TRR
JOURNAL OF THE TRANSPORTATION RESEARCH BOARD

Research Article

Transportation Research Record
2020, Vol. 2674(5) 537–552

DOI: 10.1177/0361198120917146
journals.sagepub.com/home/trr
SAGE

Development and Performance Evaluation of a Connected Vehicle Application Development Platform

Mhafuzul Islam[1], Mizanur Rahman[1], Sakib Mahmud Khan[2], Mashrur Chowdhury[1], and Lipika Deka[3]

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 system (CPS). The objective of this paper 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 while meeting the functional requirements of any CV applications. The efficacy of the CVDeP was evaluated using two types of CV applications (one safety and one mobility application) and they were validated through field experiments at the South Carolina Connected Vehicle Testbed (SC-CVT). The analyses show that the CVDeP satisfies the functional requirements in relation to latency and throughput of the selected CV applications while maintaining the scalability and security of the platform and applications.

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 phones), sensors (e.g., camera sensors), and traffic management centers (TMCs) (*1*). With integrated computing and control capabilities, these connected physical components communicate with each other to form a cyber-physical system (CPS). The architecture reference for cooperative and intelligent transportation (ARC-IT), which has been developed with the sponsorship of the US Department of Transportation (USDOT), has listed the functional requirements and provided the implementation guidelines of over a hundred CV applications for safety, mobility, and environmental benefits (*2*). For example, Vehicle Data for Traffic Operations is a CV application, which uses CV data obtained from vehicle OBUs to support roadway traffic operations (*2*). To develop such CV applications for 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, 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 and environmental sensor) (*3*). Although a user can stream CV data through the ODE platform in real-time for developing a CV application, it does not provide a platform to the application developers to build, test, and debug CV applications. Thus, it is critical to develop an application development platform and evaluate the platform in relation to latency and throughput to satisfy the temporal and spatial requirements of CV applications (*4*).

Considering a large-scale deployment of CV CPS, the concept of edge computing is introduced as the underlying computing approach (*5*). Edge computing has the potential benefits of enabling reduced communication latency and increased scalability. Such benefits are a result of bringing resources, such as storage, and computational resources, closer to the edge (*6, 7*). In an edge-

[1]Center for Connected Multimodal Mobility (C^2M^2), Glenn Department of Civil Engineering, Clemson University, Clemson, SC
[2]California Partners for Advanced Transportation Technology, Institute of Transportation Studies, University of California Berkeley, Berkeley, CA
[3]School of Computer Science and Informatics, De Montfort University, Leicester, UK

Corresponding Author:
Mhafuzul Islam, mdmhafi@clemson.edu

centric CPS, the resources for communication, computation, control, and storage are placed at different edge layers (e.g., a mobile edge as a vehicle, a fixed edge as a roadside infrastructure, and a system edge as a backend server or TMC) in a CV environment (*5*). Therefore, a CV application can be divided into sub-applications where sub-applications run in different edge layers depending on the requirements of the application.

Major challenges for developing a CV application development platform for an edge-centric CPS are to: (a) collect, process, and distribute data while running multiple CV applications concurrently in real-time in different edge layers; and (b) provide the scalability and security of the platform and applications. 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 the authors call Connected Vehicle Application Development Platform (CVDeP), has been designed to hide the underlying low-level software, hardware, and associated details. An application development graphical user interface provides the application developers an easy and secure access to the edge devices. The access control and credential management module in the application development platform prevents unwanted access to the edge devices and provides platform security. In addition, the application security module prevents malicious operations or activities propagated through an application in an edge-centric CPS. In this study, a policy-based security system is utilized to provide application security against cyberattacks. However, developing methods for detecting different types of cyberattacks and identifying related countermeasures are not the focus of this study.

Experiments were conducted to evaluate the efficacy of the CVDeP using a safety application (i.e., forward collision warning [FCW]) and a mobility application (i.e., vehicle data for traffic operations) (*2*). These applications were developed and evaluated in an emulated environment and later validated in a real-world edge-centric South Carolina Connected Vehicle Testbed (SC-CVT), which is located at Clemson, South Carolina. The FCW application was selected for the experiment, as it is a fundamental application for vehicle-to-vehicle (V2V) safety (*8*). Similarly, the vehicle data for traffic operations application was selected, 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 roadway traffic operational strategies (e.g., altering signal timing based on traffic flows, freeway speed harmonization, and optimization of ramp metering rates) (*2*). The efficacy of the CVDeP was presented using two communication-related measures of effectiveness, which are latency and throughput.

Contribution of the Study

The primary contribution of this study is the development of an architecture for an edge-centric CVDeP. This study systemically developed the architecture of the CVDeP, and evaluated and validated the CVDeP through experiments. In the Conceptual Development and Implementation of CVDeP subsection of the Connected Vehicle Application Development Platform (CVDeP) section of this paper, the architecture of the application development platform is presented, and each module of this architecture is defined. The architecture of the CVDeP supports modular development so that any user can easily include additional modules (e.g., adding an energy optimization module at a mobile and fixed edge levels for an eco-driving application) into the development platform if and when needed. Furthermore, the authors published the source code of the CVDeP in the GitHub, an open-source code management platform, so that any external users can use it and contribute to expanding the utility of CVDeP by adding more modules (*8*). The CVDeP open-source software will be maintained through a git version-control system.

Related Work

To develop the CVDeP that uses real-time CV data, the authors reviewed existing work related to the CV application development requirements, and developer access control and application security.

CV Application Development Requirements

CV applications are bounded by temporal and spatial requirements for providing the desired services (*9*). If CV data are not received within the temporal and spatial threshold as required by a CV application, CV data will not have any efficacy for real-time applications. The Michigan Connected Vehicle Testbed's Proof of Concept Testing report categorized CV data by time and spatial contexts, meaning that timestamp information and location information should be included in the CV data (*10*).

Application developers may require two kinds of data depending on the type of CV application, namely real-time disaggregated data and aggregated data. For example, a CV application, such as FCW, requires real-time disaggregated data for running and testing of the corresponding application, thus making it necessary for an application development platform to provide such data (*4*). On the other hand, a CV application like queue warning, which provides queue alerts every 5 minutes, may not require the disaggregated data, only aggregated data is sufficient (*11*). A CV environment will be one of the largest distributed networks in the future (*12*). As the size of the network grows (e.g., number of vehicles,

sensors, and roadside infrastructures), the demand for data will also increase (*13*). Thus, a platform for the CV application developers 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 relation to temporal and spatial requirements). Thus, in providing the data to the users, the CVDeP needs to meet the application requirements in relation to latency and throughput, and must be capable of handling the scalability issue related to the increasing number of CVs, in-vehicle sensors, and roadside infrastructures.

Access Control and Application Security

Security is one of the major concerns in deploying CV applications because of the safety-critical aspect of connected transportation systems (*14*, *15*). The U.S. DOT partnered with the automotive industry and industry security experts to design and develop a state-of-the-art security framework, and presented a security concept called Security Credential Management System (SCMS) to provide privacy and integrity to a CV system, as well as provide CV application security. The data shared between applications and edge devices need to be secured and it is necessary to maintain data confidentiality, integrity, and availability (*2*). 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., a communication device, such as an RSU) along with a computing device (e.g., a general-purpose processor) will provide a certificate to a CV application, which can be used by the application for exchanging messages (*16, 17*). A registration authority (RA) and a certificate authority (CA) were considered for providing the certificates. While an RA verifies the user request and checks the digital signature, a CA issues a new digital certificate or renews a certificate. This study adopted a security module for access control and credential management following the SCMS. The application security management is adopted based on security policies developed by Islam et al. (*18*). Although this study considered the access control and credential management, and application security, the network security is not a focus of this study.

