SPECTRUM OF ENGINEERING SCIENCES
ISSN (E) 3007-3138 (P) 3007-312X VOLUME 4, ISSUE 1, 2026

TOWARDS TIME-CRITICAL BIG DATA APPLICATIONS IN THE INTERNET OF VEHICLES (IOV)

Shah Hassan Khan[1], Abdul Haseeb Malik[2], Najeebullah[*3], Qazi Ejaz Ali[4], Waheed ur Rehman[5], Saif ur Rehman[6]

[1,2,*3,4,5,6]*Department of Computer Science, University of Peshawar, Peshawar, Pakistan*

[*3]najeebullah9172@uop.edu.pk

DOI: https://doi.org/10.5281/zenodo.21393759

Keywords
Internet of Vehicles (IoV); real-time stream processing; computational offloading; Road Side Units (RSU); Apache Kafka; Apache Storm; response-time analysis; Mobile Edge Computing (MEC).

Article History
Received: 30 November 2025
Accepted: 06 January 2026
Published: 30 January 2026

Corresponding Author: *
Najeebullah

Abstract
The Internet of Vehicles (IoV) connects vehicles, humans, and infrastructure to produce continual large volume data streams. Real-time processing of this data enables a new class of vehicular applications, however, sending all such events to a centralised cloud will have latency effects that will be unacceptable by many of these applications. To overcome this limitation, this paper suggests a computational offloading model that can process IoV big data in near real time by performing the computation directly on Road Side Unit (RSU) compute resources without the cloud-only processing round-trip delay. The architecture is formalized by utilizing a response-time analysis (RTA) model to determine per node and per cluster utilisation bounds and guarantee that time-critical tasks will meet their response time deadlines under concurrent application load. To test the proposed framework, a real time, location-based advertisement recommendation system was implemented on Apache Kafka and Apache Storm. The inter-RSU travel budget of 3,000 milliseconds was computed for a vehicle travelling at 120 km/h between two RSUs separated by 100 metres. Across 1,000 simulated vehicle events, the combined worst case parsing time and recommendation logic was 1.523 milliseconds, which is well within the 3,000 millisecond inter-RSU travel budget. The results prove that the stream processing can be implemented at RSU without relying on cloud and meet the timing requirement of practical IoV applications.

I. Introduction

In the last ten years, vehicular technology has made tremendous progress in the context of increasing numbers of vehicles with on-board communication and computation devices. The Intelligent Transportation System (ITS) [1] will be the backbone of the Vehicular Ad-hoc Networks (VANETs) which connect vehicles with each other. VANETs are heavily discussed for use in the automotive industry and the electronics industry, in which electrical and electronic (E/E) systems in the vehicle and between vehicles are constantly exchanging data, irrespective of their location [2]. The VANETs are a specific class of mobile ad hoc network (MANETs) with high vehicle mobility, high degree of connectivity and fast changing network topology. On-board Units (OBUs) will give information processors the computational power required for local processing of vehicular data, while radio interfaces such as IEEE 802.11p will enable direct vehicle-to-vehicle communications [3]. This trend is being fueled by the wider Internet of Things (IoT) movement, in which real-world objects communicate and share information independently through the Internet. Combined with the principles of IoT, VANET technology has led to the emergence of the Internet of Vehicles (IoV) [4] consisting of two types of direct communications: vehicle-to-vehicle (V2V) and vehicle-to-infrastructure (V2I) communications, as well as vehicle-to-pedestrian (V2H) communications. Data produced by IoV is

considered as 'Big Data' [5] as every vehicle can generate a data of up to 30 GB per day [6] of data. While some applications such as infotainment, vehicle platooning, lane change advisory are tolerant to moderate latency, some applications like collision avoidance, traffic-congestion detection, and dynamic lane management require consistently low latency [7]. The key is to meet the demand by intensive computation at the edge of the network—on-board sensors and RSUs—not in a remote data centre [8].

The computing power of on-board units is less than the amount of data generated during the movement of the vehicles [9] and the cost of computation and energy to process this amount locally is high [10, 11]. Initial traffic management systems (TMS) dealt with real-time processing by sending V2V data to a centralised server [12]. Floating Car Data (FCD) technology furthered this model by sending the vehicle-generated data to an internet-based control centre where tracking and detection can be carried out [13]. In both cases, a problem is caused by the structural nature known as backward delivery: the longer the distance between a vehicle and its serving RSU, the longer the time the vehicle will take to receive the alerts, which diminishes the utility of the real-time notification (in the case of an accident, for example, the alert should reach the approaching vehicles before they arrive at the site) [14].

In this paper, the authors aim to fill this gap by introducing a computational offloading model directly deployed on compute hardware of the RSU itself through a stream-processing cluster. The system also processes IoV data at the edge, eliminating the need to transfer it to the cloud and minimizing end-to-end latency, as required for time-critical applications in the vehicular domain. The proposed architecture is formalised by providing a response-time analysis (RTA) model which provides utilisation bounds for each RSU node and for the cluster as a whole, and validated by a real-time location-based advertisement recommendation system based on Apache Kafka and Apache Storm.

This paper continues with the following outline. The related literature is summarized in Section II, which includes the computational offloading techniques and the edge-computing techniques for Vehicular networks, including V2V, V2I and V2X communication paradigms. In Section III, a framework and the mathematical formulation for cluster management and schedulability are proposed. The implementation environment is described in Section IV and the advertisement-recommendation case study for evaluation is described. In the experimental results section (Section V), the results of extracting vehicle information and recommending advertisements are presented. These results are discussed in the context of previous studies and are considered in relation to their wider implications in Section VI. This section VII summarizes the work presented in the paper and outlines some directions for future research.

