BIG DATA ANALYTICS FOR CONNECTED VEHICLE DATA INFRASTRUCTURE RESILIENCE

FINAL REPORT

SOUTHEASTERN TRANSPORTATION CENTER

Mashrur “Ronnie” Chowdhury, PH.D., PE, F.ASCE
Kakan Dey, Ph.D.
Amy Apon, Ph.D.
Andre Luckow, Ph.D.
Linh Bao Ngo, Ph.D.
Yuheng Du, Ph.D. student
Mizanur Rahman, Ph.D. student

May, 2017

US DEPARTMENT OF TRANSPORTATION GRANT
DTRT13-G-UTC34
DISCLAIMER

The contents of this report reflect the views of the authors, who are responsible for the facts and the accuracy of the information presented herein. This document is disseminated under the sponsorship of the Department of Transportation, University Transportation Centers Program, in the interest of information exchange. The U.S. Government assumes no liability for the contents or use thereof.
A complex and vast amount of data will be collected from on-board sensors of operational connected vehicles (CVs), infrastructure data sources such as roadway sensors and traffic signals, mobile data sources such as cell phones, social media sources such as Twitter, and news and weather data services. Unfortunately, these data will create a bottleneck at data centers for processing and retrievals of collected data, and will require the deployment of additional message transfer infrastructure between data producers and consumers to support diverse CV applications. In this paper, we present a strategy for creating an efficient and low-latency distributed message delivery system for CV applications using a distributed message delivery platform. This strategy enables large-scale ingestion, curation, and transformation of unstructured data (roadway traffic-related and roadway non-traffic-related data) into labeled and customized topics for a large number of subscribers or consumers, such as CVs, mobile devices, and data centers. We evaluate the performance of this strategy by developing a prototype infrastructure using Apache Kafka, an open source message delivery system, and compared its performance with the latency requirements of CV applications. We present experimental results of the message delivery infrastructure on two different distributed computing testbeds at Clemson University: the Holocron cluster and the Palmetto cluster. Experiments were performed to measure the latency of the message delivery system for a variety of testing scenarios. These experiments reveal that measured latencies are less than the U.S. Department of Transportation recommended latency requirements for CV applications, which prove the efficacy of the system for CV related data distribution and management tasks.
ACKNOWLEDGEMENT

This study is based upon research supported by the U.S. Department of Transportation’s Southeastern Transportation Center under O/E Grant 2016-2017.
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISCLAIMER</td>
<td>ii</td>
</tr>
<tr>
<td>ACKNOWLEDGEMENT</td>
<td>iv</td>
</tr>
<tr>
<td>TABLE OF CONTENTS</td>
<td>v</td>
</tr>
<tr>
<td>EXECUTIVE SUMMARY</td>
<td>1</td>
</tr>
<tr>
<td>DESCRIPTION OF PROBLEM</td>
<td>3</td>
</tr>
<tr>
<td>RELATED WORK</td>
<td>7</td>
</tr>
<tr>
<td>Connected Vehicle Systems Deployment</td>
<td>7</td>
</tr>
<tr>
<td>Message Delivery System for Connected Vehicle Applications</td>
<td>8</td>
</tr>
<tr>
<td>APPROACH AND METHODOLOGY</td>
<td>11</td>
</tr>
<tr>
<td>Metadata Analysis</td>
<td>11</td>
</tr>
<tr>
<td>Distributed Message Delivery System for Connected Vehicle Applications</td>
<td>15</td>
</tr>
<tr>
<td>CASE STUDY: EVALUATION OF DISTRIBUTED MESSAGE DELIVERY SYSTEM</td>
<td>18</td>
</tr>
<tr>
<td>Synthetic Data Description</td>
<td>18</td>
</tr>
<tr>
<td>Infrastructure Experimental Platform Description</td>
<td>20</td>
</tr>
<tr>
<td>Experimental Scenarios</td>
<td>21</td>
</tr>
<tr>
<td>Analysis</td>
<td>24</td>
</tr>
<tr>
<td>CONCLUSIONS</td>
<td>34</td>
</tr>
<tr>
<td>REFERENCES</td>
<td>35</td>
</tr>
<tr>
<td>APPENDIX</td>
<td>39</td>
</tr>
</tbody>
</table>
EXECUTIVE SUMMARY

The real-world deployment of connected vehicle (CV) technologies, promoted in the United States and some other countries, entails developing roadway transportation systems that involve wireless interconnections between vehicles, and transportation and information infrastructures to provide safe and reliable transportation. The American Association of State Highway and Transportation Officials (AASHTO) footprint analysis report identifies three elements that are necessary for such a dynamic and complex connected transportation system. Specifically, the system must 1) improve safety through vehicle-to-vehicle communication; 2) improve traffic operations by providing travel time information to drivers, transit riders, and freight managers in real-time; and 3) reduce the adverse environmental effect using real-time information to enhance fuel efficiency.

Rather than being a separate environment, CV systems will become part of a bigger connected society, such as city wide smart disaster management systems that include smart traffic management, smart power grid, and smart healthcare, as described in the vision for Smart Networked Systems. An explosion of CV data is expected when one considers how much data can be collected continuously via a large number of sensors (e.g., sensors in vehicles, cell phones, roadside units). These data will come in varied formats (e.g., PDF, CSV, and structured/unstructured XML). In addition, different types of applications and the numbers of users requiring specific subsets of CV data from different sources will increase significantly as the market penetration of CVs in the roadway traffic stream increases over time.

A complex and vast amount of data will be collected from on-board sensors of operational connected vehicles (CVs), infrastructure data sources such as roadway sensors and traffic signals, mobile data sources such as cell phones, social media sources such as Twitter, and news and weather data services. Unfortunately, these data will create a bottleneck at data centers for processing and retrievals of collected data, and will require the deployment of additional message transfer infrastructure between data producers and consumers to support diverse CV applications. In this paper, we present a strategy for creating an efficient and low-

---

latent distributed message delivery system for CV applications using a distributed message delivery platform. This strategy enables large-scale ingestion, curation, and transformation of unstructured data (roadway traffic-related and roadway non-traffic-related data) into labeled and customized topics for a large number of subscribers or consumers, such as CVs, mobile devices, and data centers. We evaluate the performance of this strategy by developing a prototype infrastructure using Apache Kafka, an open source message delivery system, and compared its performance with the latency requirements of CV applications. We present experimental results of the message delivery infrastructure on two different distributed computing testbeds at Clemson University: the Holocron cluster and the Palmetto cluster. Experiments were performed to measure the latency of the message delivery system for a variety of testing scenarios. These experiments reveal that measured latencies are less than the U.S. Department of Transportation recommended latency requirements for CV applications, which prove the efficacy of the system for CV related data distribution and management tasks.
DESCRIPTION OF PROBLEM

The real-world deployment of connected vehicle (CV) technologies, promoted in the United States and some other countries, entails developing roadway transportation systems that involve wireless interconnections between vehicles, and transportation and information infrastructures to provide safe and reliable transportation [1]. The American Association of State Highway and Transportation Officials (AASHTO) footprint analysis report identifies three elements that are necessary for such a dynamic and complex connected transportation system. Specifically, the system must 1) improve safety through vehicle-to-vehicle communication; 2) improve traffic operations by providing travel time information to drivers, transit riders, and freight managers in real-time; and 3) reduce the adverse environmental effect using real-time information to enhance fuel efficiency [2].

