware setup #2 in **Figure 5**), and follower or target vehicle (hardware setup #1 in **Figure 5**) is moving in the same lane with 20 mph and 30 mph, respectively; ii) scenario 2: both front and follower vehicle are moving with 30 mph and the front vehicle stops suddenly. In both scenarios, FCW application is deployed in the follower vehicle, and forward-collision warnings are generated based on the comparison between calculated safety distance (using **Equation 1**) and the distance between two vehicles using real-time GPS data. To evaluate the performance of the application we have considered data delivery latency as a measure of effectiveness. In this context, latency is the time when data was generated by a mobile edge to the time when the application produced FCW message in the follower vehicle. Here, latency includes both network latency and computational latency.

Evaluation in CVDeP Emulated Environment

We evaluate the FCW application, using the experimental setup as described in the previous sections. The application is developed using CVDeP application development GUI, and then the application is tested using the two evaluation scenarios. **Table 1** provides a summary of latency recorded from both evaluation scenarios using CVDeP. For the evaluation of FCW application in CVDeP, we have taken the data of 200s containing 4000 BSMs from two mobile edges to calculate the maximum, minimum, and average latency. The average latency is 16 ms for both evaluation scenarios 1 and 2, respectively. However, the recorded maximum latencies were 95 milliseconds (ms) and 77 ms, which is below the safety application latency requirement (i.e., 200ms (*28*)). In **Table 1**, we present the network latency only. The computational latency for running the application is 1.5 ms, which is same for both evaluation scenarios. In addition, these FCW messages are sent to the fixed edge using the best available communication medium decided by communication module which takes 0.5 ms to decide, given that all communication mediums (LTE, Wi-Fi, and DSRC) are running simultaneously, and communication module is monitoring these mediums asynchronously.

Field Validation in CU-CVT

For our field evaluation of FCW in CU-CVT, we followed a similar speed profile for both evaluation scenarios provided in **Table 1** and measured the communication latency for the FCW application. **Table 1** provides the summary of latency recorded for both evaluation scenarios in the field experiment. Similar to the evaluation in an emulated environment, we have taken the data sample of 200s containing 4000 BSMs from two mobile edges to calculate the maximum, minimum, and average latency. The average latency measured is 63 ms and 49 ms for scenarios 1 and 2, respectively. The maximum latency recorded for the test is 113 ms and 105 ms, which is below the safety application latency requirements (i.e., 200ms (*28*)). In our field experiment, we have observed lower latency than the latency measured in emulated experimental setup because of no environmental effect or propagation loss. In **Table 1**, we only present the network latency, and we do not present the computational latency for an application which was 2 ms. In both cases (scenario 1 and 2), we can validate that the application developed in the emulated setup was able to fulfill the application latency requirement (200ms) in the field experiment. Same as before, the communication module takes about 0.5 ms time on average to decide the communication medium to use to send the FCW messages to upper edge layers.

TABLE 1 Summary of Latency for FCW Application Evaluation

Experimental Setup	**Evaluation Latency Parameter**	**Latency in Scenario #1**	**Latency in Scenario #2**	**Latency requirements for Safety Application** (*28*)(*29*)
Emulated environment	Maximum	95 ms	77 ms	≤ 200 ms
	Average	16 ms	16 ms	
	Minimum	2 ms	2 ms	
CU-CVT	Maximum	113 ms	105 ms	
	Average	63 ms	49 ms	
	Minimum	2 ms	3 ms	

Mobility Application

We evaluate our CVDeP using ‘Traffic data collection for traffic operations’ application. This application collects CVs’ data (e.g., BSMs) to support traffic operations, such as incident detection and localized traffic operational strategies (*4*). According to this application, it is required to divide the application into two sub-applications: i) sub-application 1: collect real-time traffic data from mobile edges; and ii) sub-application 2: collect real-time traffic data from fixed edges. Sub-application 1 runs in each fixed edge and sub-application 2 runs in the system edge.