Connected Vehicle Application Development Platform (CVDeP)

Figure 1 presents the conceptual development and implementation, and the evaluation and validation of the CVDeP. In an edge-centric CPS, the CVDeP architecture is developed including an application management platform and an application development graphical user interface for CV application development. The application management platform contains three modules: (i) control platform module; (ii) communication module; and (iii) data warehouse module. The application development graphical interface contains a graphical user interface through which an application developer can develop and deploy any CV application in the edge devices. The control platform module includes four submodules in total: (i) access control and credential management; (ii) application security management; (iii) data collection and distribution; and (iv) data broadcasting and receiving. The CVDeP was evaluated and validated using selected safety and mobility applications in two stages: (i) evaluation in an emulated environment; and (ii) field validation in a real-world edge-centric SC-CVT. The safety application is evaluated using communication and computational latency metrics. The mobility application is evaluated using communication and computational latency along with data transmission throughput (to test the scalability of the platform). Later, the experimental setup in the emulated and real-world environment and CV applications for the evaluation of the CVDeP are explained. In the following sub-sections, the study approach for developing and evaluating the CVDeP is presented in detail.

Conceptual Development and Implementation of CVDeP

In an edge-centric CPS, the physical proximity of devices to the data source reduces the wireless communication latency, and a layered architecture increases the scalability (*19*). The edge-centric CPS, as shown in Figure 2 for a CV system, consists of three edge layers: (i) mobile edge (e.g., on-board sensors and computing device inside a vehicle); (ii) fixed edge (e.g., roadside transportation data infrastructure); and (iii) system edge (e.g., backend server at TMC) (*5*). This hierarchical cyber-physical system architecture can address complexity and scale issues of CV systems. Participating CVs in the system will act as mobile edges and are equipped with a low-latency communication device. Although DSRC was considered in this study, any low-latency communication technology, such as 5G and long-term evolution (LTE) for Vehicles (LTE-V) can be incorporated in the development platform. 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 or Wi-Fi. A fixed edge can be extended to support a video camera and other sensing devices, such as weather sensors and GPS. A system edge can be a single endpoint in a cloud server. Fixed edges are connected to a system edge through a long-range communication option, such as optical fiber or LTE/Wi-Fi. Mobile edges (edge layer 1) can exchange data with fixed edges (edge

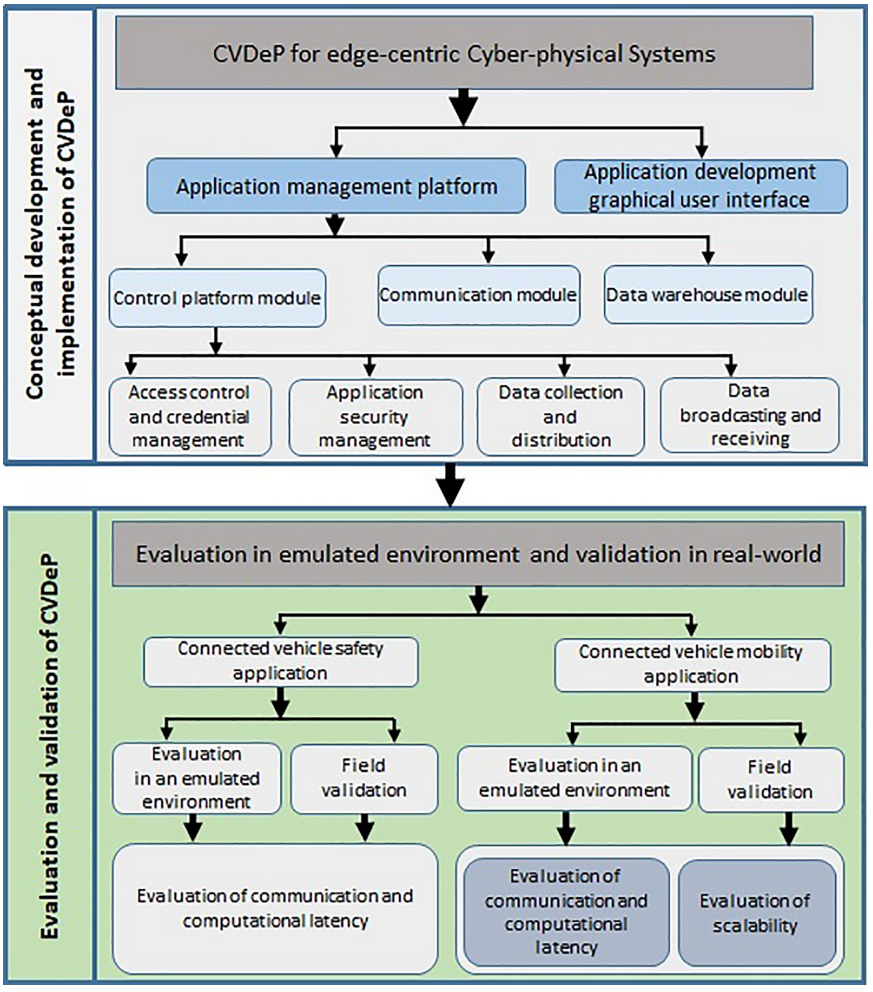

Figure 1. Approach for the Connected Vehicle Application Development Platform (CVDeP) development, evaluation, and validation.

layer 2) and system edges (edge layer 3) using DSRC and LTE/Wi-Fi communication, respectively.

In an edge-centric CPS for CVs, each component generates different types of data. For example, an OBU installed in a vehicle (i.e., mobile edge) broadcasts basic safety messages (BSMs), which contain a vehicle's information, such as location, speed, direction, acceleration, and braking status (*20*). A fixed edge collects data from the OBUs within its communication range, and acts as 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 edge layers. Edge layers can be accessed through an application development graphical user interface, which provides a way for a CV application developer to interact with the different edges. Figure 2 illustrates the architecture of the CVDeP for an edge-centric CPS, which comprises of application management platform and application development graphical user interface.

Application Management Platform

The application management platform is responsible for the selection of an appropriate communication medium for an application, and data collection, storage, broadcasting, and distribution, while providing the security of the platform by enabling secured access to the edge layers and security of the CV applications. As presented in Figure 2, application developers interact with the application management platform through an application development graphical user interface. The application management platform is a part of each edge layer of the edge-centric CPS. The application management platform is made up of the following modules: (i) control platform module; (ii) data warehouse module; and (iii)

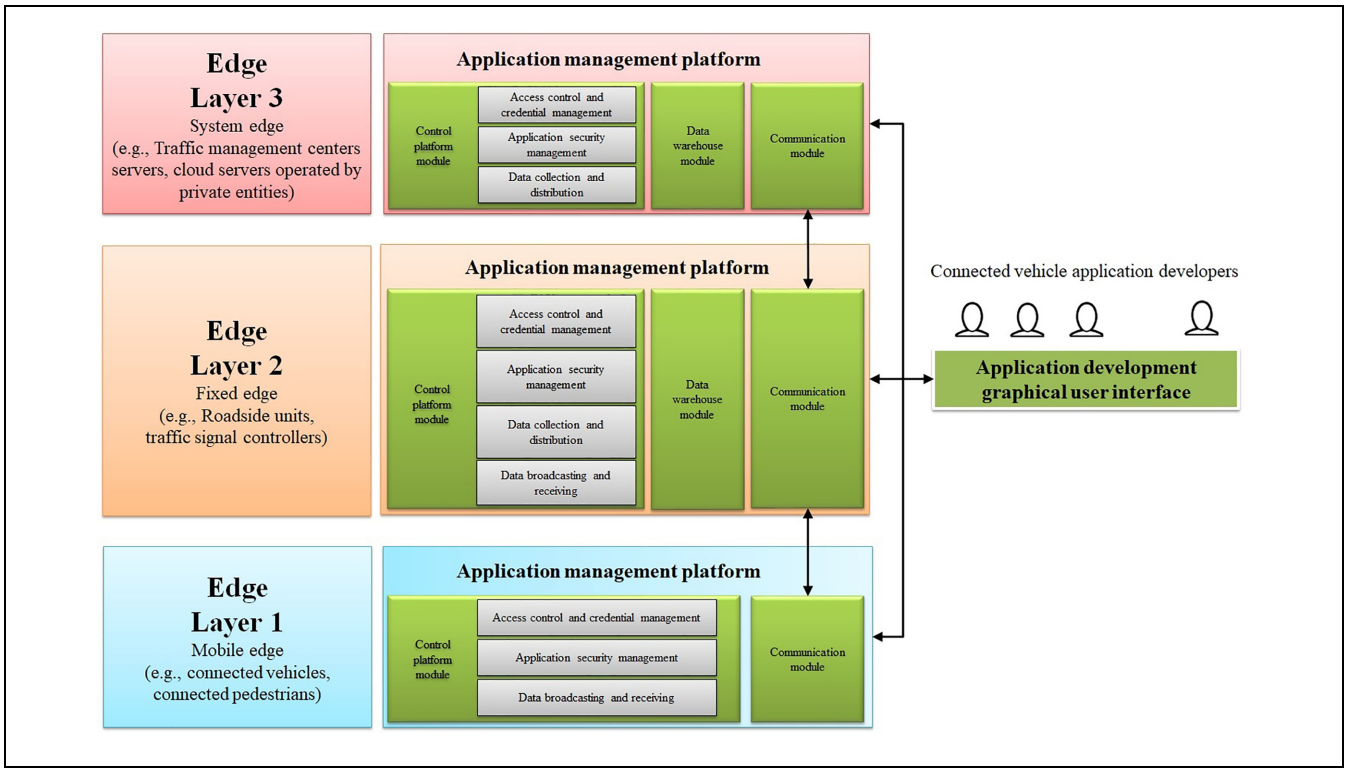

Figure 2. The Connected Vehicle Application Development Platform (CVDeP) architecture for an edge-centric cyber-physical system (CPS).