Rather than being a separate environment, CV systems will become part of a bigger connected society, such as city wide smart disaster management systems that include smart traffic management, smart power grid, and smart healthcare, as described in the vision for Smart Networked Systems [3]. An explosion of CV data is expected when one considers how much data can be collected continuously via a large number of sensors (e.g., sensors in vehicles, cell phones, roadside units). These data will come in varied formats (e.g., PDF, CSV, and structured/unstructured XML) [4-6]. In addition, different types of applications and the numbers of users requiring specific subsets of CV data from different sources will increase significantly as the market penetration of CVs in the roadway traffic stream increases over time.

The primary challenges for a CV message delivery system involve redistribution of data at various stages of the data lifecycles (i.e., raw, integrated, and processed) while meeting specific functional requirements based on time and spatial contexts. These requirements and contexts can vary depending on which CV applications being considered. For example, traffic management applications alone can generate multiple streams of data from various sensors of thousands of CVs. These data must be correlated and analyzed in real-time to identify traffic conditions and to provide further operational actions. This analysis will then make it possible to establish a hierarchy of various operational actions to ensure proper traffic condition
assessments and selection of management decisions. These challenges also lead to some potential issues, which include: 1) increased latency for delivery of data in an usable format from the raw data as per CV applications that may require specific time-sensitive data; 2) requirements for more data storage at transportation data processing centers; and 3) failure of data processing machines at specific transportation centers that could create a risk of large scale failures in the CV ecosystem. In a previous work, Lantz et al. presented a framework for the data infrastructure that can support the handling of massive volume of data in urban transit networks [7]. This study discussed that a high rate of data arrival and data access depend on the capability and capacity of the data infrastructure.

The primary objective of this study is to design a distributed message delivery infrastructure for connected vehicle systems. A key design assumption for this infrastructure is that the standard data processing centers (data warehouses), such as those managed by public agencies, are the core locations from which data is delivered to support hundreds of CV applications. In our design, the raw data are separated, tagged, and posted by a software component, which is a distributed message delivery platform. In this strategy, whichever entity requests data will contact the platform to identify the tags of interest and receive the relevant data via the platform. The conceptual vision for this infrastructure is illustrated in Fig. 1, where different roadway traffic (such as vehicles and centers) and non-traffic (news and weather) related entities are serving as both data producers and consumers to support CV and other transportation related applications. The addition of the message delivery system enables a data-focused view of the entire CV ecosystem in which any single entity can deliver data as well as acquire data.
As there is no fully functional (non-experimental) connected vehicle system to support the study of message delivery behavior, we rely on a study of Connected Vehicle Reference Implementation Architecture (CVRIA) metadata in order to estimate the potential demand of data for different CV applications [8]. More specifically, we are interested in the temporal distribution of CV data, which is very critical for modeling the various CV application scenarios and types of data flow, as the message delivery systems must maintain acceptable latency for CV applications. In this paper, we develop and evaluate the performance of a prototype infrastructure of a distributed message delivery system, which could support future...
CV systems and applications, using an open source distributed streaming platform, Apache Kafka [9].

The structure of the report is as follows. Related work section entails a discussion of the related studies on the current status of connected vehicle systems and existing message delivery systems. Approach and methodology section presents an analysis of CVRIA metadata to characterize the evolution of the data types in CV applications. Case study section describes an infrastructure for a distributed message delivery system for CV applications. This section also presents a detail evaluation of the distributed message delivery system and performance evaluation of this design in connected vehicle systems. Finally, conclusions section includes a concluding discussion.
RELATED WORK

A review of previous studies related to the evaluation of message delivery systems for connected vehicles is discussed in the following two sub-sections. The first sub-section reviews the current status of connected vehicle systems deployment. The second subsection explores existing message delivery systems for connected vehicle applications.

Connected Vehicle Systems Deployment

Connected vehicle systems demand an integrated deployment plan both from private sectors, such as automobile manufacturers, and public sectors, such as U.S. Department of Transportation (USDOT) in the US. Several large-scale pilot deployments, sponsored by the USDOT, are underway to develop market ready technologies before mandating Vehicle-to-Vehicle (V2V) technologies for new vehicle models in the next few years [10]. The Southeast Michigan Test Bed is one such federally funded publically available center for analyzing CV technologies [10]. This test bed provides a real-world laboratory to evaluate CV applications. In 2012, two CV test beds were deployed in Virginia for connected vehicle research [11]. The first is located in Blacksburg, VA, at the Virginia Smart Road and along Route 460, in the southwestern part of the state. The second is located in Fairfax County along I-66 and the parallel Routes 29 and 50, which are in the northeastern part of the state. The Connected Vehicle Infrastructure University Transportation Center (CVI-UTC) is using both sites to develop a fully operational test bed for connected vehicle mobility applications, and the Fairfax site will be used for dynamic alternate route research [11]. In 2015, Florida, California, and New York were also selected as new sites for similar test beds [12]. While the majority of operating test beds are used to analyze CV application development and evaluation, there is no test bed for data infrastructure evaluation for real-world CV applications.

The Michigan connected vehicle testbed Proof of Concept (POC) test reports identified that latency to deliver a message from a vehicle to the application server (i.e., from a connected vehicle to a roadside unit (RSU)) is between 0.5 and 1.5 seconds for CV application [13]-[14]. This time range depends on the technologies for wireless communication and backhaul data transfer, and data congestion in the communication network. As the POC test used a limited
number of connected vehicles, it was not possible to evaluate latency in the presence of network congestion. Using IntelliDrive probe vehicle data to analyze CV system performance, Dion et al. observed an average 65s latency when Vehicle-RSU protocols permitted the sequential upload of messages, while an average latency of 30 seconds was observed when vehicles were allowed to immediately transmit all messages simultaneously to RSU [14]. In a connected vehicle system, multiple CV applications could run simultaneously with a massive amount of data sent and received between vehicles and RSUs. This would increase resource contention and latency significantly, which would eventually lead to data loss if the data infrastructure design cannot handle a massive amount of data. Furthermore, no study evaluated message delivery systems between RSU and data centers of different transportation centers.