Abbreviations and Acronyms

Acronym	Definition
IoT	Internet of Things
MANET	Mobile Ad Hoc Network
VANET	Vehicular Ad Hoc Network
ETSI	European Telecommunication Standards Institute
C-ITS	Cooperative Intelligent Transportation Systems
IoV	Internet of Vehicles
DSRC	Dedicated Short Range Communication
ITS	Intelligent Transport Systems
URLLC	Ultra Reliable Low Latency Communications
RSU	Road Side Units

Acronym	Definition
MEC	Mobile Edge Computing
WAVE	Wireless Access in Vehicular Environments
CPS	Cyber Physical Systems
TMS	Traffic Management Systems
GPS	Global Positioning System
V2V	Vehicle to Vehicle
V2I	Vehicle to Infrastructure
V2X	Vehicle to Everything

II. Literature Review

The pace of change for mobile devices, especially vehicles, has set expectations of millisecond responsiveness to applications. The topologies of vehicles are dynamic and connectivity is not guaranteed, so processing vehicle data requires speed and flexibility. This section reviews all previous research related to offloading and edge processing which will guide the design that is proposed in this paper.

The first example is Satyanarayanan's idea of "cyber foraging" [15] where mobile devices outsource computations to the nearby static devices [16]. This concept was later formalised in computational offloading [17] where the computation is delegated to the nearest resource capable server, and then it is returned to the resource-constrained device immediately. Cloud computing contributed to this trend by providing on-demand compute resources more widely available, and helping to make the case for offloading a viable option.

In the Chun and Maniatis "clone cloud" proposal, the mobile device state is partly replicated on a VM that resides on a cloud, and the VM determines when to offload and what to offload [18]. Cloud computing in general has been found attractive based on its elasticity, data-insight and disaster-recovery properties, and seems to provide a clear economic benefit to both providers and clients [19]. Automotive systems became one of the key applications for such infrastructures with the introduction of Industry 4.0 [20].

However, cloud-only architectures have well-documented disadvantages, such as application response time, WAN latency, and bandwidth limitations [21] are all significant challenges in a highly mobile environment with constantly changing topology. The limitations inspired the concept of Mobile Edge Computing (MEC) which was taken up as a concept by the European Telecommunications Standards Institute (ETSI) in 2014 [22]. MEC is a converging and co-locating of telecommunications and cloud computing technologies to process data at the network edge, near the point of origin.

The requirements for a modern Intelligent Transportation System are a high computational throughput, low latency and reliable connectivity. These needs are common to RSUs, access points, base stations and even pedestrians [23]. On both sides of the Atlantic, vehicular communication has been influenced by the standardisation efforts, with both the Cooperative Intelligent Transportation Systems (C-ITS) in Europe and the Dedicated Short-Range Communications (DSRC) in the U.S. standing on the shoulders of the IEEE 802.11p amendment, both operating in the 5.9 GHz band for direct, ad hoc information exchange at the physical and MAC layers [24].

However, traditional traffic management was based on CCTV monitoring, speed trackers and pre-configured detection algorithms [25]. On-board radar or LIDAR can only detect vehicles in line-of-sight at about 200 metres [26]. The idea of edge computing first emerged in 2009 as "Cloudlet" [27] — the closer physical proximity of computation and storage to the user equipment. Edge caching goes one step further by caching and serving the content for popular requests on the network edge, which is quite appropriate for a vehicular network environment with high mobility, dynamic content requirements and heterogeneous connectivity [28].

2.1 Computational Offloading Techniques

On-board devices are resource constrained and hence often rely on external compute resources, such as the cloud, RSUs or nearby vehicles, termed computational offloading. This work is classified into the following communication patterns: V2V, V2I and V2X.

2.1.1 Computational Offloading in V2V

There are some studies that investigate offloading strategies for the V2V environment with particular attention on offloading-target-selection, transmission efficiency and optimisation algorithms.

Satyanarayanan et al. suggested a three-layer architecture for offloading in IoV namely Cloud Layer, Cloudlet Layer, and Fog Layer [29]. The Cloud Layer is located many kilometres away from vehicles and contains a conventional Traffic Management Server to process the data and issue instructions to traffic-management personnel. The Cloudlet Layer spreads this processing out: RSUs at each region are local points of access that receive data from the vehicles for immediate processing. The Fog Layer goes a step further, creating compute nodes from vehicles that fall within the RSU's communication range, and processing the data without sending it to a cloudlet.

2.1.2 Computational Offloading in V2I

V2I communication is a means of adding connectivity beyond the direct V2V communication range: When a vehicle is not within the range of its direct V2V communication, it can offload data to the nearest RSU. There have been several studies directed at specific timing requirements this brings.

Amadeo et al. introduced the idea of joint resource allocation for both backhaul and wireless links to solve the challenge of connection-time between vehicles and RSUs, which is treated in two phases: an integer linear programming (ILP) problem for maximizing the system throughput and a low complexity heuristic for link scheduling, as well as a simple bandwidth-allocation algorithm [30]. Basanta-Val et al. analysed the trade-off between the amount of data downloaded at the RSUs and the cost of the download separately, highlighting the fact that as vehicles are mobile, a single download will be spread across multiple RSUs, thus creating a delay which is very sensitive to users' cost tolerance. Their adaptive algorithm dynamically adjusts a balance of these competing factors depending on the time available between RSU handovers [31].

2.1.3 Computational Offloading in V2X

Vehicle to everything (V2X) expands V2V, V2I, vehicle-to-network (V2N), and vehicle-to-pedestrian (V2P) communication to a single concept.

To handle the massive amount of data generated by vehicles in real-time, Wang et al. investigated the ability of V2X communication to absorb this data flow and proposed a content-delivery control algorithm that leverages route-based prediction to deliver data efficiently while considering the offloading performance, the data-loss ratio and the bandwidth reduction [32]. This type of work shows that judicious scheduling at the application layer can make a significant difference in the latency overhead associated with V2X exchanges.