communication module. The following subsections describe the conceptual development and implementation of each of the modules in detail.

Conceptual Development of Control Platform Module. The control platform module of the system edge (edge layer 3) supports three types of sub-modules: (i) access control and credential management; (ii) application security management; and (iii) data collection and distribution. On the other hand, the control platform module of the fixed edge (edge layer 2) supports four types of sub-modules: (i) access control and credential management; (ii) application security management; (iii) data collection and distribution; and (iv) data broadcasting and receiving. However, the control platform module of mobile edge (edge layer 1) includes: (i) access control and credential management; (ii) application security management; and (iii) data broadcasting and receiving.

In an edge-centric CPS, edge devices continuously exchange data between different edges. The data broadcasting and receiving module in the mobile edges and fixed edges handles the continuous data exchange between other mobile edges and fixed edges. This module continuously broadcasts and receives messages that can be used to develop CV applications through application development graphical user interface. On the other hand, the data collection and distribution module in fixed edges and system edges are responsible for gathering and distributing data to and from mobile edges, fixed edges, and system edges in real-time. After the access control and credential management modules are activated, an authenticated application developer can access, gather, and visualize real-time streaming data generated from different edges of an edge-centric CPS. In addition, the 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 module contains the following sub-modules, and what sub-modules are included in each layer varies by whether the edge device is a mobile, fixed or system edge. Implementation overviews of these sub-modules are as follows:

- *Access control and credential management*. The access control and credential management sub-module ensures that only authorized users have access to CVDeP services. A CV application developer is authenticated via a login interface before giving 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 to sensors data, access to the data warehouse) like an android application system where permissions are written in a manifest file prior to developers developing an Android application (*21*). On the other hand, the credential management system (CMS) is 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. A digital signature is used to verify the authenticity of a message. The CMS is built in such a way that the functionalities of SCMS presented by the US DOT are replicated (*16, 22*). The assumptions of the National Highway Traffic Safety Administration (NHTSA)-supported CV pilot program were followed, where V2V messages are digitally signed with a digital signature, but not encrypted, and V2I messages are both signed and encrypted (*23*).

- *Application security management*. To provide the security for any applications, a data consumer and a data producer must be authenticated and completed certificate exchange (data flow1 [DF1])-and (data flow 2 [DF2]) to send any producer-generated data and receive any verified producer-generated data, respectively (as shown in Figure 3). The access control and credential management module is used to authenticate and exchange certificates to secure the access (as described before in the Access Control and Credential Management Module description) to any edge devices. As presented by Fernandez et al., a flow policy-based application security has been implemented in the application security management module, which contains trusted API and quarantine submodules (*24*). The current study has implemented the flow policies using "<*source, sink*>" tracking in which source is the producer of the data and sink is the intended consumer of that data (*24*). The trusted API submodule removes any sensitive information (e.g., driver's identify and vehicle ID of a mobile edge) from the producer-generated data (data flow 3 [DF3]). The quarantine submodule will remove any unexpected or malicious data flows between a producer and a consumer that is not listed in the flow policies. Flow policies can be pre-defined or can be changed by an administrator (e.g., a certificate authority) dynamically. Finally, verified data from a producer is passed to its intended consumer (data flow 4 [DF4]).
- *Data collection and distribution*. The data collection and distribution sub-module is the core part of the fixed and system edges of the CVDeP. Kafka (*25*) was selected as a broker-based data collection and distribution system because of the following efficacies: (i) high throughput; (ii) low-latency; (iii) reliability of data delivery; and (iv) scalability. In a publish-subscribe-based broker system, such as Kafka, Message Queuing Telemetry Transport (MQTT), or WebSphere, data producers (e.g., mobile edges, fixed edges, CV applications) produce and publish data to the broker, whereas the data consumers (e.g., fixed edge, CV applications) subscribe and consume the data available at the broker. By tagging individual data elements with a label based on a topic, producers (e.g., a CV) can produce data on a particular topic, and consumers (e.g., a CV application) can subscribe and consume the data of that topic. The broker receives data from producers and immediately makes the data available for consumers to consume. As a result, producers and consumers can generate and consume data, respectively, in an asynchronous and independent manner reducing the latency and improving reliability.
- *Data broadcasting and receiving*. The data broadcasting and receiving sub-module is developed for mobile edges and fixed edges, where it is responsible for broadcasting BSMs and receiving BSMs from other mobile edges and fixed edges. In this implementation, each mobile edge broadcasts BSMs at a default rate of 10 Hz and each BSM contains necessary attributes for safety

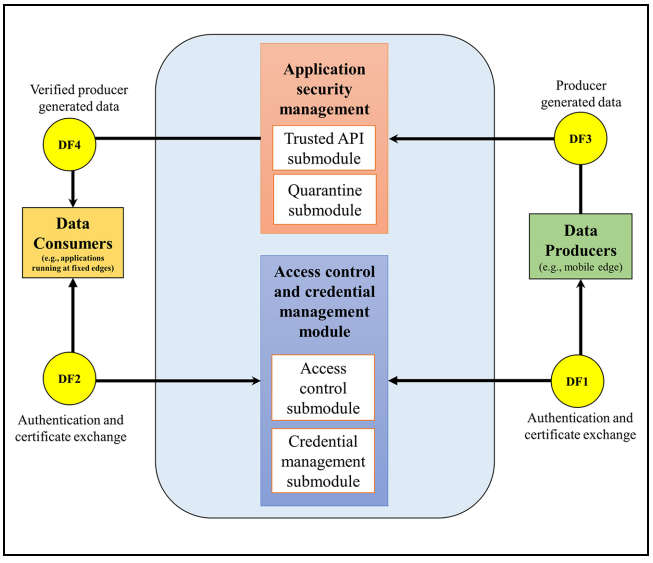

Figure 3. Implementation of application security management module, and access control and credential management module.

applications (e.g., position, speed, and direction) (*20, 26*). Additionally, each fixed edge broadcasts safety warnings (e.g., intersection safety warning) at a rate of 10 Hz, which are generated for V2I applications. In addition, each mobile edge and each fixed edge receives BSMs from all other mobile edges and fixed edges within their corresponding communication range.

Conceptual Development of Data Warehouse Module. The data warehouse module stores the data generated from different edge devices, roadside sensors, and applications deployed in the fixed and system edge layers. It is a distributed storage system that resides in fixed edges and system edges. The purpose of the data warehouse module is to store and provide necessary historical data that is needed by the CV applications. As a mobile edge is limited by computation power and storage size, a data warehouse module was not included in mobile edges. In fixed and system edges, the structure of the data warehouse module 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 this implementation, to support structured, semi-structured, as well as unstructured data, MySQL was used for structured data in a tabular format, and MongoDB was used for semi-structured and unstructured data in JSON format. The structured, semi-structured, and unstructured data together produce a huge amount of data in relation to volume. Realistically, CV applications do not need to access the raw data in their original format. Thus, a big data engineering infrastructure can be employed to reduce and compress raw data for further direct access by the CV applications. In this case, Clemson University's Cypress cluster was used for this purpose. Cypress is a Hadoop-based big data cluster and has both Hadoop Distributed File System for large-scale data storage and Apache Spark for big data processing (*27, 28*).