**Message Delivery System for Connected Vehicle Applications**

Although a significant body of work is available on message delivery among CVs and RSUs via vehicle ad hoc networks (VANETs), the focus of those studies was on direct vehicle-to-vehicle (V2V) and vehicle-to-infrastructure (V2I) communication that use direct and localized wireless communication technologies [15–17]. The message delivery system using VANETs lacks support for aggregation and processing of raw data from multiple sources, including those that are not from CVs and RSUs. This can limit the applicability of VANETs to CVRIA’s applications [8], where sources, time context, and spatial context of input data can vary. As a result, there is a need to create a middleware layer that can manage these distributed data streams independently.

Such middleware components, often described as distributed message delivery frameworks, are integral to cyberinfrastructure in various areas, including social media, manufacturing, and e-commerce [18–24]. These frameworks often consist of a cluster of interconnected computers that rapidly ingest streaming messages generated by data *producers* (e.g., sensors, roadside units). Different frameworks distribute the data to *consumers* (e.g., traffic management centers, mobile applications) via push (i.e., the data are pushed to the consumers) or pull (i.e., the consumers pull data from the frameworks’ storage space) models. Each computer can host one or more *brokers*. The ingestion and distribution processes are distributed among the message brokers to ensure a parallel operation of both processes. There
exist a number of such frameworks, including RabbitMQ [25], ActiveMQ [18], WebsphereMQ [19], and MQTT [26]. Of these, RabbitMQ is one of the most popular open source solutions and is widely used in academic and industry settings [27].

RabbitMQ [27]-[28] uses volatile storage as the primary means for storing incoming data, with messages collected and stored in memory using abstract data structure named message queue. The queues reside in memory and allow consumers to pull messages from them directly. Since the queues in memory are limited in space, messages not acquired by consumers are moved to a secondary storage level on the disk prior to being purged. Such purging can increase in data retrieval time and data loss due to message delivery latency [28-30]. This is a concern for CVRIA applications, as message delivery latency can increase when there is an increased number of consumers during peak hours or more data are being streamed to brokers in critical events. This characteristic of RabbitMQ could negatively affect some critical CV applications (e.g., cooperative adaptive cruise control) that are sensitive to data accuracy and data loss.

The rapid development of data-driven infrastructures over the last decade has led to the creation of another distributed message delivery system implementation, Apache Kafka [9]. Apache Kafka was originally used as a distributed aggregation framework for massive amount of system log messages. Apache Kafka is highly scalable and fault tolerant, and it uses the concept of topics to represent data streams. Topics are viewed as data files stored in persistent storages during message delivery. A topic can be divided into several partitions that are located across different brokers, enabling both streaming data sources and data consumers to produce and consume messages in parallel [27]. This leads to Kafka’s capability to deliver large volume of messages simultaneously [31]-[32]. A performance evaluation of Kafka shows that it outperforms similar message delivery alternatives in terms of messages delivered per second by both data-generating and data-requesting entities [9]. Thus, Kafka appears more suitable for CV applications that need high throughput and persistent message delivery. Given that there has been no previous study on the performance of distributed message delivery system based on Kafka for CV systems, our paper studies this middleware for designing distributed message delivery system. There is a difference between distributed message delivery systems (e.g., Apache Kafka) and distributed stream processing systems. As the names suggest, the former
focuses on the delivery of messages, and the latter focuses on the processing of streaming messages.

Our study focuses on a message delivery infrastructure, based on Apache Kafka, for the distributed delivery of streaming messages. The streaming message delivery infrastructure creates a preliminary data acquisition and ingestion in which stream processing and batch processing tasked with various service level agreement (SLA) can extract the relevant message streams based on their own capacity and run-time requirements. Distributed message delivery systems and distributed stream processing systems are used to support large-scale data infrastructure for interactive data analytics [33]. Examples of distributed stream processing systems include the work by Biem et al. in which they developed a data stream-processing infrastructure that supports the analytical tasks for real-time streaming data [34] and other work on advanced streaming platforms, such as, Spark [35], Flink [36], and Storm [37].

More recently, a new area of research in content-centric network paradigm in message delivery system has been explored with the focus on replacing the bottleneck created when users’ requested contents are associated with specific hosting locations via the TCP/IP networking protocol. The use of a content-centric networking paradigm makes it possible to decouple the location of the contents from a data user’s identity. This improves network security and data access for both data users and data providers [38]. Preliminary work in a content-centric network has shown to be applicable for data collection and dissemination at ranges longer than the standard VANET [38]-[39]. The message delivery concept of our distributed message delivery infrastructure is similar to the content-centric networking, where the message streams are made available and accessible to users via the streams’ content tags only, and the users are not required to know to the origin of the message streams.
APPRAOCH AND METHODOLOGY

Metadata Analysis

Key in the use of Kafka is the identification of the topics that will represent the CV data streams. To identify the topics for our case study, we use CVRIA [8], which has been developed by the USDOT and which forms the basis for a common language definitions and early CV deployment concepts. CVRIA identifies key interfaces across different entities/stakeholders in a connected vehicle system and supports standard development activities. The CVRIA architecture is composed of four viewpoints: Physical, Functional, Communications and Enterprise. The physical view represents the connection between physical and application objects, whereas the functional view shows the logical interconnections between functions. The enterprise view represents the relationship among the organizations responsible for their respective roles in a CV environment. Finally, the communications view organizes the different communication protocols that enable communication between application objects. CVRIA has defined the concept of operations for ninety-five CV applications (at the time of writing this paper), and categorized as: 1) environmental, 2) mobility, 3) safety and 4) support [8]. For each application, a physical architecture was developed that identified the physical objects/centers, application objects/equipment packages and information flows between application objects to support each application’s functional requirements. Each information flow of a connected vehicle application includes the following two contexts that are related to this study: 1) spatial and 2) time contexts. The spatial context is classified into 1) adjacent (0 m – 300 m), 2) local (300 m-3 Km), 3) regional (3 Km – 30 Km), 4) national (30 Km - National) and 5) continental (Continental US) categories [8]. The time context is classified into 1) now (less than 1 second), 2) recent (1 second – 30 minutes), 3) historical (30 minutes - 1 month) and 4) static (greater than 1 month) categories [8].