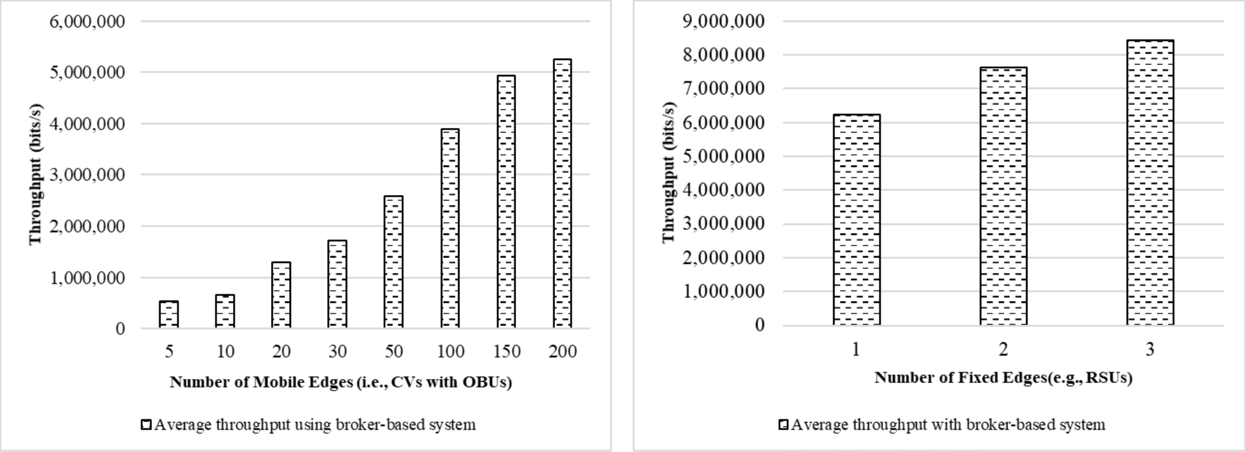
We evaluate the scalability of our designed CVDeP to ensure the CV application requirements are met in terms of latency and throughput. The latency is the time difference between the time of data generation at the edge-centric CU-CVT and the time when the data is received by the user. Data delivery latency requirement for any mobility and environmental applications must be satisfied in order to provide mobility and environmental services. As CVDeP aims to support different mobility and environmental applications, we have considered 1000 ms as the latency threshold to deliver the CV data to the developer (*8*). Also, we need to ensure a high throughput (i.e., the data transfer rate) means the high use of the allocated bandwidth. Our platform already fulfilled the spatial requirement of the application, as mobile edges will be within the communication range of fixed edges.

Evaluation Scenarios

We create two different scenarios for evaluating our application development platform by varying the number of fixed edges (RSUs) and the number of mobile edges: i) scenario 1: one system edge and one fixed edge with varying number of mobile edges (5, 10, 20, 30, 50, 100, 150, and 200); ii) scenario 2: one system edge, varying number of fixed edges (RSUs) (1, 2, and 3) and 200 mobile edges (CVs) for each fixed edge. For evaluation scenario 2, based on fixed edge's coverage, the number of CVs on Perimeter road approaching to the intersection stop line is 200 (maximum number of CVs for four-lane (two lanes in each direction) road during a congested condition according to our traffic volume count). For each scenario, we have evaluated the scalability of the application development platform in terms of data delivery latency and throughput.