Conceptual Development of Communication Module. The communication module decides the best available communication medium based on the communication latency requirement of an application. Developers will provide the requirements of an application to the communication module through the application development graphical user interface, and then the communication module creates an abstraction layer to characterize communication network attributes of the available communication networks. For example, the communication module could select DSRC, 5G, or LTE-V, or any low-latency communication medium, from the available communication mediums to satisfy the requirement of safety applications. While the application is running in an edge device, the CVDeP will provide communication metadata (e.g., available communication mediums, such as DSRC, 5G, LTE, LTE-V, and Wi-Fi, and their average, maximum, and minimum transmission latency and throughput) for evaluating the performance of the application. The decision for selecting a wireless communication medium, by the communication module, will be completed based on the characteristics of available communication mediums and the application requirements set by the application developers.

Implementation of Communication Module. The communication module manages the underlying communication network connectivity in an edge-centric CPS. The communication network services are implemented in the network layer of each edge device to manage the connectivity using the available communication mediums to connect with other edge devices. In the communication module implementation, the discovery or searching of communication mediums and their network characteristics are measured asynchronously. The communication module selects a medium to use for transmitting and receiving data based on the application requirements. A metadata support layer was added in the communication module to provide metadata to the application 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 the CVDeP provides communication attributes to the developers through the application development graphical user interface, and developers can evaluate the performance of an application through these communication attributes.

Application Development Graphical User Interface

Application developers can access the underlying edge devices of the edge-centric CPS using a graphical user interface, and can develop and deploy any CV application directly on the edge-centric CPS. Based on access

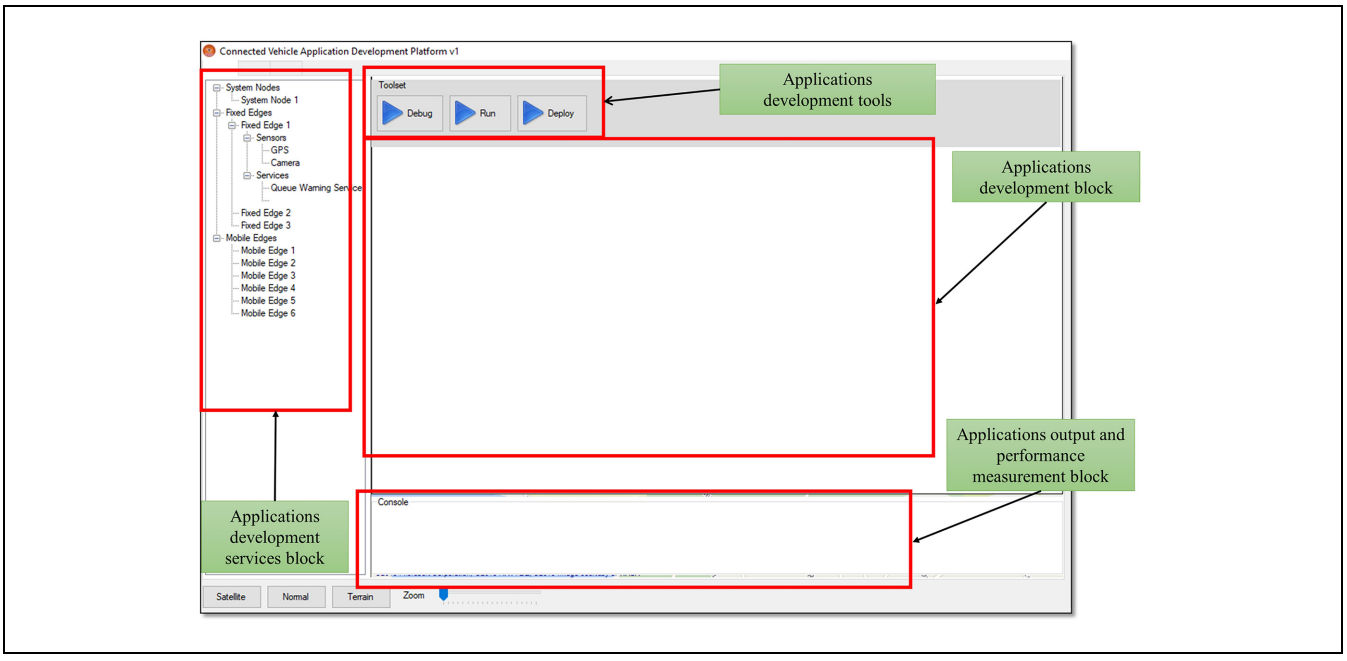

Figure 4. Implementation of application development graphical user interface.

control rights to the available services (e.g., communication services and data storage service) of the platform and the requirements of a CV application, an application developer can access different types of data (e.g., real-time and historical data) from each layer through an application development graphical user interface. Using this application development graphical user interface, application developers can also request any specific data for a specific application purpose. For example, developers can request historical data from the data warehouse module to predict future roadway traffic conditions. Application development graphical user interface will provide an interactive platform to the developers to build their own applications and test these applications by requesting real-time data from both mobile and fixed edges, and historical data from the data warehouse module from both fixed and system edges.

As shown in Figure 4, the application development graphical user interface is divided into four blocks: (i) applications development services block [using this block a developer can connect to the edge devices through an authentication procedure using the accessibility details, such as username and password. After the authentication procedure, developers will be provided with a list of available edge devices (e.g., location, number, and type of edge devices), services (e.g., available communication mediums and their characteristics), and sensors (e.g., GPS, camera) of each edge device]; (ii) applications development block (inside this block, an application developer can implement an application in an edge device using Python or C++); (iii) applications development tools (using this block, an application developer can develop, deploy, test, and debug an application in edge devices); and (iv) applications output and performance measurement block (after deploying an application, developers can visualize, and save the output and performance data of an application through this block). The application development graphical user interface is developed as a desktop application in C# (C sharp) as illustrated in Figure 4. Currently, the software has been developed for the Windows operating systems as a proof-of-concept.

Experimental Setup

This section provides a description of the experimental setup in an emulated environment as well as a real-world environment to evaluate the efficacy of the 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. As shown in Figure 5, a developer can emulate mobile edges using hardware setups #1 and #2 and fixed edges using hardware setup #3, where system edges are set up in a dedicated server at Clemson University. Each hardware setup (#1, #2, and #3) consists of one DSRC unit to send and receive the DSRC messages, and a computing device for computation. Hardware setup #1 is

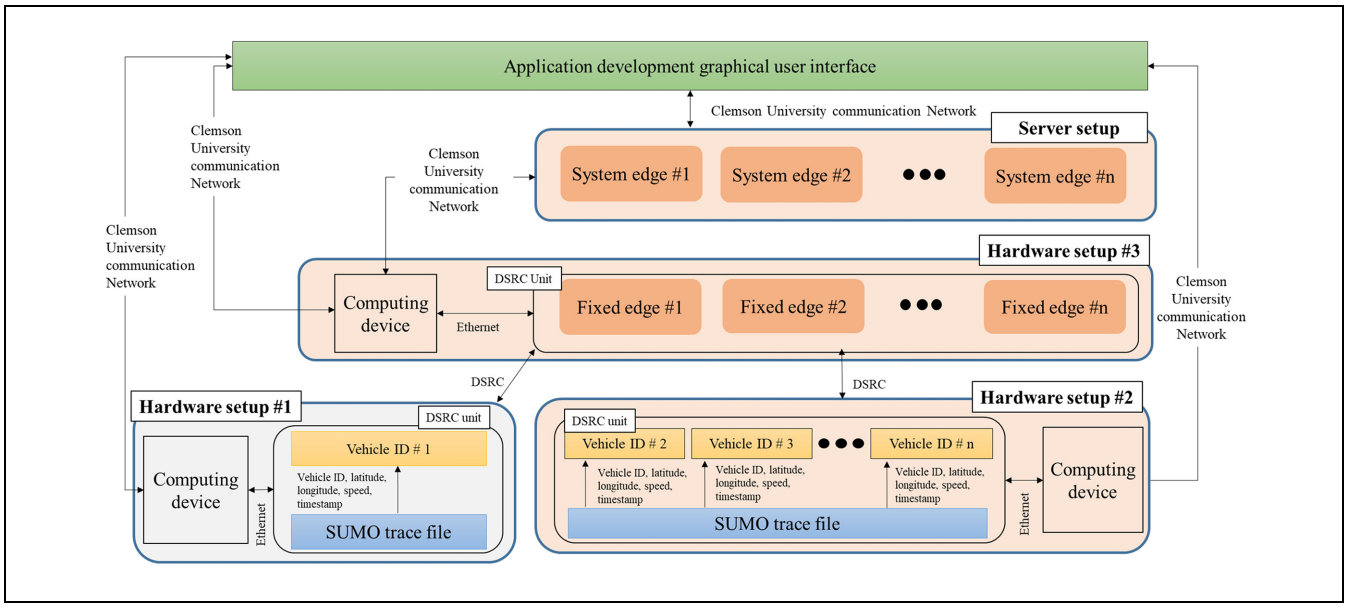

Figure 5. Connected Vehicle Application Development Platform (CVDeP) setup in an emulated edge-centric cyber-physical system (CPS).