As each connected vehicle application has a specific data characteristic in terms of spatial and time contexts, we have analyzed metadata of different CV applications to understand the real-time service requirements in terms of spatial and time contexts, which provides the basis of designing distributed message delivery system to support different CV applications. For example, in the Speed Harmonization application, the ‘variable speed limit
status’ information flows from the traffic management center to Intelligent Transportation Systems (ITS) roadway equipment. The characteristics of this information flow are local in ‘spatial context’ and recent in ‘time context’. Consequently, it is challenging to deliver data concurrently for different CV applications. An analysis of CVRIA information flow is useful to support the design of data analytics infrastructure for connected vehicle systems. The number of unique information flow can be defined as the total number of information flows required for all the CVRIA applications without any redundancy. As the same information flow can be used by different CV applications, aggregation of all information flow is required to reduce redundant information flows. For example, ‘traffic flow’ is a unique information flow, which contains traffic flow variables (e.g., speed, volume, and density measures). The source of this information flow is ‘ITS Roadway Equipment’ and destination is ‘Traffic Management Center’. This information flow is required in the following CV applications: 1) Cooperative Adaptive Cruise Control; 2) Eco-Cooperative Adaptive Cruise Control; 3) Eco-Lanes Management; 4) Eco-Ramp Metering; 5) Eco-Speed Harmonization; 6) Eco-Traffic Signal Timing; 7) Intelligent Traffic Signal System; 8) Intermittent Bus Lanes; 9) Low Emissions Zone Management; 10) Queue Warning; 11) Roadside Lighting; 12) Speed Harmonization, and 13) Variable Speed Limits for Weather-Responsive Traffic Management. To avoid duplication of information flows, we count this information flow as a single flow for all CVRIA applications. To determine the number of unique information flows, we collected and stored all information flows related to 95 CV applications identified so far (at the time of writing this paper) in CVRIA. We then wrote a python script for analyzing all the information flows to determine the number of unique information flows based on the time and spatial contexts. All unique information flows (shown in Fig. 2) that originated from different centers (e.g., Traffic Management Center) are coded as ‘from center’; and all unique information flows that are received by the different centers are coded as ‘to center’. All of the unique information flows in ‘to center’ and ‘from center’ categories are classified for spatial context (i.e., A) adjacent, B) local, C) regional, D) national, and E) continental), and time context (i.e., 1) now, 2) recent, 3) historical, and 4) static) and presented in Figs. 3 and 4, respectively. Fig. 2 shows the comparison between the frequency of unique information flows to the centers and from the centers for all the CV applications in CVRIA. CVRIA identified 25 centers that are producing
data, and 27 centers, which are receiving information. This figure also shows the number of unique information flows that need to be aggregated for each center. There are a total of 231 information flows received by all the centers with a total of 219 information flows sent from all centers. The Traffic Management Center receives and sends the highest total number of information flows. Fig. 3 presents the distribution of information flows, based on time and spatial contexts, to the centers from other physical objects. Information flows are also classified in different combinations of spatial and time contexts categories for all CV applications. We observed that most of the information flows are in recent and local (2B), and recent and regional (2C) categories. Fig. 4 presents distribution of information flow, based on time and spatial contexts, from the centers to other physical objects. We observed similar distribution in ‘to center’ information flows and most of the information flows are in 2B and 2C categories. The identification of unique information flows will reduce the total data volume by eliminating the redundancy of the information flow in a distributed message delivery system design.

![Graph showing frequency of information flow]

**Figure. 2. Frequency of information flow by “from center” and “to center”**.
Figure 3. Distribution of information flow based on time and spatial contexts “to center” from other physical objects.

Figure 4. Distribution of information flow based on time and spatial contexts “from center” to other physical objects.
Distributed Message Delivery System for Connected Vehicle Applications

As connected vehicle systems become more integrated with a smart and connected society through new technologies, such as, automated vehicles and large-scale sensor networks, the demand for access to data in a connected vehicle system will also increase rapidly.

Figure. 5. Conceptual design of distributed data-focused delivery infrastructure from RSU to different transportation centers.

Therefore, it is beneficial to shift from an application-focused view to a data-focused view. In a data-focused view, different transportation centers can provide messages that are tagged (i.e., labeled by their corresponding centers), instead of having to support messages that fit the requirements of applications. Meanwhile, the applications are responsible for initializing the communication with the brokers for the message acquisition process. This is achievable by creating a Kafka layer to manage the various tagged types of data. The other two types of entities within this infrastructure are producers, which create messages, and consumers, which acquire messages from brokers. The flow of data in a distributed message delivery infrastructure from RSU to center is illustrated in Fig. 5. Raw data are first streamed from the
producers to the brokers, where data are placed into queues. Each queue is labeled by a “topic tag”. The consumers subscribe to the relevant individual queues in order to retrieve the required messages from the brokers. The brokers, depending on the needs of the consumers, dynamically create the queues. All entities within a connected vehicle system (i.e. vehicles, traffic management centers, online news, weather, social media sites, and even the CV applications themselves) can be producers or consumers of messages. By using the Kafka brokers as a medium to facilitate data streaming, a distributed message delivery infrastructure can accomplish the following:

- **Separation of content and location**: Kafka brokers enable consumers to stream relevant data from producers without either party (consumer or producer) having to know about each other’s location. For example, an emergency management application can ingest a message packet tagged as a *roadway crash message* without first establishing contact with the source of the message package. In this case, the emergency management application is the message consumer and the *roadway crash message* source is the message producer. The message consumer and the message producer only talks directly to the brokers while their locations remain hidden with respect to each other.

- **Optimization of data management and processing through the broker layer**: The computing capability of the Kafka broker cluster enables preliminary curation of raw data before placing them into queues. By placing this responsibility into the Kafka middleware, the traffic management centers can focus more on analyzing the stored message rather than cleaning the raw data and preparing message into a usable format.

- **Dynamic balancing and scaling of message delivery infrastructure**: As CV applications have different time requirements for data arrival [40]-[41], this requires the applications to only consume data at an appropriate rate. Apache Kafka supports the dynamic addition and removal of any broker in an existing cluster, allowing the infrastructure to scale up during peak demand hours and scale down during periods of reduced demand [9].