III. Proposed Framework and Methodology

This paper's framework extends the time-critical big-data architecture introduced by Basanta-Val et al. [31] which is adapted to the vehicular domain. The result is the four-layer Flow shown in Figure 1: Tools, Applications, Infrastructure and Hardware.

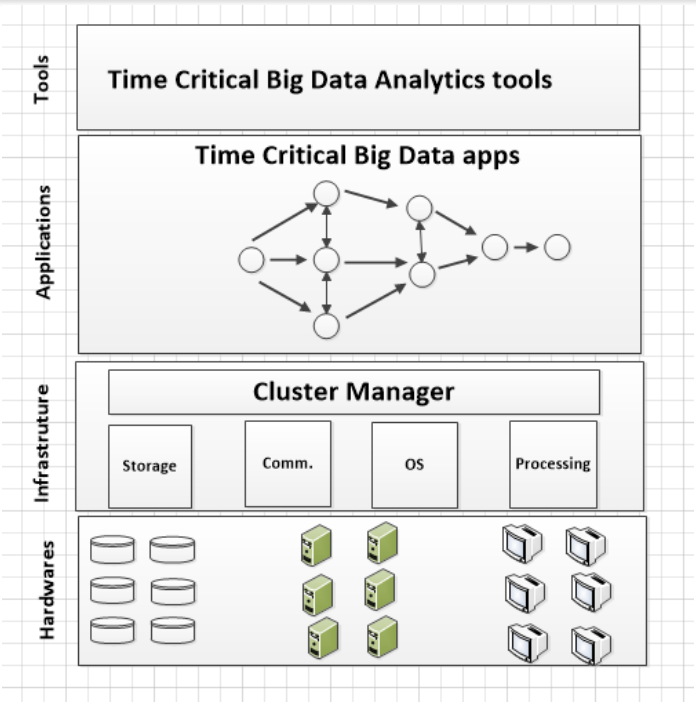

Fig. 1. Time-critical big data application flow diagram, adapted to the vehicular environment. Four layers—Tools, Applications, Infrastructure, and Hardware—govern how time-critical and analytics workloads are scheduled and executed.

3.1 Tools Layer

The Tools layer includes the time-critical big data analytics engines that process the data stream. They can execute time-critical and non-time-critical workloads, but for time critical workloads, they must be able to ensure that the deadlines are met by having direct control over the resources that are used, such as: disk bandwidth, memory, CPU cores, and network communication.

3.2 Applications Layer

Proposed system deployment models are based on Directed Acyclic Graph (DAG) applications that allow for deployment across large compute clusters and maintain the order of dependency between the computing stages. There are two types of applications:

3.2.1 Time-critical applications: Tasks that need to react in sub-second time windows, like early collision detection, traffic-congestion alerts, lane change warnings, emergency response coordination, etc.

3.2.2 Analytics applications: workloads that draw meaning from vast amounts of data and not those with strict deadlines like in-vehicle entertainment, social media browsing, and location-based ads.

3.3 Infrastructure Layer

The infrastructure layer is a layer of facilities that are available for use by applications to get the resources they need. These clusters of compute resources scale up with the demand of the applications, including disk, memory and processing resources per region. This layer consists of four modules:

3.3.1 **Operating System:** Accesses resources through a well-defined interface, some operating systems are designed to have real-time performance, and to guarantee that resources will be available by a specific time.

3.3.2 **Storage Manager:** controls the various storage tiers offered, including different quality-of-service policies of interest to time-critical processing.

3.3.3 **Communication Manager:** Selects and implements the right security protocol for a particular network and security policy.

3.3.4 **Node Manager:** Responsible for configuring individual nodes with pre-installed operating system, drivers and software stack.

3.4 Hardware Layer

The Hardware layer is the component that is essential for the rest of the layer to work. It is the physical infrastructure that upon which the other layers are dependent, without which none of the other layers would be able to support computation, storage and communication.

3.5 Stream Processing System Model

Suppose that there is a time-critical big data stream processing system with m applications in it, represented as:

$$TC\_BDi = \{ TC\_App_1, TC\_App_2, TC\_App_3, ..., TC\_App_m \}$$

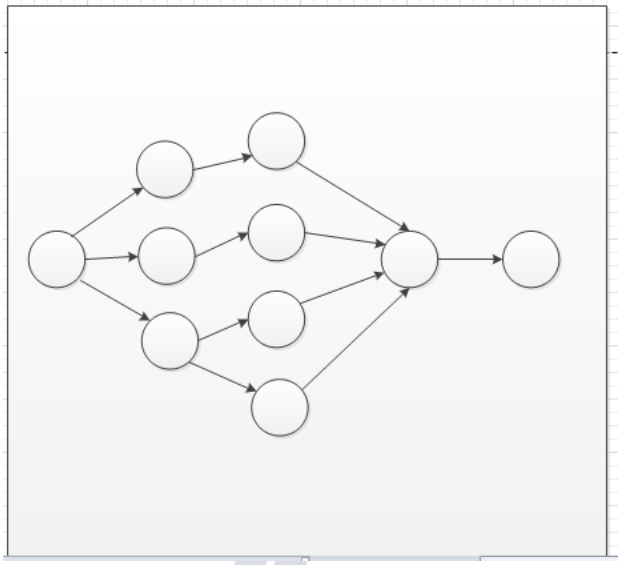

Fig. 2. Multiple-application stream-processing model. Each application contributes a continuous data stream that must be processed within its individual deadline.

It is a continuous stream, which needs to be processed in a timely manner to deliver the result in real-time. Each application has a computational model defined by three parameters: minimum inter-arrival time (T_i), worst-case execution time (C_i), and the deadline for it to finish (D_i):

$$TC\_App_i = \{ T_i, C_i, D_i \}$$

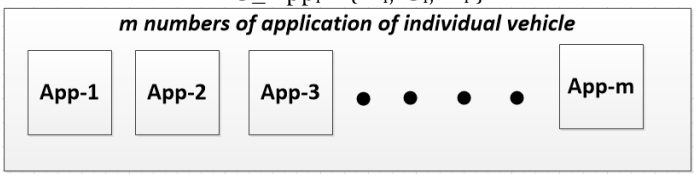

Fig. 3. The m applications associated with an individual vehicle, each potentially time-critical or non-time-critical.