used for developing the safety application, whereas hardware setup #2 is used for emulating other mobile edges for the safety application. For mobility and environmental applications, 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 is defined by a developer through the application development graphical user interface. A dedicated server located in Clemson University is used for creating system edge instances. In this emulated edge-centric CPS, mobile edges and fixed edges communicate with each other using DSRC, and fixed edges and system edges communicate using the Clemson University communication network, which includes an optical fiber and Wi-Fi connections. In addition, developers can configure the number of edges in each layer as required by an application. To generate the movement data of mobile edges, the movement of the mobile edges is exported from the Simulation of Urban Mobility (SUMO), which is a microscopic traffic simulator software, as a SUMO trace file (*29*). 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 reads that trace file and generates BSMs for each vehicle. Then, these BSMs are broadcast using DSRC to each vehicle. Fixed edges will receive BSMs from mobile edges within their corresponding communication ranges. Developers can access the edges through the CVDeP application development graphical user interface to develop and evaluate the performance of the developed CV application.

All the modules of the CVDeP were implemented in each layer, as shown in Figure 6. Hardware setups #1 and #2 represent the edge layer 1, Hardware setup #3 represents the edge layer 2, and the Server setup represents the edge layer 3 of an edge-centric CPS. The implemented modules of the CVDeP are: (i) control platform module, which consists of access control and credential management, application security management, data collection and distribution, and data broadcasting and receiving; (ii) communication module; and (iii) data warehouse module. The control platform module resides in a computing device and is implemented in each hardware setup. However, the data broadcasting and receiving sub-module of the control platform module resides in a computing device, which is a part of each mobile and fixed edges. For the data warehouse module, an external hard disk drive was used for storing data in the fixed edges, and cloud storage was used for storing data in the system edge. In this case, an application developer interacts with each hardware though the Clemson University communication network to develop, debug, and test a CV application.

Experimental Setup in SC-CVT

The SC-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 (*19*). The backend server is located at Clemson University and connected to the Clemson University communication network. Two of the fixed edges are connected to the Clemson University network with an optical fiber link and one fixed edge is connected to the Clemson University network with a Wi-Fi link. Each fixed edge has its own DSRC radio to communicate with

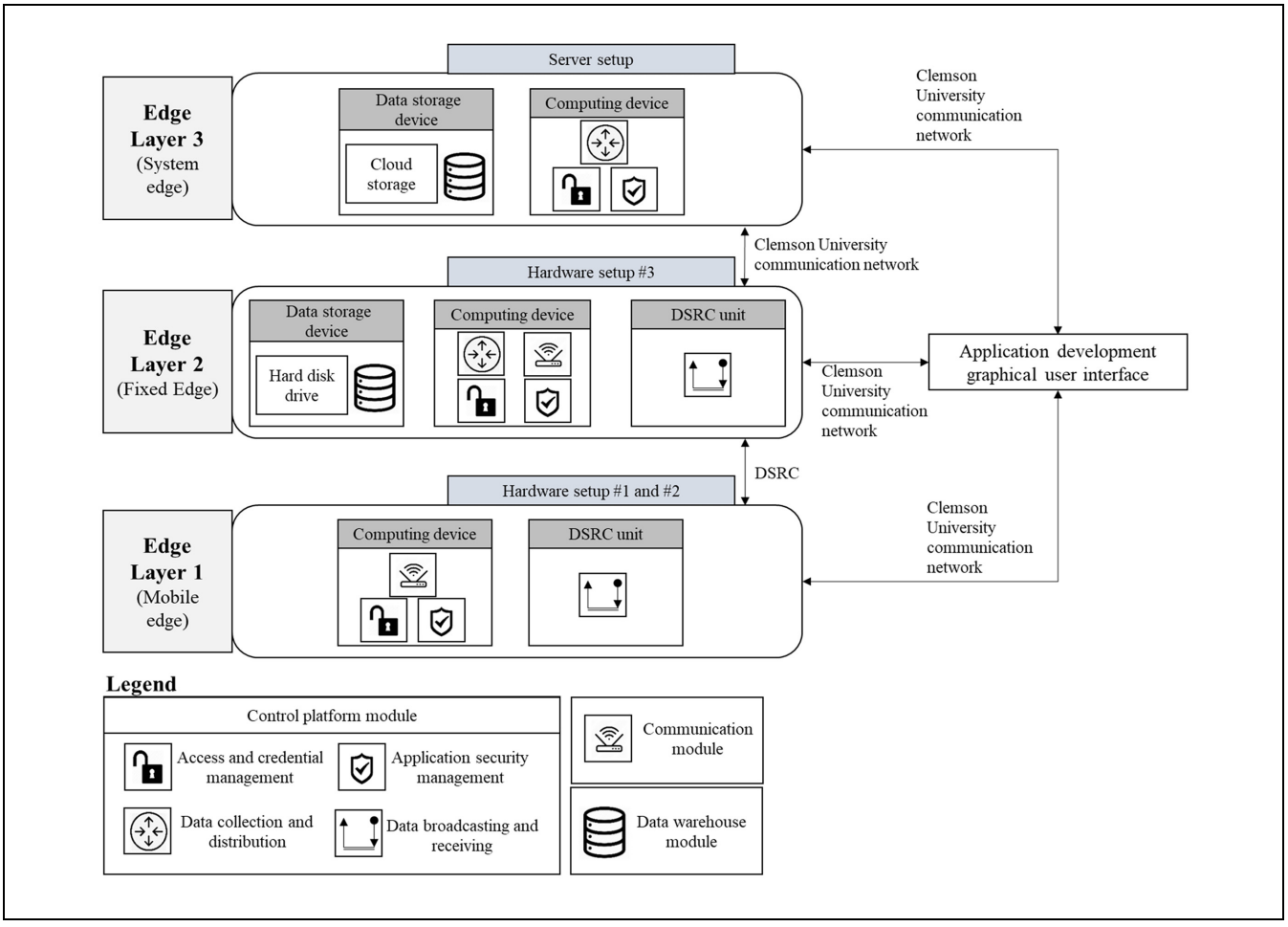

Figure 6. Implementation of Connected Vehicle Application Development Platform (CVDeP) modules in an emulated edge-centric cyber-physical system (CPS).

mobile edges. Each mobile edge (primarily OBUs on vehicles) is equipped with wireless communication devices. In this case, DSRC-enabled OBUs were used, although any low-latency communication mediums, such as 5G or LTE-V, can be used. As per the definition of a mobile edge, a CV will act as a mobile edge and a vehicle owner will own a commercially available low-latency communication device (e.g., DSRC, 5G, or LTE-V enabled communication device) along with a computing device for running an application at the vehicle level. Also, a vehicle owner can install these communication and computing devices to create a mobile edge.

Evaluation and Validation of CVDeP

For the experiments, an FCW was developed as a safety application and vehicle data for traffic operations as a mobility application using the CVDeP (*2*). Then, to prove the efficacy of the CVDeP, the FCW and vehicle data for traffic operations applications are evaluated in an emulated environment and the real-world SC-CVT (*19*).