- **Reduction of administrative and technical responsibilities for data maintenance**: In a traditional approach, message is usually accessed directly from the data centers of public Departments of Transportation (DOTs) or similar public agencies. A high level of message redundancy is required not only for backup purposes but also to support large-scale
message access (i.e., creating multiple copies of message to be accessed by a large number of users and applications). By placing the burden of facilitating message delivery on a brokerage layer, the DOT centers only need to focus on maintaining data for traditional in-house transportation applications. External applications that require message can have access to the message via the streams of the broker layer. While it is possible to duplicate these streams for higher data availability consideration, the real-time nature of these streams will prevent the accumulation of message storage duplication
CASE STUDY: EVALUATION OF DISTRIBUTED MESSAGE DELIVERY SYSTEM

We designed a prototype infrastructure of the distributed message delivery system using Kafka for CV applications. Synthetic data, which were generated using the VISSIM microscopic traffic simulator, were used to evaluate the prototype infrastructure. In the following subsections, we describe the details of the case study.

Synthetic Data Description

Microscopic traffic simulation is an efficient and economical solution for generating synthetic data to evaluate connected vehicle systems of any regional roadway network. We can model a connected vehicle system in a simulated roadway network representing drivers, vehicles, and roadways and perform experiments with different connected vehicle system architectures supporting different applications. Synthetic data generated from simulation experiments include input and output data to and from each subsystem (i.e., consumers, producers), correspondingly, within the respective message delivery architectures. For example, the vehicle position and velocity messages are transmitted from the vehicle on-board equipment (OBE) to RSU whereas RSU transmits recommended speed information to the vehicle OBEs for real-time speed management of CVs. In this case study, we simulated a roadway network along the I-26 corridor between Columbia and Charleston in South Carolina, which encompassed 91.5 miles of freeway with 19 interchanges. VISSIM (a microscopic traffic simulator) developed by PTV [42] was used to develop a simulated model of the I-26 corridor. According to the Cost Overview for Planning Ideas & Logical Organization Tool (CO-PILOT), a high-level planning tool for connected vehicle pilot deployments, we assume one RSU installation per mile on I-26 corridor [43]. There is an exception to place RSUs at the horizontal curves and the assumption is that two RSUs are required for each curve, one at each end of the curve. For the I-26 corridor, DSRC communication range of 900 ft was used for all the 92 RSUs [44]. Thus, each RSU can collect data, if a vehicle is within RSU coverage (i.e., within the 900 ft radius). In our simulation, each RSU collected CV data for a highway segment of 1800 ft considering 900 ft DSRC coverage in both directions of traffic.
From the VISSIM simulation, we recorded 62 types of data (e.g., speed, position, acceleration, lane number, lane change) for each CV within the DSRC communication range. The CV data are collected using “Vehicle Record” output option from the VISSIM simulation. We then use these data in our experiments to evaluate the distributed message delivery infrastructure prototype that models RSUs connected to different transportation centers (e.g., traffic management center, transit management center, emergency management center, commercial vehicle administration center, transportation information center). In our connected vehicle system, each CV sends data to RSU, and each RSU sends CV data to the backend transportation centers. These CV data can support different types of CV applications, such as vehicle-to-infrastructure safety (e.g., curve speed warning), mobility (e.g., speed harmonization), and environmental (e.g., variable speed limits for weather-responsive traffic management) applications as depicted in CVRIA [8]. As shown in Figs. 3 and 4, different type of information flows are sent from each RSU to different transportation centers for different CV applications. For this study, we consider only data generated from each CV and was sent to RSU; and data sent from RSU to different transportation centers. In the following, we list all 62 data types collected from the VISSIM simulation. However, our distributed message delivery system can also support other information flows (e.g., roadway sensors data) from other sources (such as roadway traffic sensors) along with vehicle-generated data. In our case study presented in the paper, other sources of traffic related data, such as traditional traffic sensors (e.g., loop detector) and mobile device (cell phone or GPS traces) data were not considered. It is expected that the substantial penetration of CVs on the roadways will reduce or eliminate the need for traditional traffic sensor data for traffic condition assessment and prediction [45]-[46].
**List of CV Data Types Collected from VISSIM Simulation**

Time stamp; date; vehicle ID; vehicle length; vehicle type; vehicle weight; vehicle name; vehicle power; fuel consumption; speed; average speed; speed difference; desired speed; theoretical speed; acceleration; safety distance; headway; desired direction; desired lane; destination lane; lane change; lane number; target lane; link number; destination link; lateral position relative to middle of the lane; leading vehicle; preceding vehicle; number of stops; occupancy; world coordinate front x; world coordinate front y; world coordinate front z; world coordinate rear x; world coordinate rear y; world coordinate rear z; route number; routing sequence; trip chain: activity; trip chain: departure time; trip chain: destination zone; trip chain: minimum duration; trip chain: parking lot number; emissions (evaporation) Hydro Carbon (HC); emissions benzene; emissions CO; emissions CO2; emissions Hydro Carbon (HC); emissions Non-Methane Hydrocarbon (NMHC); emissions Non-methane Organic Gas (NMOG); emissions NOx; emissions particulates; traffic interaction state; total distance traveled; total time in a network; delay time with respect to optimal drive time; destination parking lot; following distance; gradient; total number of queue encounters; queue flag; and queue time.

**Infrastructure Experimental Platform Description**

Our experimental platform spans two computing platforms based at Clemson University [47]. The first, the Holocron cluster, is a small-scale platform that consists of 20 machines provided to researchers in bare-metal fashion, enabling the installation of any customized software for experimentation with customized cloud infrastructures. In our experiments, we selected 16 machines from Holocron as our Kafka brokers. Each machine has 256 GB DDR4 RAM memories, 2TB 7,200 RPM SATA HDD hard disk and a 10Gbps Ethernet connection for data transfer between machines. This 16-machine broker cluster is the key component for the message delivery system, which comprises Kafka data brokers and supporting services [9]. Therefore, most of the computing resources in the message delivery system reside in this cluster. The brokers handle the incoming message traffic from message producers as well as route outgoing message traffic to message consumers.
The second platform, Palmetto, is a large-scale distributed testbed for scientific research applications consisting of 1,700 machines maintained by the Clemson University Cyberinfrastructure Technology Integration (CITI) group [47]. Since it provides a large number of physical machines, we used machines from the Palmetto cluster as both our data producers and consumers for CV applications. As the Palmetto cluster has over 1000 of the available 1700 machines for scientific research, we allocate a single physical machine from Palmetto for each producer or consumer. This guaranteed better resource isolation, thus ensuring the repeatability and reliability of our experiments. Each Palmetto machine functioned as either a producer or a consumer of the message delivery system by running a specific computer program. For instance, a producer machine runs the data generation program, representing the RSU in the connected vehicle applications. On the other hand, a consumer machine runs the data-ingesting program, representing a traffic management center or a mobile device, which runs certain CV applications that consume data from brokers. Each of the Palmetto machines had 16GB of DDR4 RAM and 100GB 7,200 RPM hard drive and 1Gbps Ethernet connection to the Holocron nodes. The distributed message delivery system for CV applications was then prototyped by setting up a cloud infrastructure between the producers and consumers in Palmetto and the brokers in Holocron. Holocron provides flexible networking among distributed nodes, which facilitated our experiments by assigning the same network configurations on each producer to broker link and each broker to consumer link. Palmetto’s integrated batch system, Portable Batch System (PBS) [48], eases the parallel execution of large number of producer and consumer computer programs, which is essential in modeling the case where all consumers are “online”.