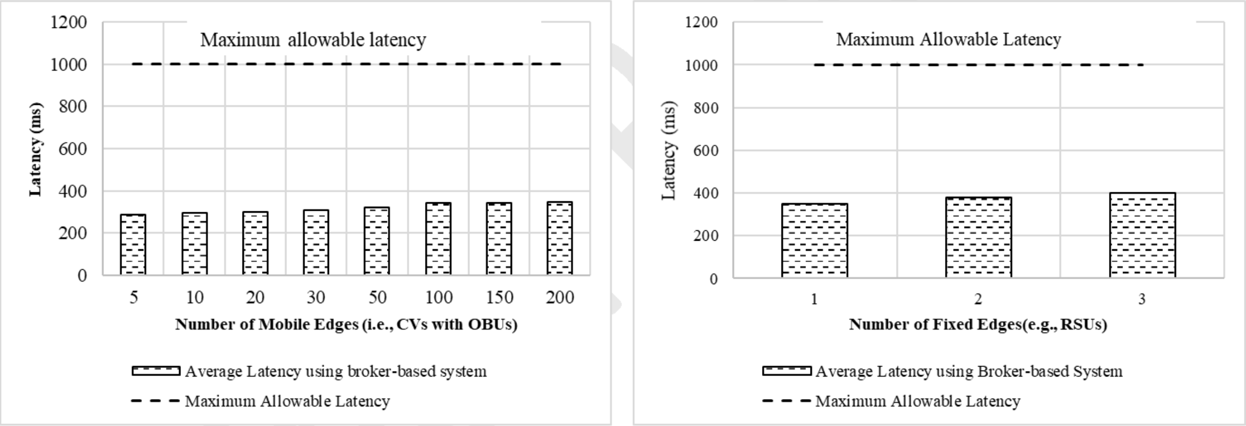
Evaluation in CVDeP Emulated Environment

We implement a data collection and distribution systems (the broker-based system) that is required for the real-time application development platform. We evaluate the scalability of the CVDeP considering access and credential management and application security modules with different data collection and distribution systems. Then we compare with latency requirement for the selected CV application. As shown in **Figure 6(a)** and **6(b)**, with the increasing number of mobile edge and fixed edge, the throughput of the broker-based system is linearly increasing and reaches a maximum at 5.2 Mbits/s and 8.4 Mbits/sec, respectively. Higher throughput ensures reliable and scalable services. The broker-based system (e.g., Kafka) uses an asynchronous mode that can collect and distribute data in memory and send them in batches in a single shot (*30*). Because of this asynchronous mode and sending data in batch, the broker-based system can ensure high throughput. In the broker-based system, the system adapts the application development platform's throughput as the number of mobile edges and fixed edge increases and thus can handle more data. We observe that CVDeP data collection and distribution system can maintain a lower latency with the increasing number of mobile edges (**Figure 6(c)**) and fixed edges (**Figure 6(d)**). The increment of latency with the broker-based method is negligible for both use case scenarios (scenarios 1 and 2). The reason is that the broker-based system uses an intelligent 'sendfile' method with zero-copy optimization (i.e., sending the data directly to the consumer without any buffering or copying to memory) (*30*). Thus, the broker-based system can maintain a lower message delivery latency irrespective of the number of producers and consumers thus ensuring scalability. In our experiment, we have used the default configuration of a Kafka broker-based system (e.g., replication factor =1, topic partition =1, and single broker). However, the configuration (e.g., topic partitions, replication, multiple Brokers) of Kafka broker-based system can be configured easily to reduce the latency if the latency is higher than the CV application threshold. In addition, by adding additional data management brokers, as presented by (*6*), CVDeP can be scaled up to receive and share data from additional connected data sources (e.g., personal handheld devices, news media and weather stations, traffic operators).



6(a) Throughput for different number for mobile edges

6(b) Throughput for different number of fixed edges



6(c) Latency for different number of mobile edges

6(d) Latency for different number of fixed edges

Figure 6 Evaluation of CVDeP for mobility application using application throughput and latency

Field Validation in CU-CVT

We have evaluated the CVDeP in CU-CVT using five mobile edges (e.g., CVs) in the field experiment. **Table 2** shows the summary of latency when we developed the application in the CVDeP emulated environment and CU-CVT. We observed higher latency (maximum, average and minimum) in the field than in the CVDeP. In the field experiment, the data exchange through DSRC between the mobile edge and fixed edge in the field was affected by the environmental inferences, such as trees, roadway slope, and curvature. This causes a higher variation in latency in the field than in the CVDeP. However, latency observed in the field was still far below the latency requirements for mobility applications.