Safety Application

For the experiment related to safety application, FCW was selected that considers two vehicles moving in the same direction on the same lane in an uncongested urban traffic condition. The FCW application is based on the study by Xiang et al., where the FCW application uses a vehicle kinematics (VK) model for generating collision warnings using DSRC communication (*30*). Based on the VK model, the FCW application generates rear-end collision warnings when two vehicles are closer than a defined safe distance. In this study, Equation 1 is used for implementing an FCW application as suggested by Xiang et al. (*30*).

$$D_w = \frac{(V_o - V_t)^2}{2 * a} + d \tag{1}$$

where, D_w is the distance threshold for collision warning; V_o is the preceding vehicle's speed; and V_t is the follower vehicle's speed. The follower vehicle is the vehicle where the FCW application is intended to run; d is

Table 1. Summary of Latency for Forward Collision Warning (FCW) Application Evaluation

Experimental setup	Evaluation latency parameter	End-to-end latency in scenario #1[a]	End-to-end latency in scenario #2[b]	Latency requirements for safety application (*17, 33, 34*)
Emulated environment	Maximum	97 ms	79 ms	⩽200 ms
	Average	18 ms	18 ms	
	Minimum	4 ms	4 ms	
SC-CVT	Maximum	115 ms	107 ms	
	Average	65 ms	51 ms	
	Minimum	4 ms	5 ms	

Note: SC-CVT = South Carolina connected vehicle testbed.
[a]***Scenario #1*:** The preceding vehicle and follower vehicle are moving in the same direction on the same lane at 20 mph and 30 mph, respectively.
[b]***Scenario #2*:** The preceding vehicle and follower vehicle both are moving on the same lane at 30 mph, and then the preceding vehicle stops suddenly.

calculated by adding the half of the length of the preceding vehicle with half of the length of the following vehicle, and a is set to 11.2 ft/s^2 (*31*). Given the emulated environment within the CVDeP platform, as shown in Figure 5, it is possible to generate complex urban scenarios, and develop and evaluate appropriate FCW application corresponding to such scenarios. Using a complex urban scenario, an application developer can develop an FCW application considering different safety constraints within that environment.

Evaluation Scenarios. Two evaluation scenarios were created for evaluating the CVDeP as a safety application development platform.

- ***Scenario 1***: The preceding vehicle (hardware setup #2 in Figure 5), and follower vehicle (hardware setup #1 in Figure 5) are moving in the same direction on the same lane at 20 mph and 30 mph, respectively.
- ***Scenario 2*:** The preceding vehicle and follower vehicle both are moving on the same lane at 30 mph, and then the preceding vehicle stops suddenly.

In both scenarios, the FCW application is deployed in the follower vehicle, and FCWs 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, data delivery latency was considered as a measure of effectiveness. In this context, latency is the duration between the time when a BSM is generated by a mobile edge and the time when the application produces an FCW message in the follower vehicle. Here, latency includes network latency, computational latency, and communication medium selection latency.

Evaluation in Emulated Environment. The FCW application was evaluated using the experimental setup as described in the previous section. The application is developed using the CVDeP, and then the application is tested using two evaluation scenarios. Table 1 provides a summary of latency recorded from both evaluation scenarios. For the evaluation of the FCW application in the emulated environment, the BSMs of 200 s observation period containing 4000 BSMs were analyzed from two mobile edges to calculate the maximum, minimum, and average latency. A CV broadcasts BSMs and receives BSMs from other CVs within its communication range. A CV safety application's critical latency requirement represents the maximum acceptable time from generating BSMs by a preceding vehicle to generating an FCW message by a follower vehicle within the preceding vehicle's communication range. If an FCW message is received by the driver of the follower vehicle within this safety-critical latency requirement, the driver can take action to avoid a collision after receiving an FCW (*32*). In this case, 200 ms was selected as a maximum safety-critical latency requirement in which a driver is able to decelerate at a deceleration rate of 11.2 ft/s^2, and avoid the forward collision if the warning message was delivered within 200 ms (*19*, *31*). Therefore, the maximum end-to-end latency requirement is considered as 200 ms, which will ensure the driver is able to stop the vehicle in case of a forward collision scenario. In the emulated experimental environment, it was found that the average latency is 18 ms for both evaluation scenario 1 and scenario 2. However, the recorded maximum latencies were 97 ms and 79 ms, for scenarios 1 and 2, respectively, which are below the safety-critical latency requirement for CVs (i.e., 200 ms) (*33*). In Table 1, the end-to-end latency is presented, which includes communication network latency, computational latency, and communication medium selection latency. The computational latency for running the application is 1.5 ms, which is the same for

both evaluation scenarios. In addition, these FCW messages are sent to the mobile edge using the best available communication medium as decided by the communication module, which takes about 0.5 ms on average to make such a determination. During this communication medium selection process, all communication mediums (LTE, Wi-Fi, and DSRC) were running simultaneously, and the communication module was monitoring these mediums asynchronously and selected the best communication medium for a CV application following the heterogeneous wireless networking concept for CVs (*33*).

Field Validation in SC-CVT. For the field evaluation of FCW application in the SC-CVT, a similar speed for the corresponding vehicles was followed for both evaluation scenarios. The end-to-end latency for the FCW application was measured in the field for both evaluation scenarios. Table 1 provides a summary of end-to-end latency recorded for both evaluation scenarios in the field experiments and in an emulated environment. Similar to the evaluation in an emulated environment, the data sample of 200 s containing 4000 BSMs from two mobile edges was analyzed to calculate the maximum, minimum, and average latency. The average end-to-end latency measured is 65 ms and 51 ms for scenarios 1 and 2, respectively. The maximum end-to-end latency recorded for the test is 115 ms and 107 ms for scenarios 1 and 2, respectively, which is below the safety-critical latency requirement (i.e., 200 ms) (*33*). In the field experiment, a higher latency than the latency measured in the emulated experimental setup was observed, because of the surrounding environmental effect or wireless communication propagation loss. Table 1 presents the end-to-end latency which includes the network latency, computational latency, and communication medium selection latency. In both cases (scenarios 1 and 2), it can be validated that the application developed using the CVDeP was able to satisfy the application's safety-critical latency requirement ($\leqslant$ 200 ms) in the field experiments.

Mobility Application

The CVDeP was evaluated using the vehicle data 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 (*2*). This application is divided 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. The sub-application 1 runs in each fixed edge and the sub-application 2 runs in the system edge.

The scalability of the CVDeP was evaluated to ensure the CV application requirements are met in relation to latency and throughput. Here, the latency is the time difference between the time of data generation at the edge-centric SC-CVT and the time when the data is received by the users (e.g., CV applications). Data delivery latency requirements for any mobility and environmental applications must be satisfied to provide mobility and environmental services. As the CVDeP aims to support different mobility and environmental applications, 1000 ms was considered as the maximum latency threshold to deliver the data from edge devices to the data consumers (e.g., CV applications) following the recommendations from Ahmed-Zaid et al. (*17*). This 1000 ms will enable the near real-time operation of mobility applications, such as queue warning and traffic rerouting applications. However, if the latency recommendations change in the future for any CV applications, the CVDeP can still be utilized by selecting appropriate underlying technologies for different communication and computing devices to meet any new requirements. The CVDeP provides a general architecture, which is independent of specific technologies. The experiments demonstrate the efficacies of the CVDeP as an application development platform using selected communication and computing technologies. Also, it is necessary to ensure a high throughput (i.e., the data transfer rate), which means the high use of the allocated bandwidth. This platform already satisfied the spatial requirement of the application, as mobile edges will be within the communication range of fixed edges.

Evaluation Scenarios. Two different scenarios were created for evaluating the application development platform by varying the number of fixed edges and the number of mobile edges.

- ***Scenario 1***: One system edge and one fixed edge with varying numbers of mobile edges (i.e., 5, 10, 20, 30, 50, 100, 150, and 200).
- ***Scenario 2*:** One system edge with varying numbers of fixed edges (i.e., 1, 2, and 3) and 200 mobile edges for each fixed edge.

For evaluation scenario 2, based on a fixed edge's communication range, the maximum number of CVs on Perimeter Road approaching the intersection is 200 vehicles per hour per lane (vphpl) during a congested traffic condition. For the evaluation in the emulated environment, SUMO was used to generate the movement data of mobile edges, and the traffic network was calibrated so that traffic volume data from SUMO simulation matches, within a tolerance level of 5%, with the field-collected data. For both scenarios, the scalability of the application development platform was evaluated in relation to data delivery latency and throughput.

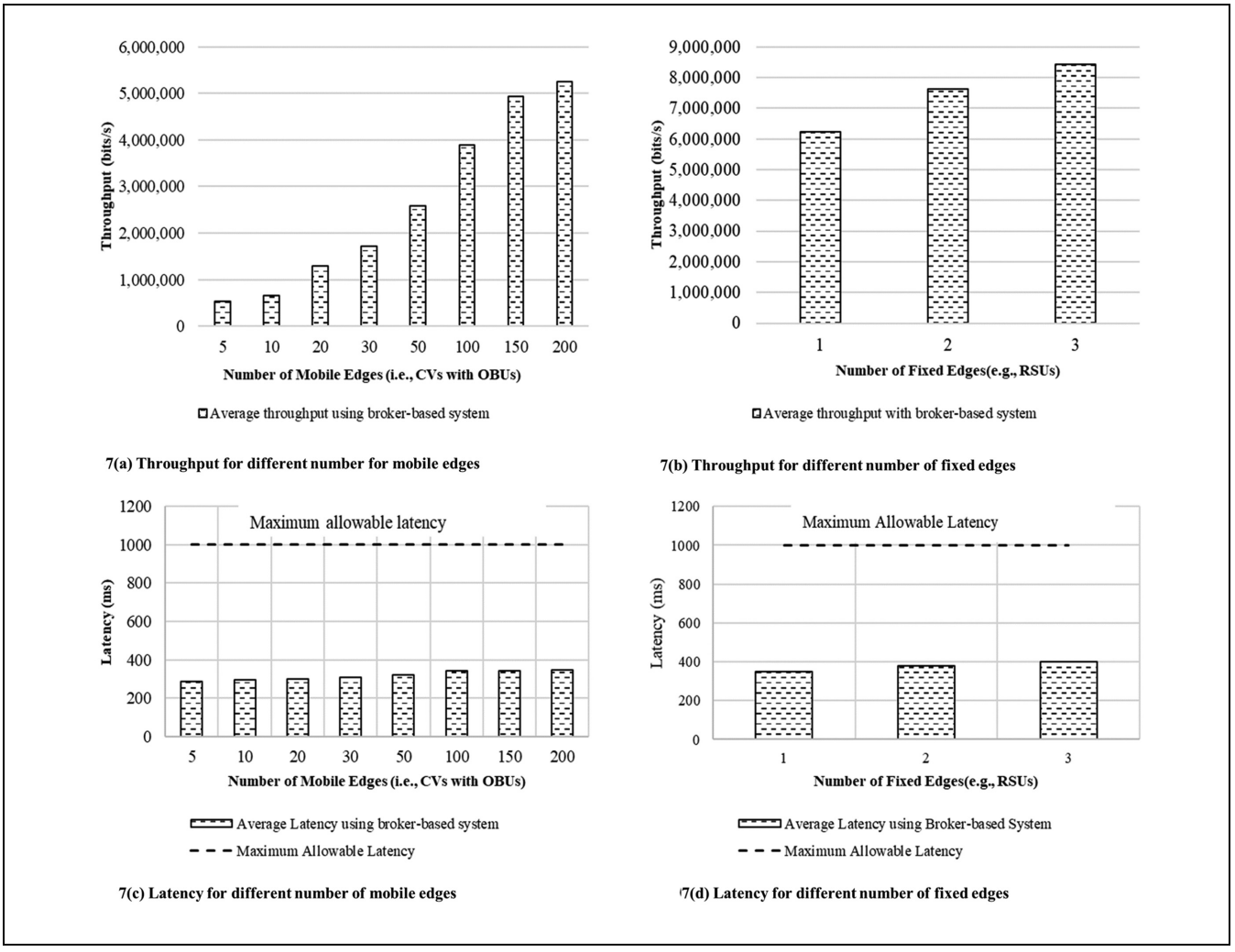

Figure 7. Evaluation of Connected Vehicle Application Development Platform (CVDeP) for mobility application using application throughput and latency.

Evaluation in Emulated Environment. A data collection and distribution system (a broker-based system) that is required for the real-time application development platform was implemented. The scalability of the CVDeP was evaluated considering a data collection and distribution system, which is a broker-based system. In addition, the recorded end-to-end latency was compared with the latency requirement for the selected CV mobility application. As shown in Figure 7, *a* and *b*, with the increasing number of mobile edges and fixed edges, the throughput of the broker-based system is linearly increasing and reaches a maximum at 5.2 Mbits/s and 8.4 Mbits/s, respectively. Higher throughput ensures reliable and scalable services. The broker-based system (e.g., Kafka for this experiment) uses an asynchronous mode that can collect and distribute data in memory and send them in batches in a single shot (*25*, *34*). Because of this asynchronous mode and sending data in batches, the broker-based system can provide the required throughput. The broker-based system can adapt the throughput requirement by the application as the number of mobile edges and fixed edge increases, and thus can handle more data as needed (*4*).

It was observed that the CVDeP data collection and distribution system can maintain a lower latency with the increasing number of mobile edges (Figure 7*c*) and fixed edges (Figure 7*d*). The increment of latency with the broker-based method is negligible for both 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 into memory) (*34*). Thus, the broker-based system can maintain a lower message delivery latency irrespective of the number of producers and consumers thus ensuring scalability. In this experiment, the default configuration of a Kafka

Table 2. Summary of Latency for Mobility Application with Five Connected Vehicles (CVs)

Evaluation latency parameter	End-to-end latency: Evaluation in emulated environment	End-to-end latency: Validation in SC-CVT	Latency requirements for mobility application (*10*)
Maximum	115 ms	267 ms	
Average	65 ms	69 ms	⩽ 1000 ms
Minimum	4 ms	6 ms	

Note: SC-CVT = South Carolina connected vehicle testbed.

broker-based system was used (*25*). However, the configuration (e.g., topic partitions, replication number, and number of brokers) of a 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 Du et al., the CVDeP can be scaled up to receive data from and share data with additional connected data sources (e.g., personal handheld devices, news media and weather stations) (*4*).

Field Validation in SC-CVT. The CVDeP was evaluated in SC-CVT using five mobile edges (e.g., CVs) in the field experiment. Table 2 shows the summary of end-to-end latency when the application in the CVDeP emulated environment and SC-CVT was developed. Higher latency (maximum, average, and minimum) was observed in the field than in the emulated environment. In the field experiment, the data exchange using DSRC technology between the mobile edges and fixed edges was affected by the environmental interferences, such as trees, roadway slope, and curvature. This causes a higher variation in latency in the field than in the emulated environment. However, the latency observed in the field was still far below the application latency requirement ($\leqslant$ 1000 ms) for the selected mobility application.

Conclusions and Future Work

CV technology holds the promise of improving the traffic safety and efficiency of roadway traffic operations. To materialize CV benefits, the active participation of CV researchers and developers is necessary. This can be hindered because of the lack of real-world application development platforms that use real-world and real-time data to support the CV application development process including testing and debugging. Using the CVDeP, the CV application developers can interact with real-world edge devices, and develop, test, and debug CV safety and mobility applications. From these experiments, it was revealed that the applications developed using the CVDeP were able to satisfy the CV safety and mobility application latency requirements and maintain the required throughput both for an increasing number of mobile edges and fixed edges. It was shown that the FCW application (a safety application) developed using the CVDeP can satisfy the safety-critical latency requirement (under 200 ms for an FCW application). Also, the vehicle data for traffic operations application (a mobility application) developed using the CVDeP with a broker-based system shows about 400 ms of latency with three fixed edges and 600 mobile edges, which is much lower than the latency requirement (under 1,000 ms) of mobility applications. This proves the scalability of the CVDeP while satisfying the latency requirement of CV applications for an edge-centric CPS. The authors published the source code of the CVDeP in the Github platform.