**Experimental Scenarios**

To evaluate the performance of the distributed message delivery system, a baseline scenario of message distribution system without any message broker was established. We considered 92 producers in this baseline scenario, all of which were connected with 10 consumers sequentially (as shown in Table I) using a direct Transmission Control Protocol (TCP) connection. We then developed four experimental scenarios to evaluate the distributed message delivery system with different number of brokers and consumers, as presented in
Table I. The number of the producers was fixed at 92 (i.e., number of RSUs installed in I-26 corridor). As various numbers of end users affect CV applications during different hours of the day, we varied the number of consumers for each broker size to 10, 20, 30, 40 and 50 to represent different times of the day. The 50-consumer scenario represents the peak hour scenario when most consumers attempt to access data simultaneously, while the 10-consumer scenario represents the off-peak hours when most users are offline. We use different number schemes for our brokers (e.g., 2, 4, 16, 32) to explore how the larger broker clusters benefit the performance of the message delivery system. We allocated the 32 brokers at 16 physical machines in Holocron. Two broker instances were managed simultaneously on each machine.

<table>
<thead>
<tr>
<th>TABLE I</th>
<th>Message Delivery System Evaluation Scenarios</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Message delivery System without Broker</strong> (Baseline Scenario)</td>
<td></td>
</tr>
<tr>
<td>Number of Producer</td>
<td>Number of Consumer</td>
</tr>
<tr>
<td>92</td>
<td>10</td>
</tr>
<tr>
<td><strong>Message delivery System with Broker</strong></td>
<td></td>
</tr>
<tr>
<td>Experimental Scenario</td>
<td>Number of Producer</td>
</tr>
<tr>
<td>Scenario 1</td>
<td>92</td>
</tr>
<tr>
<td>Scenario 2</td>
<td>92</td>
</tr>
<tr>
<td>Scenario 3</td>
<td>92</td>
</tr>
<tr>
<td>Scenario 4</td>
<td>92</td>
</tr>
</tbody>
</table>

We examined the message delivery system experimentally considering the capacity of all producers. For this purpose, each of the 92 producers published 100,000 messages at the same time to overflow the network capacity. These messages were then published to 62 different topics, each representing a single message type as discussed in our Synthetic Data Description sub-section (Sub-section A, Section V). All the messages have a fixed size of 200 bytes. Here, a topic is an abstract container in Kafka that can be seen as a pipeline in which messages flow.
through. The configuration of each topic in the broker cluster reflects the trade-off between message reliability and latency for all the messages published to that topic. For example, in extreme cases where we generated messages from a certain number of producers to arrive at the brokers at its maximum speed, messages were sent to a topic in the brokers that did not require any acknowledgements sent to producers on each message. All of those messages published to that topic thus exhibited the lowest latency.

In our experiment with a 4-node Kafka broker cluster, the average latency for sending one message dropped by 78 percent when switching from a topic, whose messages require acknowledgement ("ack" topic), to a topic whose messages that do not require acknowledgement ("no-ack" topic). However, the reliability of the "no-ack" topic is problematic. Over 98 percent of all the messages published to the "no-ack" topic is received by the consumers in the 4-broker cluster scenario, which makes this topic unsuitable for the applications that require a highly reliable message transfer (i.e., where 100 percent of messages must be received by the consumers). Since we value the message reliability for our real-time CV applications, we configured all 62 topics in our experiment to be "ack" topics, which required explicit acknowledgement from the broker for each message. The end-to-end latency of a message passing through the message delivery system is comprised of two parts. The first part of this latency included the time of communication process between the producer and the broker, represented by Tp as shown in Fig. 6. This Tp was measured from the time at which the message was generated by the producer to that time at which the broker indicated reception of the message.

![Diagram](https://via.placeholder.com/150)

**Figure. 6. Breakdown of an end-to-end latency of a message passing using the developed message delivery system.**
The second part of the latency encompassed the time of communication process between the broker and the consumer, which is denoted by $T_c$ (as shown in Fig. 6). Here, $T_c$ was measured from the time at which the consumer initialized the message-read request to the time at which the consumer successfully received the message. The parallel implementation of broker to consumer message delivery in Kafka makes the latency $T_c$ much lower than $T_p$. Therefore, at the producer side, it was necessary to reduce the per-message delivery latency to the highest extent to better serve the real-time applications. This latency is associated with the queuing latency of each message in the producer’s send buffer. The average queuing time of each message at the producer end was proportional to the quotient of buffer size and message size. If the buffer size was too large, the average queuing time of each message was increased. To prevent this queuing delay, we assigned the send buffer size of each of our producers to 400 bytes, twice of the message size. Therefore, whenever a message was loaded into the send buffer, it only had to wait for dispatch of its only predecessor. The small size of the message prevented any overflow in the producer. This approach eliminated the unnecessary queuing delay at the producer end. A total volume of 40 GB of messages was generated in the experiment (2 GB for each experimental scenario).

**Analysis**

For the baseline scenario, we considered 92 producers (i.e., RSUs), where each producer was sending 50,000 messages to 10 consumers sequentially using a direct TCP connection. Each consumer opened a TCP socket and listened for connections from the producers, and each producer made a socket connection for every message prior to transmission. Each message was 200 bytes in size with 62 types of CV data (see Sub-section A, Section V) to simulate the baseline scenario and distributed message delivery system. We used two Kafka brokers in our comparison of the baseline scenario, which is the minimum system requirement to avoid single point machine failure, for the distributed message delivery system. The average end-to-end and the maximum end-to-end message delivery latency for the baseline scenario (without any brokers) were 12 milliseconds and 3751 milliseconds, respectively, which were significantly greater than the corresponding latencies with a distributed message delivery system using two Kafka brokers (as shown in Table II). This difference is mainly due to the smaller overhead of
the distributed message delivery system when serving multiple consumers compared to the baseline scenario. For serving multiple consumers, the Kafka distributed message delivery system handles all incoming messages in parallel, where in the baseline scenario only one message is handled at a time between a producer and a consumer.