A combination of time-critical and non-time-critical workloads can be executed in each vehicle, up to m applications. All applications run within the cluster located at the closest RSU: time-critical applications are run locally on the RSU, and non-time-critical applications are sent to the cloud.

3.6 Mathematical Model for Cluster Management

In Infrastructure layer described in Section IV-C, the cluster of n nodes (machines) corresponds to one node (machine) in each RSU as shown in Figure 4:

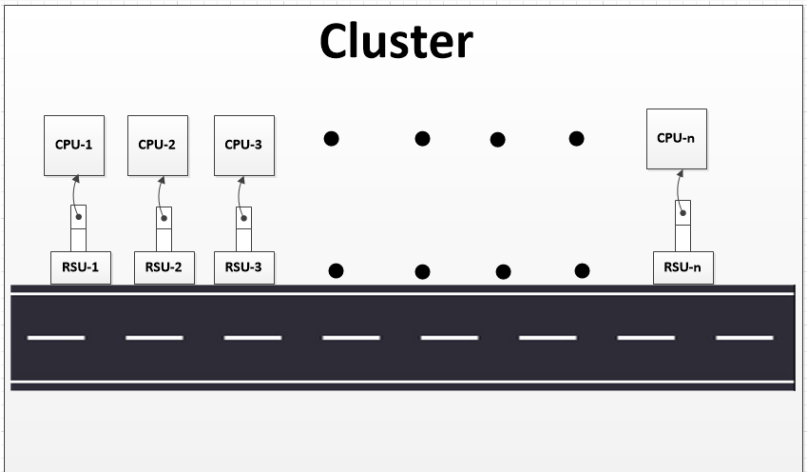

Fig. 4. Cluster environment of the proposed vehicular network. Each RSU hosts a dedicated processing node within the larger cluster.

$$TC\_Cluster_i = \{ \pi_1, \pi_2, \pi_3, \ldots, \pi_n \}$$

The worst-case response time, $WCRT_n$, of a specific node needs to be calculated first to determine the utilisation bound of an individual node. RTA is a method that calculates $WCRT_n$ using the execution time (C_i), minimum inter-arrival time (T_i), the task deadline (D_i) and the contribution of higher-priority tasks (HP_i) executing on same node (π_i). The model is iterative, and the assumption made is that each task has the same deadline as its minimum inter-arrival time ($T_i = D_i$):

$$WCRT_n = \Sigma \text{ over } HP(i,j) \text{ of } \pi_i \text{ of } (C_i / T_i)$$

The worst case execution time is further broken down into sequential and parallel compositions of processing stages. Let (i) and (j) be two consecutive stages:

$$WCRT(i \rightarrow i')(j \rightarrow j') < WCRT_{i'j'} + WCRT_{i'j'}$$

For two stages composed in parallel:

$$WCRT(i \| i')(j \| j') < \max(WCRT_{ij} , WCRT_{i'j'})$$

An individual node's utilisation bound, $U\pi$, is the sum of the execution time of all of the higher-priority tasks such that $U\pi$ is bounded as follows:

$$U\pi = \Sigma\, HP(i,j)\pi(i,j)\,(C_i/T_i) < U\_max - \min(m, 0.8)$$

That is, the overall utilisation of a node should be less than its maximum utilisation bound minus the minimum utilisation across all m applications executing on the node, and should not exceed 80% of the total execution capacity while still leaving room for spikes in demand.

In order to ensure that all tasks are completed on time under continuous streaming load, a cluster-level utilisation bound (U_cluster) is also enforced:

$$U\_cluster = \Sigma\, P(i,j)\pi(i,j)\,(C_i/T_i, D_i) < m \times (U\_available - \max\{ P(i,j)\pi(i,j)\,(C_i/T_i, D_i), 0.8 \})$$

3.7 Offloading Computational Tasks to Infrastructure

When possible, each vehicle will process time-critical data locally on the OBU. If the computation is greater than the capacity of the OBU, the task will be offloaded to the nearest RSU. The resulting processing structure is shown in Figure 5: each RSU will have one compute unit to manage time-critical data for vehicles in its range.

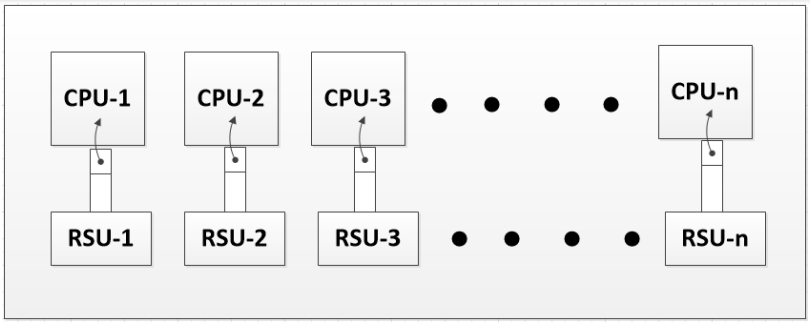

Fig. 5. Computational unit deployed at each RSU. As vehicles move out of range, processing state transfers to the next nearest RSU.

The number of CPUs being used at every RSU determines real time processing capacity. If a vehicle moves out of the coverage area of the RSU it is currently associated with, its state of being processed is passed on to the next closest RSU, thus ensuring continuity of service.

The overall flow of a time-critical task from the OBU to the processing at the RSU and the possible distribution of the task over the RSU cluster is summarised in Figure 6.

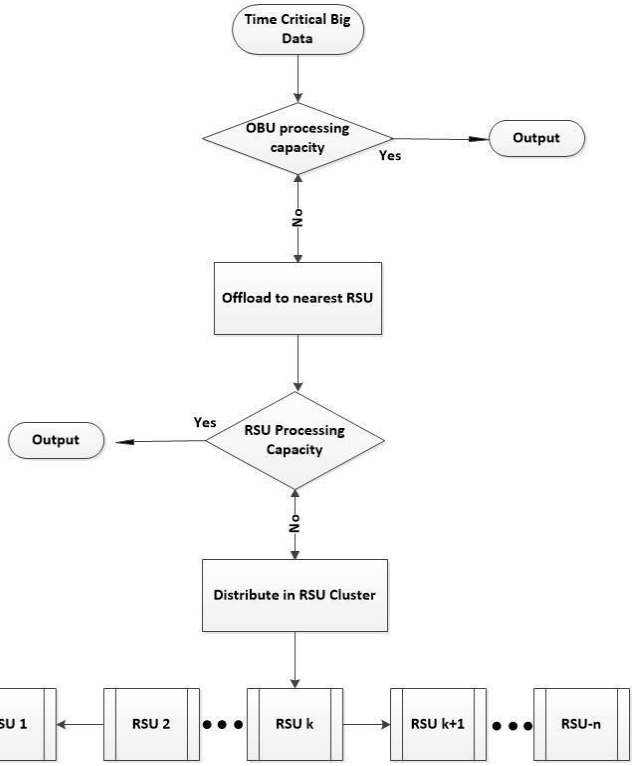

Fig. 6. Task allocation flow diagram. A time-critical task is processed locally if OBU capacity allows; otherwise it offloads to the nearest RSU, and if RSU capacity is also exceeded, the task is distributed across the RSU cluster.

IV. Implementation

The proposed framework have used Apache Zookeeper, Apache Kafka, Apache Storm and The Scalable Big Data Simulator for implementation.

4.1 Environmental Setup

It was installed on Windows 10 with Apache Storm 2.1.0 and Apache Kafka 2.11-2.4.0. The Scalable Big Data Simulator generates simulated vehicle tuples and sends them to Apache Kafka for consumption and forwards them to Apache Storm for processing. When a vehicle enters an RSU, it sends its data to the RSU which has a specialized Apache Storm node that runs the heart of the business logic and sends a result back to the vehicle. The experimental set up is detailed in Table 1.

Table 1. Experimental Setup

Parameter	Value
Operating System	Windows 10
Processor	Intel Core i7, 8th Generation
CPU Cores	8
Clock Speed	2.2 GHz
Memory	16 GB
Language	Java (Version 1.8)
IDE	Eclipse 2021-19 (4.21.0)
Apache Kafka	2.11-2.4.0
Apache Storm	2.1.0

4.2 Case Study

Based on the numerous time-critical IoV applications that could be exploited, for the purpose of validation, location-based targeted advertising was selected. Business vendors sign up on IoV platform and place ads with RSUs. The vehicle sends information to an RSU as it crosses it, and the RSU receives the information, processes it and returns an ad that is relevant to the vehicle's material. The distance between the RSUs is 100 metres, and the maximum vehicle velocity to be considered is 120km/h, that means that a vehicle has 3,000 milliseconds between two RSUs. Within this window, an apache storm node set up at each RSU will process the data flowing in to it in a few milliseconds, not seconds, and this is the time recorded in the experiment. The cluster configuration for this case-study is shown in Figure 7.

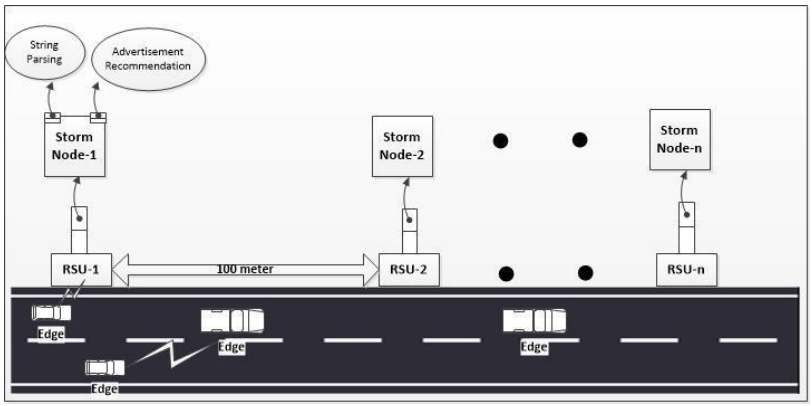

Fig. 7. Cluster setup for the advertisement-recommendation case study. Each RSU hosts a Storm node responsible for string parsing and advertisement recommendation.

V. Results

In the results part, the experiments were carried out against the implemented system. Advertisement recommendation based on vehicle position and vehicle direction was used as the scenario to test the proposed framework, targeting to reduce latency of data processing at the cloudlet layer. The processing was carried out in two phases: string parsing, and advertisement recommendation, and the time consumed by each phase was measured in each RSU in milliseconds.

5.1 Vehicle Information Extraction

On receiving a message at an RSU, the system first parses the message to identify the vehicle ID and vehicle number, as well as the direction, position, number and type of passengers. The processing time is plotted against the sequence of 1,000 advertisement messages in figure 8.