As the CVDeP is being refined further, the authors' follow-up studies of CVDeP include: (i) evaluation of the fault tolerance and resiliency of the platform; (ii) evaluation of multiple applications running simultaneously in multiple system edges, and merging information from diverse data sources of a large roadway network (i.e., data residing at local or city/county level, regional or state level, and national level); (iii) incorporation of data from other traditional data sources (e.g., traffic signals, video detectors, or loop detectors) and non-traditional data sources (e.g., news media, weather sensors, social networking sites); and (iv) strategy identification to make the system more secure by incorporating different security threat detection and protection mechanisms against different malicious activities including cyber-attacks.

Acknowledgments

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: M. Islam, M. Rahman, S. Khan, M. Chowdhury, L. Deka; data collection: M. Islam, M. Rahman; interpretation of results: M. Islam, M. Rahman, S. Khan, M. Chowdhury, L. Deka; draft manuscript preparation: M. Islam, M.

Rahman, S. Khan, M. Chowdhury, L. Deka. All authors reviewed the results and approved the final version of the manuscript.

Declaration of Conflicting Interests

The author(s) declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

Funding

The author(s) disclosed receipt of the following financial support for the research, authorship, and/or publication of this article: This study is based upon work supported by the Center for Connected Multimodal Mobility (C^2M^2) (a US Department Transportation Tier 1 University Transportation Center) headquartered at Clemson University, Clemson, South Carolina, US.

References

1. Sotelo, M. A., J. W. C. van Lint, U. Nunes, L. B. Vlacic, and M. Chowdhury. Introduction to the Special Issue on Emergent Cooperative Technologies in Intelligent Transportation Systems. *IEEE Transactions on Intelligent Transportation Systems*, Vol. 13, No. 1, 2012, pp. 1–5.
2. ARC-IT. *Architecture Reference for Cooperative and Intelligent Transportation*. 2019 https://local.iteris.com/arc-it/index.html. Accessed February 26, 2020.
3. ITS JPO U. *The Operational Data Environemnt (ODE)*. https://github.com/usdot-jpo-ode/jpo-ode. Accessed February 26, 2020.
4. 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. *IEEE Transactions on Intelligent Transportation Systems*, Vol. 19, No. 3, 2018, pp. 787–801.
5. Rayamajhi, A., M. Rahman, M. Kaur, J. Liu, M. Chowdhury, H. Hu, J. McClendon, K. C. Wang, A. Gosain, and J. Martin. ThinGs in a Fog: System Illustration with Connected Vehicles. *Proc., 2017 IEEE 85th Vehicular Technology Conference (VTC Spring)*, Sydney, Australia, 2017;
6. Lopez, P. G., A. Montresor, D. Epema, A. Datta, T. Higashino, A. Iamnitchi, M. Barcellos, P. Felber, and E. Riviere. Edge-Centric Computing: Vision and Challenges. *ACM SIGCOMM Computer Communication Review*, Vol. 45, No. 5, 2015, pp. 37–42.
7. Grewe, D., M. Wagner, M. Arumaithurai, I. Psaras, and D. Kutscher. Information-Centric Mobile Edge Computing for Connected Vehicle Environments: Challenges and Research Directions. *Proc., Workshop on Mobile Edge Communications*, ACM, 2017, pp. 7–12.
8. Islam, M. *CVDEP*. https://github.com/mahfuz195/CVDEP. Accessed February 6, 2020
9. 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 & Tutorials*, Vol. 13, No. 4, 2011, pp. 584–616.
10. *Southeast Michigan Test Bed 2014 Concept of Operations*. U.S. Department of Transportation, 2010.
11. Balke, K., H. Charara, and S. Sunkari. *Report on Dynamic Speed Harmonization and Queue Warning Algorithm Design*. Texas A&M Transportation Institute, Texas A&M University System, College Station, 2014.
12. Qian, Y., and N. Moayeri. Design of Secure and Application-Oriented Vanets. *Proc., VTC Spring 2008 - IEEE Vehicular Technology Conference*, Singapore, IEEE, New York, 2008, pp. 2794–2799.
13. Baker, E. H., D. Crusius, M. Fischer, W. Gerling, K. Gnanaserakan, H. Kerstan, F. Kuhnert, J. Kusber, J. Mohs, M. Schulte, J. Seyfferth, J. Stephan, and T. Warnke. *Connected Car Report 2016: Opportunities, Risk, and Turmoil on the Road to Autonomous Vehicles*. Connected Car Report, Strategy, New York, 2016.
14. Zarki, M. E., S. Mehrotra, G Tsudik, and N. Venkatasubramanian. Security Issues in a Future Vehicular Network. *Symp A Q J Mod Foreign Lit*, Vol. 35, 2002, pp. 270–274.
15. 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.
16. Whyte, W., A. Weimerskirch, V. Kumar, and T. Hehn. A Security Credential Management System for V2V Communications. *Proc., 2013 IEEE Vehicular Networking Conference*, Boston, Mass., IEEE, New York, 2013, pp. 1–8.
17. 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. *Vehicle Safety Communications–Applications (VSC-A) Final Report: Appendix Volume 3 Security*. National Highway Traffic Safety Administration, Washington, D.C., 2011.
18. Islam, M., M. Chowdhury, H. Li, and H. Hu. Cybersecurity Attacks in Vehicle-to-Infrastructure Applications and Their Prevention. *Transportation Research Record: Journal of the Transportation Research Board*, 2018. 2672: 66–78.
19. 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: Journal of the Transportation Research Board*, 2018. 2672: 10–23.
20. 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.
21. Felt, A. P., E. Ha, S. Egelman, A. Haney, E. Chin, and D. Wagner. Android Permissions: User Attention, Comprehension, and Behavior. *Proc., 8th Eighth Symposium on Usable Privacy and Security*, 2012. http://dl.acm.org/citation.cfm?doid = 2335356.2335360.
22. 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.

23. Weil, T. VPKI Hits the Highway: Secure Communication for the Connected Vehicle Program. *IT Professional*, Vol. 19, No. 1, 2017, pp. 59–63.
24. Fernandes, E., J. Paupore, A. Rahmati, D. Simionato, M. Conti, and A. Prakash. FlowFence: Practical Data Protection for Emerging IoT Application Frameworks. *Proc., 25th {USENIX} Security Symposium ({USENIX} Security 16)*, 2016, pp. 531–548.
25. Kreps, J., N. Narkhede, and J. Rao. Kafka: A Distributed Messaging System for Log Processing. *Proc., ACM SIGMOD Work Netw Meets Databases*, 2011, p. 6.
26. 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.
27. Hadoop. *HDFS Architecture Guide*. https://hadoop.apache.org/docs/r1.2.1/hdfs_design.html.
28. Zaharia, M., M. Chowdhury, M. J. Franklin, S. Shenker, and I. Stoica. Spark: Cluster Computing with Working Sets. *Proc., HotCloud'10: Proceedings 2nd USENIX Conference on Hot Top Cloud Computing*, 2010, p. 10.
29. DLR - Institute of Transportation Systems. *Eclipse SUMO – Simulation of Urban Mobility*. 2017. http://www.dlr.de/ts/en/desktopdefault.aspx/tabid-9883/16931_read-41000/. Accessed February 6, 2020.
30. 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, 3, 2014, 1054–1065.
31. *A Policy on Geometric Design of Highways and Street*, 6th ed. AASHTO, Washington, D.C., 2011.
32. Qing, X., R. Segupta, D. Jiang, and D. Chrysler. Design and Analysis of Highway Safety Communication Protocol in 5.9 GHz Dedicated Short Range Communication Spectrum. *Proc., The 57th IEEE Semiannual Vehicular Technology Conference*, 2003. VTC 2003-Spring, Jeju, South Korea, IEEE, 2003, pp. 2451–2455.
33. Dey, K. C., 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://www.sciencedirect.com/science/article/pii/S0968090X16300018.
34. *Apache Kafka*. https://kafka.apache.org/. Accessed February 6, 2020.

Any opinions, findings, and conclusions or recommendations expressed in this paper are those of the author(s) and do not necessarily reflect the views of the Center for Connected Multimodal Mobility (C^2M^2), and the U.S. Government assumes no liability for the contents or use thereof.