**TABLE II**

Comparison of the Message Delivery Latency Between the Baseline and Distributed Message Delivery System

<table>
<thead>
<tr>
<th>Experimental Scenario (92 producers and 10 consumers)</th>
<th>Total Average Latency (ms)</th>
<th>Maximum Latency (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline (without broker)</td>
<td>12.00</td>
<td>3751</td>
</tr>
<tr>
<td>Distributed message delivery system with broker (2 brokers)</td>
<td>2.03</td>
<td>1463</td>
</tr>
</tbody>
</table>

The broker’s cached message pool then processes multiple message deliveries to different consumers simultaneously. Our analyses for distributed message delivery system with four different experimental scenarios are presented in Table III and Fig. 7, 8, 9 and 10. In Table III, we presented average Tp and Tc in addition to total average latency for different experimental scenarios. Note the smaller Tc compared to Tp, because of the efficient implementation of the message delivery function in Apache Kafka. When a consumer needs to read messages from the brokers, the Kafka brokers use the Linux sendfile Application Programming Interface (API) to transfer messages to the consumer socket without further buffering or copying the messages. This reduces the transmission latency, and therefore affects Tc. Moreover, when consumers simultaneously read a large message set, Kafka batched the messages in groups to leverage the receiving buffer at the consumer end in order to improve the latency performance.
TABLE III
Tp and Tc For Distributed Message Delivery System in Different Experimental Scenarios

<table>
<thead>
<tr>
<th>Number of Consumers</th>
<th>Average Tp (ms)</th>
<th>Average Tc (ms)</th>
<th>Total Average Latency (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>Experimental Scenario 1:</strong> Number of Producers =92 and Number of Brokers = 2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>1.55</td>
<td>0.48</td>
<td>2.03</td>
</tr>
<tr>
<td>20</td>
<td>3.19</td>
<td>0.83</td>
<td>4.02</td>
</tr>
<tr>
<td>30</td>
<td>4.23</td>
<td>1.06</td>
<td>5.29</td>
</tr>
<tr>
<td>40</td>
<td>5.42</td>
<td>1.34</td>
<td>6.76</td>
</tr>
<tr>
<td>50</td>
<td>5.97</td>
<td>1.98</td>
<td>7.95</td>
</tr>
<tr>
<td></td>
<td><strong>Experimental Scenario 2:</strong> Number of Producers =92 and Number of Brokers = 4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>1.36</td>
<td>0.31</td>
<td>1.67</td>
</tr>
<tr>
<td>20</td>
<td>2.23</td>
<td>0.49</td>
<td>2.72</td>
</tr>
<tr>
<td>30</td>
<td>3.20</td>
<td>0.72</td>
<td>3.92</td>
</tr>
<tr>
<td>40</td>
<td>3.97</td>
<td>0.89</td>
<td>4.86</td>
</tr>
<tr>
<td>50</td>
<td>5.02</td>
<td>1.16</td>
<td>6.18</td>
</tr>
<tr>
<td></td>
<td><strong>Experimental Scenario 3:</strong> Number of Producers =92 and Number of Brokers = 16</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>1.21</td>
<td>0.28</td>
<td>1.49</td>
</tr>
<tr>
<td>20</td>
<td>1.59</td>
<td>0.32</td>
<td>1.91</td>
</tr>
<tr>
<td>30</td>
<td>2.03</td>
<td>0.44</td>
<td>2.47</td>
</tr>
<tr>
<td>40</td>
<td>2.54</td>
<td>0.49</td>
<td>3.03</td>
</tr>
<tr>
<td>50</td>
<td>3.10</td>
<td>0.61</td>
<td>3.71</td>
</tr>
<tr>
<td></td>
<td><strong>Experimental Scenario 4:</strong> Number of Producers =92 and Number of Brokers = 32</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>1.18</td>
<td>0.26</td>
<td>1.44</td>
</tr>
<tr>
<td>20</td>
<td>1.80</td>
<td>0.29</td>
<td>2.09</td>
</tr>
<tr>
<td>30</td>
<td>2.24</td>
<td>0.36</td>
<td>2.60</td>
</tr>
<tr>
<td>40</td>
<td>2.73</td>
<td>0.43</td>
<td>3.16</td>
</tr>
<tr>
<td>50</td>
<td>3.14</td>
<td>0.51</td>
<td>3.65</td>
</tr>
</tbody>
</table>

The data collected from our experiment regarding the total average latency of each message for the distributed message delivery system with varying number of consumer are shown in Fig. 7. For real-time CV applications, latency is the most important performance factor of a delivery system. An increase in the number of brokers (see Fig. 7) causes a decrease in average end-to-end latency, a trend that is more obvious at peak hours when additional consumers utilize most of the processing power of the broker cluster. Therefore, a larger broker
cluster must be available during peak hours, since the increased processing power is amortized by the added workload from consumer requests. Even during these peak hours, the use of two brokers for our experimental scenario (considering 50 consumers) will give an approximate end-to-end latency average of 7.95 milliseconds, providing ample room for a stream processing latency to meet all the time and spatial contexts requirements for real-time applications specified in CVRIA [8]. Increasing the number of brokers in the broker cluster makes this latency even more insignificant. However, increasing the number of brokers provides the most for the applications with a large number of users at peak hours; otherwise, the performance gain is minimal, as shown in the case of the 10 consumers.

Another essential metric, in addition to latency, is producer and consumer throughput. Throughput can be defined as the number of records sent or received per second. The throughput at the producer end (as shown in Fig. 8) helps establish the upper limit of RSU’s report frequency, which is the maximum velocity of message producing events that the delivery system can handle if no message buffering occurs. The y-axis shows the MB/s throughput for each of the 92 producers. Even though the total message volume of the 92 producers overflows the network bandwidth, the actual accumulated throughput of each producer is far less than network bandwidth. The handshake time between the producers and the brokers, which slowed the message transfer and reduced the throughput could be the cause of this reduction. During the off peak hours, the best throughput was 0.35MB/s for each producer when using 32 brokers, creating a 1500 record-per-second limit for processing message flow from either RSUs or CVs. The number of consumers less likely affects the improvement in a producer’s throughput when the numbers of brokers increases, as shown in Fig. 8. Thus, increasing the number of brokers appeared to exhibit a similar throughput improvement pattern among all consumer cases, regardless of peak or off-peak hours.
Figure 7. Total average latency for the distributed message delivery system by varying the number of consumers and brokers.