Table 2 Summary of latency for mobility application with five CVs

Evaluation Latency Parameter	Latency		Latency requirements for Mobility Application (*8*)
	Evaluation in Emulated Environment	Validation in CU-CVT	
Maximum	115 ms	267 ms	≤1000 ms
Average	65 ms	69 ms	
Minimum	4 ms	6 ms	

CONCLUSIONS AND FUTURE WORK

CV technology holds the promise of improving traffic safety and efficiency of traffic operations. For CV benefits to materialize, the active participation of CV researchers and developers is necessary. This can be hindered due to the lack of real-world application development platforms that uses real-world and real-time data to support the CV application development process including testing and debugging. Our research and development contribute directly by developing a CV application development platform, CVDeP, for an edge-centric CPS. Using this CVDeP, CV application developers can interact with a real-world edge device, and develop, test and debug CV safety and mobility applications using real-time data. From our case study, it is revealed that the applications developed using CVDeP are able to fulfill the CV safety and mobility application latency requirements and provide high throughput both for an increasing number of mobile edges, and multiple fixed edges. We showed that forward collision warning application (a safety application) developed using CVDeP can fulfill the latency requirement (200 milliseconds) of safety applications. Also, traffic data collection for traffic operations application (a mobility application) developed using CVDeP with the broker-based system shows about 400 milliseconds of latency with three fixed edges and 600 mobile edges, which is much lower than the latency requirement (1000 milliseconds) of mobility applications. This also proves the scalability of our CVDeP while fulfilling the latency requirement of CV applications for an edge-centric CPS. We are also in the process of publishing architecture and code of the CVDeP via Github platform.

There exist few limitations such as the resiliency and fault tolerance of the platform have not been evaluated. This research is conducted using multiple mobile edges (CVs) and fixed edges (RSUs), and the evaluation is conducted with two CV applications only. In addition, only one system edge is used for our evaluation and only data from mobile and fixed edges are collected to evaluate CVDeP and not the data from other sensors or roadside infrastructure (e.g. Traffic signal controllers). As CVDeP is being developed and refined further, future studies shall include: i) incorporation of data from other traditional data sources (e.g., traffic signal, loop detector) and non-traditional data sources (e.g., news media, weather sensors, social networking sites); ii) evaluation of the fault tolerance and resiliency of the platform; iii) evaluation of multiple applications running simultaneously in multiple system edges while merging information from diverse data sources from a large network (i.e., data residing at local or city/county level, regional or state level, and/or national level); and iv) strategy identification to make the system more secure by incorporating different security threat detection and protection mechanisms against different malicious activity including cyber-attacks.

ACKNOWLEDGMENTS

This study is based upon work supported by the USDOT Connected Multimodal Mobility University Transportation Center (C^2M^2) (Tier 1 University Transportation Center) headquartered at Clemson University, Clemson, South Carolina, USA. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the USDOT Center for Connected Multimodal Mobility (C^2M^2), and the U.S. Government assumes no liability for the contents or use thereof. Also, the authors would like to thank Aniqa Chowdhury for editing the paper.

AUTHOR CONTRIBUTIONS

The authors confirm contribution to the paper as follows: study conception and design, Mhafuzul Islam Mizanur Rahman, Sakib Khan, Mashrur Chowdhury and Lipika Deka; data collection, Mhafuzul Islam and Mizanur Rahman; interpretation of results, Mhafuzul Islam Mizanur Rahman, Sakib Khan, Mashrur Chowdhury and Lipika Deka; draft manuscript preparation, Mhafuzul Islam Mizanur Rahman, Sakib Khan, Mashrur Chowdhury and Lipika Deka. All authors reviewed the results and approved the final version of the manuscript.

REFERENCES

1. Lopez, P. G., A. Montresor, D. Epema, A. Iamnitchi, P. Felber, and E. Riviere. Edge-Centric Computing : Vision and Challenges. Vol. 45, No. 5, 2015, pp. 37–42.
2. Grewe, D., M. Wagner, M. Arumaithurai, I. Psaras, and D. Kutscher. Information-Centric Mobile Edge Computing for Connected Vehicle Environments : Challenges and Research Directions. 2017.
3. Rayamajhi, A., M. Rahman, M. Kaur, J. Liu, M. Chowdhury, D. Ph, H. Hu, and D. Ph. ThinGs In a Fog : System Illustration with Connected Vehicles. 2017. https://doi.org/10.1109/VTCSpring.2017.8108558.
4. ARC-IT. Architecture Reference for Cooperative and Intelligent Transportation. https://local.iteris.com/arc-it/index.html. Accessed Jul. 10, 2019.
5. Intelligent Transportation Systems - Real-Time Data Capture and Management. https://www.its.dot.gov/factsheets/realtime_dcm_factsheet.htm. Accessed Jul. 24, 2017.
6. Du, Y., M. Chowdhury, M. Rahman, K. Dey, A. Apon, A. Luckow, and L. B. Ngo. A Distributed Message Delivery Infrastructure for Connected Vehicle Technology Applications. Vol. 19, No. 3, 2018, pp. 787–801.
7. Karagiannis, G., O. Altintas, E. Ekici, G. Heijenk, B. Jarupan, K. Lin, and T. Weil. Vehicular Networking: A Survey and Tutorial on Requirements, Architectures, Challenges, Standards and Solutions. *IEEE Communications Surveys and Tutorials*, Vol. 13, No. 4, 2011, pp. 584–616. https://doi.org/10.1109/SURV.2011.061411.00019.
8. U.S. Department of Transportation (US DOT). *SOUTHEAST MICHIGAN TEST BED 2014 CONCEPT OF OPERATIONS*. 2010.
9. Agmon, N., and N. Ahituv. Assessing Data Reliability in an Information System. *Journal of Management Information Systems*, Vol. 4, No. 2, 1987, pp. 34–44. https://doi.org/10.1080/07421222.1987.11517792.
10. Balke, K., H. Charara, and S. Sunkari. Report on Dynamic Speed Harmonization and Queue Warning Algorithm Design. 2014, pp. 1–79.
11. Qian, Y., and N. Moayeri. Design of Secure and Application-Oriented Vanets. 2008.
12. Baker, E. H., D. Crusius, M. Fischer, W. Gerling, K. Gnanaserakan, H. Kerstan, F. Kuhnert, J. Kusber, J. Mohs, M. Schule, J. Seyfferth, J. Stephan, and T. Warnke. *Connected Car Report 2016: Opportunities, Risk, and Turmoil on the Road to Autonomous Vehicles*. 2016.

13. Zarki, M. El, S. Mehrotra, G. Tsudik, and N. Venkatasubramanian. Security Issues in a Future Vehicular Network. *Symposium A Quarterly Journal In Modern Foreign Literatures*, Vol. 35, 2002, pp. 270–274. https://doi.org/10.1073/pnas.0914054107.

14. Raw, R. S., M. Kumar, and N. Singh. Security Challenges, Issues and Their Solutions for VANET. *International Journal of Network Security & Its Applications*, Vol. 5, No. 5, 2013, pp. 95–105. https://doi.org/10.5121/ijnsa.2013.5508.

15. Whyte, W., A. Weimerskirch, V. Kumar, and T. Hehn. A Security Credential Management System for V2V Communications. *IEEE Vehicular Networking Conference, VNC*, No. December, 2013, pp. 1–8. https://doi.org/10.1109/VNC.2013.6737583.

16. Ahmed-Zaid, F., Bai, F., Bai, S., Basnayake, C., Bellur, B., Brovold, S., Brown, G., Caminiti, L., Cunningham, D., Elzein, H., Hong, K., Ivan, J., Jiang, D., Kenney, J., Krishnan, H., Lovell, J., Maile, M., Masselink, D., McGlohon, E., Mudalige, P., Popov, S., F. Ahmed-Zaid, F. Bai, S. Bai, C. Basnayake, and S. Ahmed-Zaid, F., Bai, F., Bai, S., Basnayake, C., Bellur, B., Brovold, S., Brown, G., Caminiti, L., Cunningham, D., Elzein, H., Hong, K., Ivan, J., Jiang, D., Kenney, J., Krishnan, H., Lovell, J., Maile, M., Masselink, D., McGlohon, E., Mudalige, P., Popov. Vehicle Safety Communications–Applications (VSC-A) Final Report: Appendix Volume 3 Security. Vol. 3, No. September, 2011.

17. Islam, M., M. Chowdhury, H. Li, and H. Hu. Cybersecurity Attacks in Vehicle-to-Infrastructure Applications and Their Prevention. *Transportation Research Record*, Vol. 2672, No. 19, 2018, pp. 66–78. https://doi.org/10.1177/0361198118799012.

18. Kenney, J. B. Dedicated Short-Range Communications (DSRC) Standards in the United States. *Proceedings of the IEEE*, Vol. 99, No. 7, 2011, pp. 1162–1182. https://doi.org/10.1109/JPROC.2011.2132790.

19. Felt, A. P., E. Ha, S. Egelman, A. Haney, E. Chin, and D. Wagner. Android Permissions. *Proceedings of the Eighth Symposium on Usable Privacy and Security - SOUPS ’12*, 2012, p. 1. https://doi.org/10.1145/2335356.2335360.

20. Brecht, B., D. Therriault, A. Weimerskirch, W. Whyte, V. Kumar, T. Hehn, and R. Goudy. A Security Credential Management System for V2X Communications. *IEEE Transactions on Intelligent Transportation Systems*, Vol. 19, No. 12, 2018, pp. 3850–3871. https://doi.org/10.1109/TITS.2018.2797529.

21. Weil, T. VPKI Hits the Highway: Secure Communication for the Connected Vehicle Program. *IT Professional*, Vol. 19, No. 1, 2017, pp. 59–63. https://doi.org/10.1109/MITP.2017.5.

22. Fernandes, E., J. Paupore, A. Rahmati, D. Simionato, M. Conti, and A. Prakash. FlowFence: Practical Data Protection for Emerging IoT Application Frameworks. *Usenix Security*, 2016, pp. 531–548.

23. Kreps, J., N. Narkhede, and J. Rao. Kafka: A Distributed Messaging System for Log Processing. *ACM SIGMOD Workshop on Networking Meets Databases*, 2011, p. 6.

24. Park, Y., and H. Kim. Application-Level Frequency Control of Periodic Safety Messages in the IEEE WAVE. *IEEE Transactions on Vehicular Technology*, Vol. 61, No. 4, 2012, pp. 1854–1862. https://doi.org/10.1109/TVT.2012.2190119.

25. Chowdhury, M., M. Rahman, A. Rayamajhi, S. M. Khan, M. Islam, Z. Khan, and J. Martin. Lessons Learned from the Real-World Deployment of a Connected Vehicle Testbed. *Transportation Research Record*, Vol. 2672, No. 22, 2018, pp. 10–23. https://doi.org/10.1177/0361198118799034.

26. DLR - Institute of Transportation Systems - SUMO – Simulation of Urban MObility. http://www.dlr.de/ts/en/desktopdefault.aspx/tabid-9883/16931_read-41000/. Accessed Jun. 5, 2017.
27. Xiang, X., W. Qin, and B. Xiang. Research on a DSRC-Based Rear-End Collision Warning Model. *IEEE Transactions on Intelligent Transportation Systems*, Vol. 15, No. 3, 2014, pp. 1054–1065. https://doi.org/10.1109/TITS.2013.2293771.
28. Dey, K., A. Rayamajhi, M. Chowdhury, P. Bhavsar, and J. Martin. Vehicle-to-Vehicle (V2V) and Vehicle-to-Infrastructure (V2I) Communication in a Heterogeneous Wireless Network - Performance Evaluation. *Transportation Research Part C: Emerging Technologies*, Vol. 68, 2016, pp. 168–184. https://doi.org/10.1016/j.trc.2016.03.008.
29. Ahmed-Zaid, F., F. Bai, S. Bai, C. Basnayake, B. Bellur, S. Brovold, G. Brown, L. Caminiti, D. Cunningham, H. Elzein, K. Hong, J. Ivan, D. Jiang, J. Kenney, H. Krishnan, J. Lovell, M. Maile, D. Masselink, E. McGlohon, P. Mudalige, Z. Popovic, V. Rai, J. Stinnett, L. Tellis, K. Tirey, and S. VanSickle. VSC-A Final Report. *National Highway Traffic Safety Administration*, No. September, 2011, pp. 1–102.
30. Apache Kafka. https://kafka.apache.org/. Accessed Jul. 23, 2017.