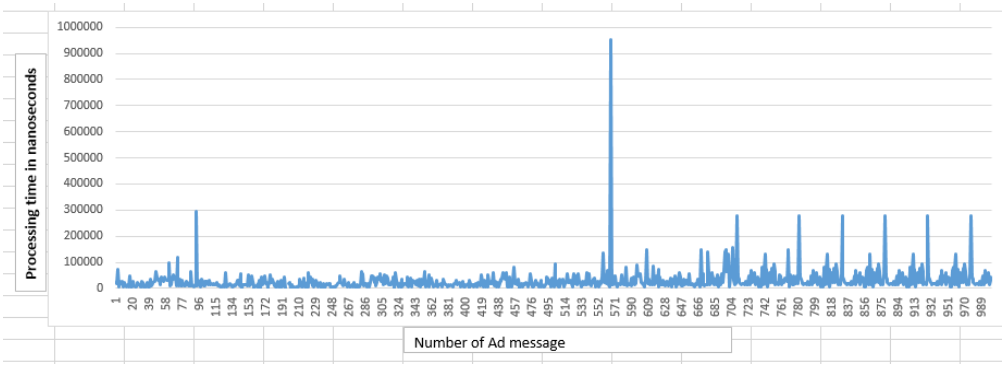

Diagram 1. Vehicle information extraction: processing time (nanoseconds) across 1,000 advertisement messages.

Most messages are being parsed in a few nanoseconds, but a few messages have spikes in the trace which are in the nanoseconds range, and in the case of one message, it's up at nearly 950,000 nanoseconds or 0.95 milliseconds, which is much higher than normal. This does not represent any systematic error in the parsing code, but rather a pause that could either be a well-understood characteristic of JVM-hosted real-time systems (garbage collection pause) or a transient scheduling delay on the OS level during the course of the actual Java program execution. Table 2 summarises the descriptive statistics for this stage.

Table 2. Vehicle Information Extraction — Summary Statistics

Messages	Mean (ms)	Std. Dev. (ms)	Min (ms)	Max (ms)
1,000	0.0301	0.031	0.001	0.572

The average parsing time for 1000 messages is 0.0301 ms, meaning the typical message can get processed in a blink, relative to the allowed parsing time. The standard deviation of 0.031 ms represents the normal variation of data, including occasional outliers like the one in Figure 8, and the minimum and maximum values shown (0.001 ms and 0.572 ms respectively) define the range of data observed.

5.2 Advertisement Recommendation

The system then uses recommendation logic based on the extracted information after string parsing. The advertisements are sorted into audience categories (Male, Female, Children) and one of four time slots, the appropriate advertisement is chosen depending on time of day, passenger numbers and passenger type. Figure 9 shows processing time against number of passengers for the same set of 1000 messages.

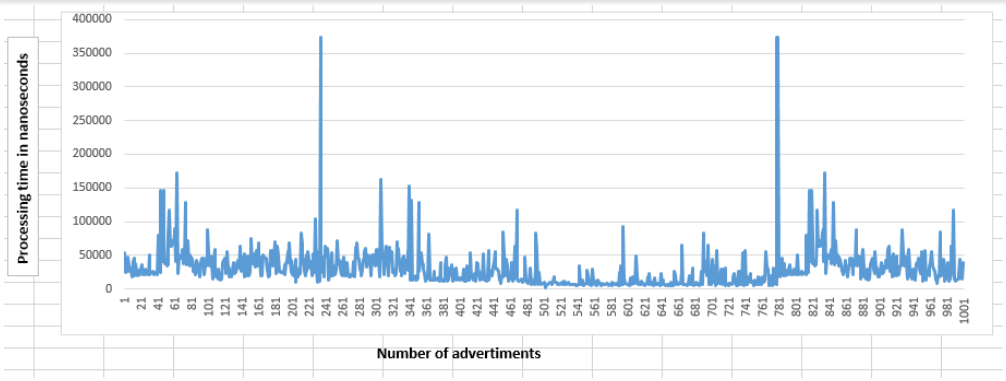

Diagram 2. Advertisement recommendation: processing time (nanoseconds) across 1,000 messages.

Table 3. Advertisement Recommendation — Summary Statistics

Messages	Mean (ms)	Std. Dev. (ms)	Min (ms)	Max (ms)
1,000	0.029	0.042	0.003	0.950

The results provide a similar profile for the recommendation stage (0.029ms average, 0.042ms standard deviation, 0.950ms maximum), together with the parsing stage, forming the following deadline analysis:

Table 4. Deadline Analysis Parameters

Parameter	Value
Vehicle speed	120 km/h
Distance between RSUs	100 metres
Time for a vehicle to travel between adjacent RSUs	3,000 milliseconds
Worst-case execution time, string parsing	0.572 milliseconds
Worst-case execution time, advertisement recommendation	0.950 milliseconds
Combined worst-case computational cost	0.572 + 0.950 = 1.523 milliseconds

With a combined worst case computation time of 1.523 ms, you have a margin of more than 1,998 x the measured cost for these computations within a travel-time budget of 3,000 ms. Despite the occasional outlier cases in Figures 8 and 9, the system meets its deadline requirement, providing strong evidence that RSU-level Kafka–Storm processing is suitable for this type of time-critical IoV application.

VI. Discussion

6.1 Comparison with Related Work

For comparison, the proposed system is placed in Table 5 with three representative prior systems, including the three-layer offloading architecture based on cloudlets [29], the ETSI MEC reference model [22] and the V2X content delivery using route-prediction [32] algorithms.

Table 5. Comparison with Related Offloading Approaches

Property	Proposed System	Cloudlet [29]	MEC ETSI [22]	V2X Routing [32]
Stream Processing Engine	Kafka + Storm	Generic VM	Not specified	Custom controller

Property	Proposed System	Cloudlet [29]	MEC ETSI [22]	V2X Routing [32]
Formal Schedulability Model	✓ RTA-based	✗ No	✗ No	✗ No
RSU-Level Deployment	✓ Yes	Partial	✓ Yes	✓ Yes
Empirical Latency Validation	✓ 1,000 events	✗ No	✗ No	Simulation only
Cluster Utilization Bound	✓ Mathematical	✗ No	✗ No	✗ No
Case Study Application	Ad recommendation	Generic offload	Generic	Content delivery

Formal schedulability models and empirical latency validation on a real stream-processing stack are not found in any of the compared systems. The Cloudlet architecture [29] sets the concept of the three layers separation we use in this paper, but it does not define a specific processing engine, nor give any quantitative latency data. ETSI MEC model [22] sets forth the architectural principles behind edge processing in telecom networks and it does not go beyond vehicular context. The validation of Wang et al.'s work on content delivery, V2X [32] is only done in simulation, and there is no working prototype of a RSU. In contrast, the system proposed here is built on an explicit model of response time analysis, used on an actual Kafka–Storm infrastructure, and put to the test with 1,000 real processing events – providing some empirical backing compared to the infrastructure used by the other systems.

6.2 Significance of the Results

Combined worst case computation time of 1.523ms, with a travel budget of 3,000ms, shows that stream processing at the RSU level is not only theoretically correct but also practically achievable with commodity hardware. This margin is important because it allows for significant additional load: although the system might be serving multiple applications simultaneously per vehicle, or be subject to occasional spikes in load, as the examples in Figures 8 and 9 show, these are still within the margin of available headroom.

The model for cluster-utilisation developed in Section IV is a generalisable basis for reasoning about RSU capacity planning beyond this one case study. The same formulation also applies to other time-critical applications in IoV, such as collision warnings, lane change advisories, and congestion warnings, for an arbitrary number of applications, rather than deriving a new one for every application. This is important because the advertisement-recommendation case study is not just a narrow-spectrum, one-purpose outcome, but a proof of concept for a class of applications.

It is important to note the outlier values observed in both Figures 8 and 9 as they are not noise. While they are still quite small, up to 0.95 ms, in such a deployment with many parallel applications running in a vehicle, the sum total of all the pauses along a longer processing chain could consume this margin. This observation inspires the JVM-tuning and alternative-runtime considerations in future work presented in Section VIII.

VII. Conclusion, Limitations, and Future Work

This paper suggested a computational offloading solution for time-critical IoV applications by implementing a stream processing cluster in the RSU level to process vehicle data in near real time without passing through the cloud. Utilisation bounds, both per node and per cluster, were formalised in a response-time-analysis model that ensures time-critical tasks are completed on time. The system of location-based ads, consisting of Apache Kafka and Apache Storm was successfully tested by the application of 1000 simulated events, and the worst case computation time of 1.523ms was within the 3,000ms time budget for the travel time between RSUs, proving that the architecture meets the timing requirement with a great margin.

These results come with two caveats. First, the current implementation did not consider privacy; the advertising data supplied by the vendors would need access control, as well as anonymisation

measures before it could be released in a real-world environment, and personal vehicle information would also require such measures. Second, the recommendation logic itself is relatively simple, rule-based, and a production system would benefit from more sophisticated targeting that would be specific to the different vendors and would be more sophisticated with richer media types like image and video ads.

In addition to both of these, there are three additional directions that are worth considering. Data privacy demands that only approved parties have access to the data sent by the vehicles; the data provided by each vehicle is sent to the RSU it is serviced by for processing. This pipeline needs to be secured against external attacks and malicious software, which is not assessed in the current prototype. For workloads with various latency-throughput trade-offs, explore other or alternative stream-processing tools beyond Apache Kafka and Apache Storm to improve data efficiency.

More broadly, in the future, advertisement recommendation could be expanded to include richer media (image and video), and the underlying transport could move to Ultra-Reliable Low-Latency Communication (URLLC) based on 5G, which would enable data rates of a gigabytes per second, bringing the latency ceiling down even more for time-critical IoV applications.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

REFERENCES

S. Yousefi, M. S. Mousavi, and M. Fathy, "Vehicular ad hoc networks (VANETs): challenges and perspectives," in 2006 6th International Conference on ITS Telecommunications, 2006, pp. 761–766.

S. Zeadally, R. Hunt, Y.-S. Chen, A. Irwin, and A. Hassan, "Vehicular ad hoc networks (VANETS): status, results, and challenges," Telecommunication Systems, vol. 50, pp. 217–241, 2012.

V. Kumar, S. Mishra, and N. Chand, "Applications of VANETs: present & future," Communications and Network, vol. 5, p. 12, 2013.

W. Wu, Z. Yang, and K. Li, "Internet of Vehicles and applications," in Internet of Things, Elsevier, 2016, pp. 299–317.

W. Xu, H. Zhou, N. Cheng, F. Lyu, W. Shi, J. Chen, et al., "Internet of vehicles in big data era," IEEE/CAA Journal of Automatica Sinica, vol. 5, pp. 19–35, 2017.

C. Huang, R. Lu, and K.-K. R. Choo, "Vehicular fog computing: architecture, use case, and security and forensic challenges," IEEE Communications Magazine, vol. 55, pp. 105–111, 2017.

C. Liu, K. Liu, S. Guo, R. Xie, V. C. Lee, and S. H. Son, "Adaptive offloading for time-critical tasks in heterogeneous internet of vehicles," IEEE Internet of Things Journal, vol. 7, pp. 7999–8011, 2020.

K. Sultan, H. Ali, and Z. Zhang, "Big data perspective and challenges in next generation networks," Future Internet, vol. 10, p. 56, 2018.

Z. W. Lamb and D. P. Agrawal, "Analysis of mobile edge computing for vehicular networks," Sensors, vol. 19, p. 1303, 2019.

Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "A survey on mobile edge computing: The communication perspective," IEEE Communications Surveys & Tutorials, vol. 19, pp. 2322–2358, 2017.

S. Wang, X. Zhang, Y. Zhang, L. Wang, J. Yang, and W. Wang, "A survey on mobile edge networks: Convergence of computing, caching and communications," IEEE Access, vol. 5, pp. 6757–6779, 2017.

M. Othman, S. A. Madani, and S. U. Khan, "A survey of mobile cloud computing application models," IEEE Communications Surveys & Tutorials, vol. 16, pp. 393–413, 2013.

S. Ancona, R. Stanica, and M. Fiore, "Performance boundaries of massive Floating Car Data offloading," in 2014 11th Annual Conference on Wireless On-demand Network Systems and Services (WONS), 2014, pp. 89–96.

C. Song, J. Wu, W.-S. Yang, M. Liu, I. Jawhar, and N. Mohamed, "Exploiting opportunities

in V2V transmissions with RSU-assisted backward delivery," in 2017 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), 2017, pp. 271–276.

M. Satyanarayanan, "Pervasive computing: Vision and challenges," IEEE Personal Communications, vol. 8, pp. 10–17, 2001.

J. Zhu, D. S. Chan, M. S. Prabhu, P. Natarajan, H. Hu, and F. Bonomi, "Improving web sites performance using edge servers in fog computing architecture," in 2013 IEEE 7th Int. Symp. Service-Oriented System Engineering, 2013, pp. 320–323.

M. Sharifi, S. Kafaie, and O. Kashefi, "A survey and taxonomy of cyber foraging of mobile devices," IEEE Communications Surveys & Tutorials, vol. 14, pp. 1232–1243, 2011.

B.-G. Chun and P. Maniatis, "Augmented smartphone applications through clone cloud execution," in HotOS, 2009, pp. 8–11.

K. Kumar, J. Liu, Y.-H. Lu, and B. Bhargava, "A survey of computation offloading for mobile systems," Mobile Networks and Applications, vol. 18, pp. 129–140, 2013.

H. Lasi, P. Fettke, H.-G. Kemper, T. Feld, and M. Hoffmann, "Industry 4.0," Business & Information Systems Engineering, vol. 6, pp. 239–242, 2014.

M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. H. Katz, A. Konwinski, et al., "Above the clouds: A Berkeley view of cloud computing," Technical Report UCB/EECS-2009-28, EECS Department, University of California, Berkeley, 2009.

Y. C. Hu, M. Patel, D. Sabella, N. Sprecher, and V. Young, "Mobile edge computing—A key technology towards 5G," ETSI White Paper, vol. 11, pp. 1–16, 2015.

N. Abbas, Y. Zhang, A. Taherkordi, and T. Skeie, "Mobile edge computing: A survey," IEEE Internet of Things Journal, vol. 5, pp. 450–465, 2017.

K. Bilstrup, E. Uhlemann, E. G. Strom, and U. Bilstrup, "Evaluation of the IEEE 802.11p MAC method for vehicle-to-vehicle communication," in 2008 IEEE 68th Vehicular Technology Conference, 2008, pp. 1–5.

H. Moustafa and Y. Zhang, Vehicular Networks: Techniques, Standards, and Applications. Auerbach Publications, 2009.

T. T. Dandala, V. Krishnamurthy, and R. Alwan, "Internet of Vehicles (IoV) for traffic management," in 2017 Int. Conf. Computer, Communication and Signal Processing (ICCCSP), 2017, pp. 1–4.

F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog computing and its role in the internet of things," in Proc. 1st Edition MCC Workshop on Mobile Cloud Computing, 2012, pp. 13–16.

C. B. Math, "Decentralized congestion control for reliable vehicular communication," PhD thesis, 2019.

M. Satyanarayanan, P. Bahl, R. Caceres, and N. Davies, "The case for VM-based cloudlets in mobile computing," IEEE Pervasive Computing, vol. 8, pp. 14–23, 2009.

M. Amadeo, C. Campolo, and A. Molinaro, "Information-centric networking for connected vehicles: a survey and future perspectives," IEEE Communications Magazine, vol. 54, pp. 98–104, 2016.

P. Basanta-Val, N. C. Audsley, A. J. Wellings, I. Gray, and N. Fernández-García, "Architecting time-critical big-data systems," IEEE Transactions on Big Data, vol. 2, pp. 310–324, 2016.

X. Wang, Z. Ning, and L. Wang, "Offloading in Internet of vehicles: A fog-enabled real-time traffic management system," IEEE Transactions on Industrial Informatics, vol. 14, pp. 4568–4578, 2018.

J. Chen, B. Liu, L. Gui, F. Sun, and H. Zhou, "Engineering link utilization in cellular offloading oriented VANETs," in 2015 IEEE Global Communications Conference (GLOBECOM), 2015, pp. 1–6.

N. Wang and J. Wu, "Opportunistic WiFi offloading in a vehicular environment:

Waiting or downloading now?," in IEEE INFOCOM 2016, 2016, pp. 1–9.

S. Otsuki and H. Miwa, "Contents delivery method using route prediction in traffic offloading by V2X," in 2015 Int. Conf. Intelligent Networking and Collaborative Systems, 2015, pp. 239–245.

E. A. Lee, "Cyber physical systems: Design challenges," in 2008 11th IEEE Int. Symp. Object and Component-Oriented Real-Time Distributed Computing (ISORC), 2008, pp. 363–369.

I. Parvez, A. Rahmati, I. Guvenc, A. I. Sarwat, and H. Dai, "A survey on low latency towards 5G: RAN, core network and caching solutions," IEEE Communications Surveys & Tutorials, vol. 20, pp. 3098–3130, 2018.

P. Alexander, D. Haley, and A. Grant, "Cooperative intelligent transport systems: 5.9-GHz field trials," Proceedings of the IEEE, vol. 99, pp. 1213–1235, 2011.

A. Festag, "Cooperative intelligent transport systems standards in Europe," IEEE Communications Magazine, vol. 52, pp. 166–172, 2014.

P. Basanta-Val, N. C. Audsley, A. J. Wellings, I. Gray, and N. Fernández-García, "Architecting time-critical big-data systems," IEEE Transactions on Big Data, vol. 2, pp. 310–324, 2016.

Ali, Qazi Ejaz, et al. "ASPA: Advanced strong pseudonym based authentication in intelligent transport system." PloS one 14.8 (2019): e0221213.