Figure 8. Throughput of producer for the distributed message delivery system by varying the number of consumers and brokers.
Figure. 9. Throughput of different number of consumers for the distributed message delivery system by varying the number of brokers.

![Graph showing throughput of different number of consumers and brokers.]

Figure. 10. The latency distribution for the distributed message delivery system with 16 brokers by varying the number of consumers.

![Graph showing latency distribution with different number of consumers and brokers.]

While in latency case as shown in Fig. 7, higher performance improvements are observed when the number of consumers are large (peak hours). Fig. 9 shows the throughput of consumers with different number of broker and consumer scenarios. The throughput of each consumer is much larger than the throughput of each producer. This discrepancy is due to Kafka’s efficient implementation of consumer read process [9]. For the 32-broker scenario,
the per-consumer throughput dropped linearly as the number of consumers increased from 10 to 50, indicating that the accumulated consumer throughput was approximate to the network bandwidth limit. For other cases, the consumer throughput was satisfactory even with low numbers of broker clusters, indicating that the improved performance in consumer throughput was insensitive to the broker cluster size. In most real-time application, consumers cannot receive large batches of data simultaneously. Therefore, when the number of brokers increases, the percentage of latency reduction is higher than the percentage of increment in consumer throughput. Specifically as shown in Figs. 7, 8, 9, it is desirable to use a larger broker cluster for CV applications managing many consumers at peak hours, while a broker cluster size between 4 or 16 for applications with few consumers ensures performance without additional expense on the larger cluster. We did however notice a significantly higher maximum latency than average latency in all of our experimental scenarios. Note the end-to-end latency distribution of the 16-broker system scenario shown in Fig. 10. Here, the average latencies for all consumer cases were less than 5 milliseconds with the 99th percentile latencies, which is still less than 12 milliseconds while the maximum latencies jumped to 500 milliseconds. Although there was a less than 1 percent occurrence of these maximum latencies, they nonetheless determined the degree to which a delivery system served a real-time application. This discrepancy may be due to a slow File Input/Output (I/O) thread, a networking congestion error, broker unavailability, or a slow partition of a topic. Determining the precise origin of this long maximum latency problem is difficult due to the lack of fine-grained tools to monitor such behaviors. It is also difficult to determine the correlation between these minor events and their root causes. Although it is possible to send multiple duplicate messages through different producers simultaneously to avoid the maximum latency ceiling of sending each message, a message redundancy will occur in the brokers with this strategy. Therefore, we will conduct further studies to determine the possible causes of this long maximum latency problem to derive a solution.

USDOT’s system requirements of Intelligent Network Flow Optimization (INFLO) Prototype for CV Dynamic Mobility Applications (DMA) state that the Traffic Management Entity (TME) shall have the capability to obtain data from the traffic sensors on every 20 seconds [49]. The highest total average end-to-end latency is 7.95 milliseconds in our
experiments. These latencies were much lower than the recommended values for the USDOT system requirements for CV pilot deployments. However, the minimum system requirement, such as the minimum number of brokers, is decided by the message delivery workload for the Kafka message delivery system, e.g., how many producers sending messages at the same time and how many consumers are reading those messages at the same time. The strength of our distributed message delivery system is that the delivery capability of the brokering system is dynamic and adaptable to workload in run time. For example, we can always start with the minimal number of Kafka brokers (i.e., 2) and add up the number of broker as needed when the latency performance rises above the USDOT requirement. Our experiments demonstrated that we could satisfy the USDOT latency requirement for different CV applications with the least number of machines (i.e., 2). In addition, our experiments also demonstrated that the latency performance improves further when we add more number of brokers under experimental scenarios 2, 3 and 4 (as shown in Table III and Fig. 7). Also, we found that in moving the design of ITS data infrastructure from a vertical, top-down approach as shown in [7] (Fig. 11(a)) to a horizontal, one-level approach (Fig. 11(b)) using publish/subscribe streaming message delivery model improved the flexibility, scalability, and resiliency of the entire data infrastructure. Publish and subscribe streaming messages represent sending and receiving of streaming messages, respectively. Moreover, our new approach simultaneously maintains the different storage components required at different scales of operation, and significantly reduces the level of dependency among the different storage components. As connected vehicle systems can support multiple applications simultaneously (such as multiple safety, mobility, environmental and energy applications could be supported by the same CV infrastructure on a roadway) and public investments will only include infrastructure investments (such as investments in roadside units and backend computing infrastructure), connected vehicle systems can potentially provide significant economic benefits compared to its cost [50].
Figure 11. a) Hierarchical multi-level data infrastructure architecture for Big Data/Data Intensive computing;
Figure 11. b) Single-level distributed message delivery infrastructure architecture for Big Data/Data Intensive computing.
CONCLUSIONS

In this paper, we evaluated a distributed message delivery system for connected vehicle systems in which multiple CV applications ran simultaneously with data transfers between RSUs and different transportation centers. For this purpose, all information flows defined for diverse CV applications in CVRIA were characterized based on time and spatial contexts of data sent from various transportation centers to RSUs and data sent from RSUs to various transportation centers. Our analyses indicated that the message delivery system reduces message redundancies by identifying unique information flows in multiple CV applications. This efficient message delivery system provides a strategy that enables large-scale ingestion, curation, and transformation of unstructured data (including roadway traffic-related and roadway non-traffic-related data) into labeled and customized topics, which can be used by large numbers of subscribers or consumers for various CV applications.

We evaluated the distributed message delivery system by developing a prototype infrastructure using Kafka, which is an open source message broker platform. We then compared the performance of this system to the minimum latency requirement for CV applications. Experimental analyses were performed using two distributed computing testbeds, and the latencies and throughput of the message delivery system for CV applications were examined.

Different experimental scenarios with different numbers of Kafka brokers and consumers were executed to evaluate the message delivery system’s performance. The evaluation of this distributed message delivery system shows that the highest total average latency was 7.95 ms, which was measured as the difference between the time when a message was generated by a RSU to the time of its reception by a Traffic Management Center. The highest total average latency, out of all the scenarios considered in our experiments, was found to comply with the recommended USDOT system requirements for CV pilot deployments. However, this latency includes only the delivery time of a message and not the processing times, such as the time required for aggregation and complex data transformation. The results of this study will help ITS professionals to develop a message delivery system to support CV applications and provide an efficient distributed message delivery system that provides many benefits over the current paradigm of centralized message delivery systems.
REFERENCES


APPENDIX

Publications, presentations, posters resulting from this project:

Journal:

Peer-reviewed conference publications and poster presentation: