Article

SCARE: A Novel Switching and Collision Avoidance pRocEss for Connected Vehicles Using Virtualization and Edge Computing Paradigm

Mauro Tropea ¹ , Floriano De Rango ^{1,*}, Nicolas Navigato ², Luigi Bitonti ² and Francesco Pupo ¹

¹ Department of Informatics, Modeling, Electronics and System Engineering (DIMES), University of Calabria, Via P.Bucci 39/c, 87036 Rende, CS, Italy; mtropea@dimes.unical.it (M.T.); pupo@dimes.unical.it (F.P.)

² NTT Data, c.da Cutura, Via Spagna 50, 87036 Rende, CS, Italy; nicolas.navigato@nttdata.com (N.N.); luigi.bitonti@nttdata.com (L.B.)

* Correspondence: derango@dimes.unical.it (F.D.R.); Tel.: +39-0984-494-706

Abstract: In this paper, some collision avoidance systems based on MEC in a VANET environment are proposed and investigated. Micro services at edge are considered to support service continuity in vehicle communication and advertising. This considered system makes use of cloud and edge computing, allowing to switch communication from edge to cloud server and vice versa when possible, trying to guarantee the required constraints and balancing the communication among the servers. Simulation results were used to evaluate the performance of three considered mechanisms: the first one considering only edge with load balancing, the second one using edge/cloud switching and the third one using edge with load balancing and collision avoidance advertising.

Keywords: VANET; Mobile Edge Computing (MEC); Connected Vehicle (CV); virtualization; collision avoidance; switching mechanism

Citation: Tropea, M.; De Rango, F.; Navigato, N.; Bitonti L.; Pupo F. SCARE: A Novel Switching and Collision Avoidance pRocEss for Connected Vehicles Using Virtualization and Edge Computing Paradigm. *Sensors* **2021**, *21*, 3638. <https://doi.org/10.3390/s21113638>

Academic Editors: Fatih Kurugollu, Rasheed Hussain, Syed Hassan Ahmed, Farhan Ahmad and Chaker Abdelaziz Kerrache

Received: 6 April 2021
Accepted: 19 May 2021
Published: 24 May 2021

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Copyright: © 2021 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

Mobile Edge Computing (MEC) [1] is a distributed and decentralized Information Technology (IT) architecture able to elaborate and store critical data locally and transmit all received/elaborated data to a data center on the cloud. In this way, the elaboration, storage and networking information is closer to the devices that generate data. The typical use case is that of Internet of Things (IoT) [2] devices and implementations and the Connected Vehicles (CVs) field [3], which often faces latency problems and lack of bandwidth and reliability, not addressable through the conventional cloud model. Here, the MEC architecture is able to reduce the amount of data to be sent to the cloud, processing critical data, sensitive to latency, at the origin point, via smart devices, or sending it to an intermediate server, located nearby; the less 'time-sensitive' data can instead be transmitted to the cloud infrastructure or data center, to allow more complex processing, such as big data analysis. In this context, particular attention has to be addressed to congestion reduction (e.g., [4,5]), routing (e.g., [6]) and mobility prediction (e.g., [7]).

In this paper, we consider two important aspects of a vehicular network [8]: one regards handover issues for guaranteeing service continuity to vehicles inside the considered area taking into account also the Road Side Unit (RSU) load balancing and the other aims to provide a mechanism able to avoid collision among vehicles. Both considered aspects are based on the use of the MEC paradigm [9] in order to consider a critical point in VANET environment: the latency. In fact, in a vehicular scenario, the number of events that happens in a second can be very high, and then the possibility of communicate with devices near the road can contribute to avoid vehicle crashes. In particular, in this work, we propose a solution that considers RSUs implementing the MEC paradigm, able to communicate with each other and vehicles in order to allow a ready knowledge of any dangerous event, guaranteeing service continuity and collision avoidance.

RSUs are designed as elements able to perform resource management in the network and thus guarantee correct functionality to vehicles. Two different service continuity mechanisms were implemented in Java language in the RSU: the first one is based on the threshold Received Signal Strength (RSS) technique [10] and the second one tries to exploit the cell breathing mechanisms typical of UMTS networks [11]. Moreover, together with these mechanisms, a collision avoidance system is proposed that can utilize RSUs and the communication between vehicles and RSUs based on new connectivity technology (i.e., 5G) in order to make safer the drivers on the roads through assisted driving systems. A study on the human reaction time is shown in order to analyze how different human reaction times can affect the human behavior on the road bringing to a different impact on the number of collisions.

Furthermore, it is possible to use virtualization [12] with Docker containers, for developing lightweight and portable servers [13], and the container orchestrator Kubernetes [14]. Moreover, to manage efficiently the server containers, they are grouped in a Cluster orchestrated by Kubernetes software platform [15]. For the management of vehicular traffic, constrained requirements in terms of latency, safety and fault tolerance are required. For this reason, MEC is preferred to cloud architecture [9]. Furthermore, we choose to use containers instead of virtual machines, due to their lightness and easy management [16].

This paper is an extension of two previous works [17,18], concerning the study of the possibility of switching between cloud and edge servers in the road context and how this can affect the collision impact on a VANET environment. In particular, it unifies these papers by providing a more detailed description of the proposed ideas and showing how human reaction times are related with vehicles' collisions. Moreover, a unified mechanism called SCARE (a novel Switching and Collision Avoidance pRocEss) is proposed to improve load balancing between cloud and edge servers and propagating in real-time collision avoidance warnings at the EDGE.

The paper is organized as follows. A literature overview on CV and edge computing and collision avoidance mechanisms in VANET is presented in Section 2. The reference scenario is presented in Section 3. Section 4 describes the design of the critical parameters: reaction and latency times. The proposed services continuity mechanisms are described in Section 5. A description of the collision avoidance mechanism is provided in Section 6. Some details on the software implementation are described in Section 7. Finally, the performance evaluation and conclusion are presented, respectively, in Sections 8 and 9.

2. Related Work

In this section, we provide a quick overview on the state of art for edge computing and the collision avoidance mechanisms related to VANET is summarized in this section. Many research papers deal with the new paradigm called Internet of Vehicles (IoVs); for example, Ji et al. [19] presented a survey on IoV, highlighting architectural and application aspects. Moreover, blockchain technology has been attracting great interest in ITS environment due to its capability of achieving a decentralized system [20].

2.1. Connected Vehicles and Edge in VANET

The term Connected Vehicle (CV) is related to a new concept which sees all users involved (not only cars but also pedestrians, bikes, traffic lights, infrastructures, etc.) able to communicate with each other in order to exchange information for increasing road safety. It constitutes one of the new frontiers of mobile technology. It poses important challenges in terms of number of users managed, latency and bandwidth. The new technologies related to 5G and the possibility of transferring part of the intelligence present in the network to its borders (MEC) will be a key enabling component. Recently, many technological innovations have been proposed to realize VANET technology by using Vehicle-to-Cloud (V2C), Vehicle-to-Infrastructure (V2I) and Vehicle-to-Vehicle (V2V) communications. In the literature, different works exist that deal with these new topics (Connected Vehicle (CV),

Mobile Edge Computing (MEC), etc.). Researchers have proposed many solutions for the challenges that these new technologies arise.

In [21], the authors presented a framework for delivering data/service to the CVs. Using the optimization framework, they studied costs and parameters for using optimal bandwidth and vehicles flow in a data/service delivery system.

In [22], the authors studied the traffic volume in vehicular environment exploiting edge computing assisted 5G-VANET. Their proposal concerns a hierarchical architecture for distributing data using a scheduling scheme able to efficiently schedule vehicular content distribution. Their simulation results demonstrate the goodness of the proposed scheme.

In [23], the authors studied the performance of a system considering different protocols such as MQTT, HTTP and CoAP in a vehicular context showing how the performance is at the edge or cloud network infrastructure. In particular, they provided empirical insights on the advantages that an edge–cloud-based service provisioning can provide in comparison to a centralized cloud-based approach.

In [24], the authors presented a survey on the designs of autonomous driving edge computing systems, with particular attention to the security issues in autonomous driving and on how edge computing can resolve these issues.

In [25], an exploration of edges for connected and autonomous driving is shown. The authors suggested performing data elaboration closer to vehicles, thus guaranteeing a reduced end-to-end latency. They showed the goodness of using a mobile/multi-access edge computing (MEC) over 5G telecommunications.

In [26], the authors proposed a context-aware edge-based packet forwarding scheme for vehicular networks employing on a fuzzy logic-based edge node selection protocol to find the best edge nodes in a decentralized manner. They used a reinforcement learning algorithm to optimize the last two-hop communications in order to improve the adaptiveness of the communication routes. They evaluated their proposal through extensive simulation experiments.

2.2. Collision Avoidance and Edge in VANET

Many papers exist in the literature on topics that regard the automotive environment based on the distribution and dissemination of data between vehicles in order to solve many different situations in the road context. This data sharing is fundamental for all those emergency situations in which the propagation of vehicles information can play a key role for solving different issues and avoiding possible car collisions [27,28]. Many of these works exploit the edge paradigm to guarantee better performance in vehicular environment thanks to the low-latency and high-bandwidth characteristics of this approach [29,30].

In [31], the authors reviewed a series of models and algorithms proposed for avoiding collision events in the roads. In particular, they focused on two types of common accidents: rear-end crashes and intersection collisions. They highlighted the main technologies used in this field: cloud computing, neural networks, 4G and 5G.

In [32] the authors dealt with the autonomous vehicle issues that raise the problem of collisions and how to avoid them. They proposed a decentralized framework named RACE (Reinforced Cooperative Autonomous Vehicle Collision Avoidance) that exploits a multi-agent deep reinforcement learning model to be used during autonomous driving. They implemented the RACE framework in the widely used TORCS simulator and conducted various experiments to show the goodness of their proposal by comparing their approach with existing autonomous driving systems.

In [33,34], the authors proposed a MEC approach based on a cooperative Collision Avoidance service, called MECAV, which tries to preview road hazards through the use of vehicles sensors equipment. Their collision avoidance service is allocated in the MEC infrastructure, which receives and processes information sent by vehicles and selectively alerts other vehicles of a possible emergency event on the road. They implemented a demo of the MECAV system, composed of two vehicles, two WiFi access points and

the CAV service running on two MEC servers to validate the proposal architecture and functionalities.

Other studies about the use of MEC and collision avoidance system can be found in [35,36], where the authors proposed a MEC-based collision avoidance system able to protect not only vehicles but also vulnerable users, such as pedestrian, from possible collision events. Their experimental results provide indications on the reliability of algorithms for two types of road events: car-to-car and car-to-pedestrian collisions. They provided analyses for scenarios where a human driver is considered and automated vehicles (characterized by faster reaction times) populate the streets.

Another important study about collision was presented by Abou El Assad et al. [37], who proposed a fusion framework based on machine learning models able to provide a collision avoidance system capable of performing a prediction in real-time traffic crash. In particular, the framework takes into account a dataset including vehicle kinematics, driver behavior and weather conditions and combines several classifiers in order to make the final prediction. In [38], the authors proposed a new risk assessment-based decision-making algorithm able to guarantee collision avoidance in multi-scenarios for autonomous vehicles. In particular, the proposal taking into account driving style preferences (e.g., aggressive or conservative) and uses Carla simulator for evaluating and validate their approach.

2.3. Main Paper Contributions

In this work, we focus our attention on the a vehicular scenario, where two mechanisms are proposed for allowing vehicles switching among RSUs in order to guarantee RSUs load balancing and for making secure the considered system trying to avoid as many collisions as possible in the considered area. In particular, this work is an extension of two previous ones [17,18] published in conferences.

The first paper [17] deals with the problem of guaranteeing vehicles service continuity throughout the use of MEC technology. It provides two mechanisms of vehicles handover in the covered area showing how a better distribution of vehicles under the coverage of each RSU can provide better performance of the system at the expense of system latency time.

The second paper [18] presents a proposal of a collision avoidance mechanism that, exploiting edge technology, is able to switch between cloud server when the required constraints allow this type of communication characterized by high latency time. It shows how the use of 5G communication technology guarantees better performance of the system thanks to the low latency guaranteed by this technology.

In this paper, starting from these two works, we provide a merged study that analyzed the system behavior considering both the switching mechanism for vehicles handover process and the collision avoidance mechanism for the safety of drivers in a VANET environment balancing the communication among the cloud and edge servers. In particular, the main contributions of this work are summarized as follows:

- We propose two service continuity mechanisms able to guarantee in the handover phase the continuity of communication of all vehicles inside the considered VANET scenario. We show that the use of a cell breathing-like approach is able to better guarantee a load balancing of RSU covered areas.
- We propose a collision avoidance mechanism able to guarantee a low number of collision inside the VANET scenario, providing a method for utilizing a cloud or an edge RSU server on the basis of the latency time computed as human reaction plus the communication latency time.
- We analyze the contribution of the human reaction time in the vehicles scenario showing how times of reaction subdivided in three different categories can affect the response of the driver in the road context.
- The goodness of the proposed approach is proved by a set of simulation results in terms of latency time, switching number, collision number, collision percentage and server utilization percentage.

- We compare an edge-based mechanism versus the SCARE proposal in terms of collisions percentage and server utilization rate, in order to show how an approach taking into account load balancing and real-time advertising propagation is the right compromise in a emergency events scenario.

3. Reference Scenario

In this work, the considered scenario is a multi-layered architecture composed of three different parts: the vehicle layer, where there are vehicles traveling on the road; the edge layer, where there are RSU devices able to cover a specific portion of the map; and the cloud layer, where there is a centralized server able to perform different complex tasks (Figure 1). The roadside RSUs communicate with each others through the backbone infrastructure. Each RSU is equipped with an on-board processor for elaborating data, and for making decisions on the vehicles management. By taking advantage of 5G networks, latency is reduced to a few milliseconds [39]. A brief characteristics comparison between MEC and Cloud computing is shown in Table 1 [23].

Figure 1. Reference Scenario.

Table 1. Comparison between MEC and Cloud computing in VANET.

	MEC	Cloud
location	near users	remote location
latency	low	high
mobility support	high	limited
decision making	local	remote
communication	real-time	no real-time
context awareness	yes	no
computing capability	medium/low	high

When a problem occurs, the RSU evaluates whether to propagate messages arriving from previous RSUs towards the next ones or to stop forwarding because it is not necessary. All vehicles registered with the RSU are tracked; if the last vehicle has to be notified of a problem, then the message must be propagated to the adjacent RSU; instead, if the last vehicle is too distant from the event, then the message is not forwarded.

In this work, without loss of generality, it is assumed that all vehicles travel in one direction. To facilitate the simulation and make it faster, the two servers are supposed to be at a distance of 100 m from each other and their coverage range is 90 m. In this way, it is possible to know what happens during switching mechanism since the vehicles remain in the area covered by both nodes for a sufficiently long time (see Figure 1). Figure 1 shows three different zones: *Zone 1* is covered by first RSU; *Zone 2* is where a mobile device can be

covered by both RSUs, here the switching mechanism takes place; and *Zone 3* is covered by only second RSU.

Messages in the Scenario

In the considered scenario, the main actors are RSUs and vehicles that are able to exchange information by the use of specific messages. Five different typologies of messages exchanged between vehicles and servers were defined. The messages format is depicted in Figure 2. It is possible to view the main fields that the message transports in order to allow the correct functionality of proposed switching mechanisms.

Figure 2. Message format exchanged among devices.

In Table 2, a description of different message typologies used for the device communication is provided.

Table 2. Messages description.

Message Type	Message Description
Type 0	Periodic message that each vehicle sends to its reference server to update it on its speed and position
Type 1	Message exchanged among RSU devices and among RSUs and vehicles for notifying particular events
Type 2	Sent from the vehicle to the server to notify switching completion
Type 3	Message reply to a Type 2 message to notify the vehicle dissociation
Type 4	Used to communicate the resizing of the coverage area of the server

The messages exchange can allow a switching process that causes the change of RSU covered area. In this work, we propose two different switching mechanisms: one based on RSS threshold and the other one on a cell breathing-like approach typical of UMTS networks. We describe the two mechanisms and compare them to make some considerations about them.

4. Critical Parameters Design

For *latency*, in our work, we refer to the connectivity technology used in the vehicular context. In particular, we considered a 5G technology used by vehicles for communicating with the RSU placed on the road side (MEC server) or with the cloud server. The choice of communication technology clearly determines a greater or smaller latency time to take into account in the considered experiments.

4.1. Reaction and Latency Time

In road terminology, the term reaction time usually indicates the space and time, simultaneously necessary, for the driver of a vehicle to perceive a situation to be faced (in most cases, it is a danger that arises suddenly), realize perception in the mind and react with direct voluntary actions, with sufficient awareness of them, to a specific purpose; these actions and reactions, which take place in the context of the reaction time, are of a sensory, psychic, muscular and mechanical nature.

The duration of the reaction time is variable in relation to the psycho-physical conditions of the driver, the desired maneuver, the road and environmental conditions, the degree of efficiency of the vehicle and its speed.

The duration of the reaction time can be considered to be between approximately 0.75 and 1.50 s; in particular cases, it can be even greater [40,41]. In fact, from a general point of view, the duration of time t_r can be considered variable in relation to the following typical conditions:

- perception and reaction of *Fast Reaction (FR)*, $t_r = 0.75$ s;
- perception and reaction of *Normal Reaction (NR)*, $t_r = 1$ s; and
- perception and reaction of *Slow Reaction (SR)*, $t_r = 1.50$ s.

From when the vehicle driver sees an obstacle, he takes 1 s before starting to brake and travels a reaction space $\Delta s_r = v \cdot (1 \text{ s})$ which depends on the speed v at which he proceeds. If the obstacle is less than Δs_r meters from the car, the driver does not even have time to start braking and collides with the obstacle with speed v . In Table 3, different speeds and spaces of reaction are shown.

Table 3. Speed and space of reaction.

v (km/h)	v (m/s)	Δs_r (m)
25	7	7
50	14	14
70	19	19
90	25	25
130	36	36

It is as if the vehicle is Δs_r meters longer and its length increases with speed.

4.1.1. Braking Distance

The braking distance is the distance a vehicle travels between the start of deceleration and stopping. In the fairly realistic hypothesis that the deceleration produced by the brakes is constant, the motion of the vehicle is uniformly accelerated. The initial speed v_0 , the final speed $v = 0$ m/s, the acceleration $-a$ and the braking distance Δs_f are linked by the relationship (1):

$$\Delta s_f = \frac{1}{2} \frac{v^2 - v_0^2}{a} = \frac{1}{2} \frac{v_0^2}{a} \quad (1)$$

The braking distance depends on the condition of the vehicle and the road surface, which determine the value of the deceleration $-a$. However, the important thing is that the braking distance increases with the square of the speed. For a car in good condition, on a road with medium grip, the braking distances are very similar to the values indicated in Table 4 and shown in Figure 3.

Table 4. Speed and braking distance.

v (km/h)	Δs_f (m)
25	4
50	16
70	31
90	52
130	110

Figure 3. Braking distance (m).

4.1.2. Safety Distance

The safety distance is the distance that a vehicle must maintain from the one in front of it in order to stop without hitting it. The safety distance Δs_s is the sum of the reaction space and the braking distance:

$$\Delta s_s = \Delta s_r + \Delta s_f \quad (2)$$

For example, at a speed of 90 km/h, we have: $\Delta s_s = 25m + 52m = 77m$.

4.1.3. Latency Time

The latency time concerns the communication time with cloud or edge server. The vehicle is able to decide if it should communicate with one of the two servers on the basis of the traffic condition in order to guarantee avoiding collisions. It is expressed in milliseconds (ms) and depends on the distance between vehicle and server location. The cloud server is remotely placed, while the MEC server is located in the road side unit (RSU) on the road side and, thus, near the vehicle and able to guarantee a lower latency in the communication. In our study, the latency time was set to 1 ms, considering as communication technology 5G [42,43]. In particular, this value was set equal for the communication with both server typologies.

4.2. Collision Condition Computing

To know latency and reaction times to be used for the collisions avoidance phase, first the design of some system parameters that characterize the considered scenario need to be accounted.

In the considered system, the vehicle is identified by a number called identifier (ID). The first vehicles of the queue is indicated with $ID = 0$, while the vehicle behind has identifier $ID = 1$. The superscript in our formulation indicate the vehicle identifier. Moreover, before describing the mathematical formulation of the considered motion equations, whose symbols are defined in Table 5, we considered some assumptions:

- constant speed of vehicles on the road “ v ”;
- same deceleration value “ a ”; and
- distance d among vehicles.

Considering that suddenly the first vehicles brakes abruptly and indicating with t_{lr} the latency time as sum of latency and reaction times needed to the vehicle behind to start braking, the following motion equation for vehicle with $ID = 0$ at $t_0 = 0$ is considered:

$$\begin{cases} x_0^0 - \frac{1}{2}at^2 + v_0^0t & t < \frac{v}{a} \\ x_0^0 + \frac{1}{2}\frac{(v_0^0)^2}{a} & t \geq \frac{v}{a} \end{cases} \quad (3)$$

Instead, for vehicle with $ID = 1$ at $t_0 = 0$, the motion equation is as follows:

$$\begin{cases} x_0^1 + v_0^1t & t < t_{lr} \\ x_0^1 + (v_0^1 + at_{lr})t - \frac{1}{2}a(t^2 + t_{lr}^2) & t \geq t_{lr} \end{cases} \quad (4)$$

Under the assumption that, due to an unexpected event, the first vehicle $ID = 0$ suddenly brakes and the vehicles path are the same, we can calculate the collision time between the two vehicles as follows:

$$x_0^0 + \frac{1}{2}\frac{(v_0^0)^2}{a} = x_0^1 + (v_0^1 + at_{lr})t - \frac{1}{2}a(t^2 + t_{lr}^2) \quad (5)$$

By computing, we obtain the following second-degree equation:

$$\frac{1}{2}at^2 - (v_0^1 + at_{lr})t + \frac{1}{2}at_{lr}^2 + d + \frac{1}{2}\frac{(v_0^0)^2}{a} = 0 \quad (6)$$

This equation has solutions when the delta (Δ) is greater than 0, that is:

$$\Delta = (v_0^1 + at_{lr})^2 + 4\left(\frac{1}{2}a\right)\left(\frac{1}{2}at_{lr}^2 + d + \frac{1}{2}\frac{(v_0^0)^2}{a}\right) > 0 \quad (7)$$

$$\implies t_{lr} > \frac{d}{v}$$

If Equation (7) is satisfied, that is the t_{lr} time is greater than the ratio between considered vehicles distance and speed, we can consider that a collision happens between the vehicles when the first one suddenly brakes.

Table 5. Symbols description.

Symbol	Description
ID	Vehicle identifier
v	Vehicle speed
a	Vehicle acceleration/deceleration
d	Distance among vehicles
t_{lr}	Latency plus reaction time

5. Services Continuity Proposed Mechanisms

In this section, two proposed switching mechanisms for providing vehicles' services continuity are described. The first one considers the RSS threshold value in order to determine the time of switching towards the new RSU device coverage. The second one, on the basis of a cell breathing-like approach typical of a UMTS network, is able to perform a switching mechanism with load balancing for avoiding overload of RSU devices.

5.1. RSS Threshold Mechanism

This mechanism is based on messages received by RSU from vehicles under its coverage and on RSS value in order to perform the switching between the RSU stations [44]. It is depicted in Algorithm 1.

At the beginning, the RSU must know some information about RSU neighbors in order to communicate with them. Figure 1 shows that a vehicle inside *Zone 2* is able to receive a signal from RSU1 while in *Zone 3* it receives only the signal from RSU2.

Shortly, the RSU behavior:

- Upon receipt of a *Type 0* message, the RSU receives this message from the vehicle and becomes aware of a new vehicle in its area coverage. Thus, if an alert event happens, it can warn all coverage vehicles near this event and other RSUs with a *Type 1* message, informing about emergency situation.
- Upon receipt of a *Type 1* message, the RSU receives this message from the RSU and is aware of emergency situation and must check the presence of vehicles under its control near the vehicle braking sharply in order to avoid possible accident situations.
- Upon receipt of a *Type 2* message, the RSU receives this message from the vehicle and learns about switching completion by vehicle passed under its coverage. As a consequence, the RSU sends a *Type 3* message, containing vehicle ID, to the adjacent RSU to notify this change. Thus, the RSU can store this new condition in its database and no messages are more exchanged with this vehicle.

Clearly, this mechanism does not take into account load balancing behavior.

Algorithm 1: RSS Control Mechanism.

```

1 for each received message do
2   if messageType==Type 0 then
3     save vehicle position;
4     if emergencyEvent then
5       send messageType 1;
6   else if messageType==Type 1 then
7     send messageType 1;
8   else if messageType==Type 2 then
9     notifyHandover;
10  else if messageType==Type 3 then
11    deleteDBEntry;
12 end

```

5.2. Cell Breathing-Like Mechanism

An enhanced mechanism was considered in RSUs to improve the switching mechanism modifying the RSU behavior in the coverage intersection zone (*Zone 2*). The enhancement is based on the cell breathing concept [11], typical of UMTS networks. The idea is to perform a sort of load balancing into the RSU devices, subdividing their load in terms of managed vehicles. Three new data structures were introduced:

1. *VehiclesCommonZone*: It stores the number of vehicles in *Zone 2*, where potentially both RSU devices are able to manage vehicles.
2. *BlackList*: It manages vehicles that, even if they are still in the coverage area of a RSU, this one does not manage these vehicles because assigned to other RSU.

3. *VehiclesToControll*: It stores vehicles that have been assigned to the new RSU.

Upon receiving a message from a vehicle, the RSU verifies whether its ID is on its vehicles list:

- Vehicle not in the list: The RSU discards the message without further analysis since the vehicle is managed by another RSU.
- Vehicle in the list: The RSU analyzes the position of the vehicle and controls if it can potentially be transferred to another RSU by controlling position and coverage range.

Once the list has been updated, the RSU controls the maximum number of connections it can manage for vehicles belonging to *Zone 2*. Upon exceeding the limit, the RSU virtually decreases its coverage passing a vehicle control to the adjacent RSU, so to relieve its computational load, by sending a *Type 4* message. Thus, the new RSU can store vehicle ID in its database. This enhanced mechanism behavior, using the cell breathing-like paradigm, and its time diagram are reported in Algorithm 2 and Figure 4, respectively.

Algorithm 2: Cell breathing-like Control Mechanism.

```

1 for each received message do
2   if messageType==Type 0 then
3     if is the vehicle under RSU management? then
4       save vehicle position;
5       if maxConnection limit reached then
6         vehicleManagmentTransfer;
7       if emergencyEvent then
8         send messageType 1;
9     else
10      discard message;
11   end
12   else if messageType==Type 1 then
13     send messageType 1;
14   else if messageType==Type 2 then
15     notifyHandover;
16   else if messageType==Type 3 then
17     deleteDBEntry;
18   else if messageType==Type 4 then
19     insertDBEntry for vehicle;
20 end

```

Figure 4. Time diagram of cell breathing-like mechanism.

Figure 5 shows the different vehicles management with the two different proposed approaches.

Figure 5. Switching mechanism: (a) without cell breathing-like approach; and (b) with cell breathing-like approach.

6. Collision Avoidance Proposed Mechanism

In the following section, the proposed collision avoidance mechanism is described showing, through algorithm and flow chart representations, the behavior of servers and vehicles. Each vehicle is a CoAP client able to communicate with the server through the POST method exploiting Californium library [45]. The information about the vehicle status (i.e., ID, speed and position) is enclosed in a message. Furthermore, every command that vehicles receive from the server is managed by the simulator through a Java class, which implements vehicle behavior, as shown in Figure 6.

Dynamic Collision Avoidance Mechanism

The main characteristic of this mechanism is to avoid collision among vehicles taking into account the load of the edge and cloud server to avoid of overloading the edge server by using the cloud when possible. In particular, the decision of which server to be used is made by the vehicle on the basis of traffic conditions with the aim of reducing the number of collisions. In particular, this decision is based on the calculated distance among a considered vehicle and its neighbors.

A vehicle, for the choice of the appropriate server, considers the total latency and reaction time in a scenario where vehicles have the same speed and deceleration values, as shown in Section 4.1. If the ratio of the distance between two vehicles over the speed is less than cloud communication latency plus vehicle reaction, then the vehicle will communicate with the edge to avoid collision; otherwise, it chooses to communicate with the cloud. This dynamic algorithm behavior is described in Algorithm 3.

Algorithm 3: Dynamic Switching behavior.

```

1 vehicle identifies in front and following ones;
2 if is communication with edge essential for in front or following vehicle? then
3   | vehicle communicates with edge;
4 else
5   | vehicle communicates with cloud;
6 end

```

The main task of the MEC server implemented on the RSU is, upon receiving data from vehicles, to process this information in order to verify the presence of a danger or emergency braking by a vehicle, and then alert all following vehicles to avoid collisions. In

particular, the RSU calculates the difference between two consecutive speeds of a vehicle, and, if it is greater than a certain threshold, a dangerous condition is identified and an alert is sent timely to all vehicles in a certain range to avoid collision. The operating principle is described briefly in Algorithm 4.

Algorithm 4: MEC server behavior.

```

1 precSpeed;
2 currSpeed;
3 Threshold;
4 for each received message do
5   server decodes message;
6   if ( $precSpeed - currSpeed > Threshold$ ) then
7     server alerts other vehicles;
8     vehicle stops its run;
9 end

```

Figure 6. Flow chart of vehicle behavior.

7. Software Implementation

To implement switching processes and collision avoidance mechanism, it was necessary to create two servers that work in parallel, able to simulate two different RSUs. This was applied considering three virtual machines able to form the Kubernetes cluster. One machine was used as a master, the other two as worker nodes. The virtual machines were hosted on a normal PC, as shown in Figure 7.

Figure 7. Considered scenario.

7.1. Virtualization Environment: Docker and Kubernetes

The simulated system was realized through the use of normal PCs that simulate the RSU units running on Docker container [46] managed by Kubernetes [47] and the vehicular traffic by using SUMO simulator [48] (see Figure 7).

Finally, the implementation of the MEC server and, therefore, the communication from server to vehicle was created. The final system can simulate vehicles in traffic that communicate with each other and the server to manage a collision avoidance mechanism, thus guaranteeing security in the vehicular environment. Vehicles periodically, by a device installed on board, send messages that contain information about their status towards the RSU MEC server. In particular, the message contains the identification number (*ID*) and the position and speed of the considered vehicle. Messages exchanged between vehicles and RSUs are processed and used to understand when an emergency braking is taking place, and therefore anticipate any collisions. In the case of emergency event, the MEC server generates an alert message and immediately sends it to the following vehicles within a certain range. When the alert message reaches the vehicles, the braking system is automatically activated to stop the vehicle safely and avoid a collision. A dynamic switching algorithm was also implemented that allows the vehicle to instantly decide whether to send the message to the MEC or cloud server, in order to avoid of overloading the devices on the roadside when it is not necessary.

Docker is an open-source project that automates the process of deploying applications within software containers, providing an additional abstraction thanks to virtualization at the Linux operating system level [46]. It is based on isolation features of Linux kernel known as *cgroups* and *namespaces* that make possible the creation of independent *containers* that can coexist on the same Linux instance, with the great advantage of being more lightweight and simpler than virtual machines. The main characteristic of Docker software is the absence of an hypervisor layer, typical of the virtual machine, and thus the possibility of being executed directly on the host machine.

Kubernetes is an open-source container orchestration and management system [47]. Initially developed by Google, it is now maintained by the Cloud Native Computing Foundation. It works with many container systems, including Docker. It is composed of two different components called masters and nodes, also called workers. They coordinate

to run the workload on the servers that form a cluster controlled by Kubernetes that guarantees there is no downtime in a production environment.

The cluster is the core of Kubernetes, because it offers the ability to plan and run containers in a group of machines of any type: physical or virtual, on premise or in the cloud. It has at least one master and one worker node. The master is the component that provides tasks to worker nodes. The smallest and simplest object of a cluster is called a *pod*. A *pod* groups together containers that share storage and network resources and run on the same node. Another important object of Kubernetes is represented by *service*. It defines how to expose pods on an internal or external network. The service defines a name that is resolved by the cluster-internal DNS with one of the pods associated with it. The pods associated with the service are those that share the label defined by the service. By default a service is exposed inside a cluster, but it can also be exposed outside the cluster.

7.2. Vehicle Traffic Simulator

An urban mobility simulator, SUMO, was used to generate data from vehicles. SUMO is an open source software that allows creating various traffic scenarios [49]. Sumo can interact with the Java programs through *TraaS* library [50–52] making it easier to collect vehicle data at runtime. The simulator runs on a different PC, as shown in Figure 3. Eclipse was used as a development environment, able to interface with the simulator through the *SumoTraciConnection* library [48]. To carry out the communication between vehicles and MEC server and between vehicles and Cloud server, the CoAP protocol [53] was chosen using the Californium library written in Java [54]. Each vehicle was created as a CoAP client, and, periodically, makes requests to the MEC or Cloud server acting as CoAP server.

The vehicular traffic management is made by Sumo simulator where a stretch of road of *SS107 Silana-Crotonese*, in southern Italy, was considered and some XML files were created containing all information regarding the road portion such as the *IDs* of the two lanes of the roadway with the information of their direction, the maximum speed in m/s, etc. Moreover, the XML files contain the vehicles traveling information such as acceleration and deceleration values, *ID*, length, maximum speed in m/s and minimum gap with the previous vehicle, etc. Sumo allows importing the road network via *Open Street Map*, simplifying the process of developing the mobility model. This real world road network can be easily downloaded and modified and imported into the simulator through the *netconvert* command. In this way, a realistic vehicular traffic in the urban environment is generated. The Eclipse development environment is used to manage the simulator using code written in Java language and the library called *SumoTraciConnection* [48]. To start the simulation, a XML configuration file with the *.config* extension was created containing: the file related to the road route, the file related to vehicles and an additional file to display an alert on the map in case of a collision between two vehicles. Moreover, the configuration file contains a time section where the start and the end of the simulation with steps are specified. It is also possible to generate an output file with the simulation logs. The macro-diversity mechanism, typical of UMTS [55], is managed by simulator that allows vehicles to be connected simultaneously to different RSUs, so to guarantee vehicle connection also during the switching phase. Through the mechanism described above and the exchanged message, a vehicle is able to pass under the control of a new RSU.

7.3. Server Implementation

Servers were implemented in Java language and use CoAP protocol [53] to communicate with vehicles and each other. For the management of switching mechanism, two paradigms were used. The first one consists of the technique of RSS with threshold and the second one uses a combination of RSS with threshold and cell breathing. Moreover, to avoid collisions among vehicles, it processes the receiver data and notifies timely all vehicles of the dangerous event.

8. Performance Evaluation

In this section, we present the results achieved using the simulator environment described in Section 3 and represented in Figure 1. It consists of two RSUs, covering a portion of road in which vehicles travel, and a cloud server for managing data information sent by vehicles. Different simulations were performed varying some input parameters and showing results in terms of number of packets, latency in milliseconds, switching number, collision percentages and server utilization. In Table 6, the main parameters used in the simulation experiments are summarized.

Table 6. Simulation parameters.

Parameter	Value
vehicles number	1–10
vehicle distance d (m)	6, 6.5, 7, 7.5, 8, 8.5, 9, 9.5
vehicle speed v (km/h)	25–130
reaction time t_r (ms)	750–1000–1500
latency time t_l (ms)	1 (5G technology)

8.1. Switching Mechanism Analysis

In the following, we report graphics related to the proposed switching mechanisms showing the considered output parameters: packet number, latency time and switching number.

8.1.1. Packet Number Analysis

A first result is the packets number managed by two RSUs that communicate with vehicles on the road, without cell breathing-like approach (Figure 8a) and with cell breathing-like mechanism (Figure 8b). In the first case, the management of vehicles and packets is entirely covered by the first RSU. This is because vehicles are at a point where the RSS parameters is above the minimum allowed threshold, and therefore no switching mechanism is performed. In the second case, instead, even though the vehicle positions are the same, a better load distribution can be appreciated. This is because some vehicles, although still in the range of the first RSU, are forced to move under the management of the second RSU.

In this simulation campaign we considered five different vehicle values (2–6 vehicles). The load distribution can be seen with the introduction of the fifth vehicle. This is due to the maximum number of established connections set to three to guarantee a load balancing among six vehicles. When there are still four vehicles in the system, the first vehicle of the set of vehicles is under the management of the second RSU, while the other three are still under the management of the first one. Therefore, the second RSU has no vehicles to notify, but merely informs the first RSU of any warnings. By adding another vehicle, a configuration with two vehicles managed by the second RSU is possible to observe, while three of five remain under the management of the first one. In this case, it is possible to view a small load distribution, because the first RSU has to manage three vehicles while the second one has two vehicles.

(a)

(b)

(c)

(d)

(e)

Figure 8. (a) Packets managed without cell breathing technique; (b) packets managed with cell breathing technique; (c) latency without cell breathing technique; (d) latency with cell breathing technique; and (e) switching per time unit.

8.1.2. Latency Analysis

In this paragraph, the latency analysis is provided. In particular, we show the elapsed time between the detection of the danger event and its notification to the last vehicle considering both switching techniques.

Figure 8c shows the system behavior considering the RSS approach. It is possible to observe a mean latency time of about 40 ms with a standard deviation of about 4.9 ms. The value of 40 ms represents the time needed to: send a message by vehicle to RSU, process it by RSU and send alert to all other vehicles. We observed that the number of involved vehicles does not contribute to increase latency time because the experiments show that vehicles receive the alert almost simultaneously, with a time lag of less than 1 ms. Figure 8d, instead, shows the behavior considering cell breathing approach. In this case, RSU1 and RSU2 control half of vehicles in the map respectively. In these experiments, we measured a mean latency time of about 60 ms with a standard deviation of about 4.6 ms. We observed that this mean value is increased because the message has to cross another hop (RSU) for reaching vehicles.

The results show a significant increase in latency if the cell breathing-like technique is adopted. This is due to the fact that, to notify the last vehicle, the message must go from the vehicle that generated the event to RSU 2, from RSU 2 to RSU 1 and, finally, from RSU 1 to the last vehicle.

8.1.3. Switching Number Analysis

Finally, the last result for the switching analysis shows the number of switching per time unit considering into the map a number of five vehicles. The ratio between the switching number over the switching time has been calculated, obtaining the Switching Rate (SWR) per seconds (SWR/s). This value is affected by vehicles speed and by the distance from the vehicle ahead. Figure 8e shows the behavior of the switching mechanism. It is possible to observe that vehicles, increasing their speed, generate an increase number of switching given that they change more frequently their position. Moreover, it is possible to note that, at the same vehicle speed, the switching number per seconds increases when decreasing the distance between vehicles.

8.2. Collision Avoidance Analysis

In this section, we present the experiments' results of the proposed collision avoidance system. Experiments were performed considering a scenario composed of a different number of vehicles in the map traveling with a speed of 50 km/h, the same deceleration values, a distance d from each other and reaction times in the range shown in Table 6 [56].

We provide experimental results of a collision analysis showing first a study about the number of collision considering different human reaction times and then a study on the collisions percentage in the considered scenario varying the distance between vehicles and between servers and emergency event point.

8.2.1. Collision Number Analysis

To show the collision analysis considering human reaction times, in the following, the results about the number of collisions in the considered scenario varying the number of vehicles, the vehicles speed and traffic congestion are shown in Figure 9. The study aimed to indicate how the driving distraction, resulting in a slow human reaction, brings to a high number of collisions. In the next experiments, we considered different human reaction times and different numbers of vehicles with other parameters listed in Table 6.

Figure 9a gives some details about system behavior varying the vehicles number between 2 and 10. In particular, the analysis shows two important aspects: first how the collision number varies when in the map the number of vehicles increases and second how the human reaction time affects the increase of collisions. The so-called normal reaction time causes about 16% more collisions than the fast reaction time. Figure 9b gives other important indications about the number of collisions when the vehicle speed increases until

arriving at typical highway speeds. Finally, Figure 9c shows indications of how congestion affects the number of collisions and how the human reaction times are related to it.

Figure 9. Number of collisions varying: (a) vehicle number; (b) vehicle speed; and (c) traffic scenario.

8.2.2. Collision Percentage Analysis

The obtained results on collisions percentage are shown in Figure 10. Figure 10a shows details about the collisions percentage varying the vehicles distance. It is possible to note that with MEC server we have a lower percentage compared to the case with communication towards the cloud server. In addition, considering the same distance, MEC communication is able to guarantee better performance in terms of collisions percentage. Only by increasing the vehicles distance does the performance between server communications achieve similar values. Figure 10b shows the percentage of server (MEC and cloud) utilization evaluated considering three different conditions of congestion. The graphic shows how the aim of our proposal is respected; it allows, especially in certain vehicular traffic conditions, not to overload the MEC and communicate with the cloud server which has higher performance. In particular, when traffic is congested, the vehicles communicate more with the MEC server than with the cloud one since low latency times are required to avoid collisions. The figure shows that the MEC device is used a lot. If the traffic is not very congested, the utilization rate of the two servers is almost the same. On the other hand, considering non-congested traffic, each vehicle communicates with the cloud since

the MEC server is not essential. Therefore, in this last case, only the cloud is used, as shown in the figure.

Figure 10. (a) Collision percentage vs. vehicles distance; (b) server utilization vs. traffic congestion.

Figure 11. (a) Collision percentage vs. traffic congestion; and (b) server utilization percentage vs. traffic congestion.

8.3. Edge vs. SCARE Approach Comparison

In Section 8.1, we emphasize the approach using load balancing at the edge, as it can reduce the server CPU usage percentage at the expense of a higher latency in the communication and advertising propagation.

Moreover, in Section 8.2, we show that, in the case of emergency situations, the communication with the edge server is more suitable than cloud because it permits alower collisions number.

Now, in this section, we present the SCARE approach that takes advantage of both previous mechanisms: the load balancing at edge of Section 8.1, where we show how a load balancing approach permits to avoid overloading the RSU resources, and the EDGE mechanism in the advertising propagation of Section 8.2, where we prove how the use of edge server in the management of traffic with high priority such as alert messages highly sensitive to the latency communication is more useful then cloud one. The SCARE mecha-

nism joins the highlighted characteristics of the previous techniques in order to allow a reduced vehicles collisions number guaranteeing a better usage of edge servers' resources.

Thus, Figure 11 shows how the edge with cell breathing-like mechanism and warning propagation, the SCARE approach, can perform in comparison with the approach of the edge with warning propagation but without RSU load balancing, the EDGE one. It is possible to see that, in the case of congestion, the SCARE approach can improve the performance of the server load, but it can increase the collisions percentage because some extra message delay due to the additional involved RSU. So, the warning condition is not respected producing collisions among vehicles. However, when the road congestion is medium-low, "little congested scenario" in the figure, the SCARE approach improves its performance for both collision and server usage percentage, guaranteeing lower values for both output parameters. In the last case, where no congested roads are considered, both approaches perform in a similar way because the RSU load balancing is not activated and the warning can be propagated in the single RSU respecting the threshold time for the car collision.

9. Conclusions

In this paper, we propose a vehicular architecture where the MEC paradigm is used and implemented in RSU devices exploiting the low latency MEC characteristic. The implemented system is based on virtualization concepts and uses Docker and Kubernetes in the development of the reference scenario.

The paper proposes a twofold approach. Firstly, it deals with the study of the services continuity issue proposing two different approaches based on the handover mechanism. A mechanism based only on RSS value is compared with an approach that takes into account a cell breathing-like mechanism in order to show how it is possible to balance the RSUs load in the common zone, avoiding of overloading RSUs. The drawback is a latency that increases with the number of vehicles introducing delay in the messages delivery. Then, a collision avoidance mechanism based on cloud and edge paradigms is shown. The paper describes how to use and communicate with a cloud or edge server in order to exploit the low latency MEC characteristic, fundamental in a scenario where communication times are strict requirements. The proposal shows that vehicles can use the cloud server in normal condition and the edge one in emergency situation in order to avoid overloading the edge server normally equipped with small resource in comparison to the cloud. Finally, on the basis of these two approaches, we present a joint approach, called SCARE, that takes advantage of both mechanisms and is able to allow a reduction in the vehicles' collisions number and a better usage of the edge servers' resource. The performed analysis gives some indications about system behavior in terms of collision number and percentage. Further, the paper gives analyses of how human reaction times can affect the collision behavior, providing useful indications on three different human reaction responses and how the driving distraction results in a slow human reaction that brings to a high collisions number.

Author Contributions: Conceptualization and methodology, F.D.R., M.T. and F.P.; software and validation, L.B. and N.N.; resources and data curation, L.B. and N.N.; writing—original draft preparation, M.T. and F.P.; writing—review and editing, M.T., F.D.R. and F.P.; supervision, F.D.R.; All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Data sharing is not applicable to this article.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

CV	Connected Vehicles
CoAP	Constrained Application Protocol
FR	Fast Reaction
IoT	Internet of Things
IoV	Internet of Vehicles
IT	Information Technology
ITS	Intelligent Transportation Systems
MEC	Mobile Edge Computing
NR	Normal Reaction
RSS	Received Signal Strength
RSU	Road Side Unit
SR	Slow Reaction
SWR	Switching Rate
VANET	Vehicle Ad-hoc NETwork
V2C	Vehicle-to-Cloud
V2I	Vehicle-to-Infrastructure
V2V	Vehicle-to-Vehicle

References

1. Porambage, P.; Okwuibe, J.; Liyanage, M.; Ylianttila, M.; Taleb, T. Survey on multi-access edge computing for internet of things realization. *IEEE Commun. Surv. Tutor.* **2018**, *20*, 2961–2991. [\[CrossRef\]](#)
2. Shah, S.H.; Yaqoob, I. A survey: Internet of Things (IOT) technologies, applications and challenges. In Proceedings of the 2016 IEEE Smart Energy Grid Engineering (SEGE), Oshawa, ON, Canada, 21–24 August 2016; pp. 381–385.
3. Liu, J.; Liu, J. Intelligent and connected vehicles: Current situation, future directions, and challenges. *IEEE Commun. Stand. Mag.* **2018**, *2*, 59–65. [\[CrossRef\]](#)
4. Aleko, D.R.; Djahel, S. An efficient adaptive traffic light control system for urban road traffic congestion reduction in smart cities. *Information* **2020**, *11*, 119. [\[CrossRef\]](#)
5. Santamaria, A.F.; Fazio, P.; Raimondo, P.; Tropea, M.; De Rango, F. A new distributed predictive congestion aware re-routing algorithm for CO₂ emissions reduction. *IEEE Trans. Veh. Technol.* **2019**, *68*, 4419–4433. [\[CrossRef\]](#)
6. Fazio, P.; Sottile, C.; Santamaria, A.F.; Tropea, M. Vehicular networking enhancement and multi-channel routing optimization, based on multi-objective metric and minimum spanning tree. *Adv. Electr. Electron. Eng.* **2013**, *11*, 349–356. [\[CrossRef\]](#)
7. Fazio, P.; Tropea, M.; De Rango, F.; Voznak, M. Pattern prediction and passive bandwidth management for hand-over optimization in QoS cellular networks with vehicular mobility. *IEEE Trans. Mob. Comput.* **2016**, *15*, 2809–2824. [\[CrossRef\]](#)
8. Ghorri, M.R.; Zamli, K.Z.; Quosthoni, N.; Hisyam, M.; Montaser, M. Vehicular ad-hoc network (VANET): Review. In Proceedings of the 2018 IEEE International Conference on Innovative Research and Development (ICIRD), Bangkok, Thailand, 11–12 May 2018; pp. 1–6.
9. Pan, J.; McElhannon, J. Future edge cloud and edge computing for internet of things applications. *IEEE Internet Things J.* **2017**, *5*, 439–449. [\[CrossRef\]](#)
10. Ayyappan, K.; Dananjayan, P. RSS measurement for vertical handoff in heterogeneous network. *J. Theor. Appl. Inf. Technol.* **2008**, *4*, 989–994.
11. Thng, K.L.; Yeo, B.S.; Chew, Y.H. Performance study on the effects of cell-breathing in WCDMA. In Proceedings of the 2005 2nd International Symposium on Wireless Communication Systems, Siena, Italy, 5–7 September 2005; pp. 44–49.
12. Jain, N.; Choudhary, S. Overview of virtualization in cloud computing. In Proceedings of the 2016 Symposium on Colossal Data Analysis and Networking (CDAN), Indore, India, 18–19 March 2016; pp. 1–4.
13. Bernstein, D. Containers and cloud: From lxc to docker to kubernetes. *IEEE Cloud Comput.* **2014**, *1*, 81–84. [\[CrossRef\]](#)
14. Shah, J.; Dubaria, D. Building modern clouds: Using docker, kubernetes & Google cloud platform. In Proceedings of the 2019 IEEE 9th Annual Computing and Communication Workshop and Conference (CCWC), Las Vegas, NV, USA, 7–9 January 2019; pp. 0184–0189.
15. Bachiega, N.G.; Souza, P.S.; Bruschi, S.M.; de Souza, S.d.R. Container-based performance evaluation: A survey and challenges. In Proceedings of the 2018 IEEE International Conference on Cloud Engineering (IC2E), Orlando, FL, USA, 17–20 April 2018; pp. 398–403.
16. Doan, T.V.; Nguyen, G.T.; Salah, H.; Pandi, S.; Jarschel, M.; Pries, R.; Fitzek, F.H. Containers vs. Virtual Machines: Choosing the Right Virtualization Technology for Mobile Edge Cloud. In Proceedings of the 2019 IEEE 2nd 5G World Forum (5GWF), Dresden, Germany, 30 September–2 October 2019; pp. 46–52.

17. Bitonti, L.; Tropea, M. Implementation and Simulation of Handover Techniques to Guarantee Service Continuity through Microservices at Edge. In Proceedings of the 10th International Conference on Simulation and Modeling Methodologies, Technologies and Applications, SIMULTECH 2020, Paris, France, 8–10 July 2020; pp. 287–293. [\[CrossRef\]](#)
18. Nevigato, N.; Tropea, M.; De Rango, F. Proposal of an Automotive Collision Avoidance System based on Edge Computing. In Proceedings of the 10th International Conference on Simulation and Modeling Methodologies, Technologies and Applications, SIMULTECH 2020, Paris, France, 8–10 July 2020; pp. 128–135. [\[CrossRef\]](#)
19. Ji, B.; Zhang, X.; Mumtaz, S.; Han, C.; Li, C.; Wen, H.; Wang, D. Survey on the Internet of Vehicles: Network Architectures and Applications. *IEEE Commun. Stand. Mag.* **2020**, *4*, 34–41. [\[CrossRef\]](#)
20. Peng, C.; Wu, C.; Gao, L.; Zhang, J.; Alvin Yau, K.L.; Ji, Y. Blockchain for vehicular Internet of Things: Recent advances and open issues. *Sensors* **2020**, *20*, 5079. [\[CrossRef\]](#) [\[PubMed\]](#)
21. Gangadharan, D.; Sokolsky, O.; Lee, I.; Kim, B.; Lin, C.W.; Shiraishi, S. Bandwidth Optimal Data/Service Delivery for Connected Vehicles via Edges. In Proceedings of the 2018 IEEE 11th International Conference on Cloud Computing (CLOUD), San Francisco, CA, USA, 2–7 July 2018; pp. 106–113.
22. Luo, G.; Yuan, Q.; Zhou, H.; Cheng, N.; Liu, Z.; Yang, F.; Shen, X.S. Cooperative vehicular content distribution in edge computing assisted 5G-VANET. *China Commun.* **2018**, *15*, 1–17. [\[CrossRef\]](#)
23. Laaroussi, Z.; Morabito, R.; Taleb, T. Service provisioning in vehicular networks through edge and cloud: an empirical analysis. In Proceedings of the 2018 IEEE Conference on Standards for Communications and Networking (CSCN), Paris, France, 29–31 October 2018; pp. 1–6.
24. Liu, S.; Liu, L.; Tang, J.; Yu, B.; Wang, Y.; Shi, W. Edge computing for autonomous driving: Opportunities and challenges. *Proc. IEEE* **2019**, *107*, 1697–1716. [\[CrossRef\]](#)
25. Gillam, L.; Katsaros, K.; Dianati, M.; Mouzakitis, A. Exploring edges for connected and autonomous driving. In Proceedings of the IEEE INFOCOM 2018—IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), Honolulu, HI, USA, 15–19 April 2018; pp. 148–153.
26. An, C.; Wu, C.; Yoshinaga, T.; Chen, X.; Ji, Y. A context-aware edge-based VANET communication scheme for ITS. *Sensors* **2018**, *18*, 2022. [\[CrossRef\]](#) [\[PubMed\]](#)
27. Bujari, A.; Gottardo, J.; Palazzi, C.E.; Ronzani, D. Message Dissemination in Urban IoV. In Proceedings of the 2019 IEEE/ACM 23rd International Symposium on Distributed Simulation and Real Time Applications (DS-RT), Cosenza, Italy, 7–9 October 2019; pp. 1–4.
28. Santamaria, A.F.; Tropea, M.; Fazio, P.; De Rango, F. Managing emergency situations in VANET through heterogeneous technologies cooperation. *Sensors* **2018**, *18*, 1461. [\[CrossRef\]](#) [\[PubMed\]](#)
29. El-Sayed, H.; Chaqfeh, M. Exploiting mobile edge computing for enhancing vehicular applications in smart cities. *Sensors* **2019**, *19*, 1073. [\[CrossRef\]](#) [\[PubMed\]](#)
30. Rakesh, G.; Belwal, M. Vehicle collision avoidance in a vanet environment by data communication. In Proceedings of the 2019 3rd International Conference on Computing Methodologies and Communication (ICCMC), Erode, India, 27–29 March 2019; pp. 238–242.
31. Wang, Z.; Xu, G.; Zhang, M.; Guo, Y. Collision Avoidance Models and Algorithms in the Era of Internet of Vehicles. In Proceedings of the 2020 IEEE 3rd International Conference of Safe Production and Informatization (IICSPI), Chongqing, China, 28–30 November 2020; pp. 123–126.
32. Yuan, Y.; Tasik, R.; Adhatarao, S.S.; Yuan, Y.; Liu, Z.; Fu, X. RACE: Reinforced cooperative autonomous vehicle collision avoidance. *IEEE Trans. Veh. Technol.* **2020**, *69*, 9279–9291. [\[CrossRef\]](#)
33. Olmos, A.G.; Vazquez-Gallego, F.; Sedar, R.; Samoladas, V.; Mira, F.; Alonso-Zarate, J. An automotive cooperative collision avoidance service based on mobile edge computing. In Proceedings of the 18th International Conference on Ad-Hoc Networks and Wireless (AdHoc-Now 2019), Luxembourg, 1–3 October 2019; pp. 601–607.
34. Vázquez-Gallego, F.; Vilalta, R.; García, A.; Mira, F.; Vía, S.; Muñoz, R.; Alonso-Zarate, J.; Catalan-Cid, M. A mobile edge computing-based collision avoidance system for future vehicular networks. In Proceedings of the IEEE INFOCOM 2019—IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), Paris, France, 29 April–2 May 2019; pp. 904–905.
35. Malinverno, M.; Avino, G.; Casetti, C.; Chiasserini, C.F.; Malandrino, F.; Scarpina, S. Performance analysis of C-V2I-based automotive collision avoidance. In Proceedings of the 2018 IEEE 19th International Symposium on “A World of Wireless, Mobile and Multimedia Networks” (WoWMoM), Chania, Greece, 12–15 June 2018; pp. 1–9.
36. Malinverno, M.; Avino, G.; Casetti, C.; Chiasserini, C.F.; Malandrino, F.; Scarpina, S. Mec-based collision avoidance for vehicles and vulnerable users. *arXiv* **2019**, arXiv:1911.05299.
37. Abou El Assad, Z.E.; Mousannif, H.; Al Moatassime, H. A real-time crash prediction fusion framework: An imbalance-aware strategy for collision avoidance systems. *Transp. Res. Part Emerg. Technol.* **2020**, *118*, 102708. [\[CrossRef\]](#)
38. Li, G.; Yang, Y.; Zhang, T.; Qu, X.; Cao, D.; Cheng, B.; Li, K. Risk assessment based collision avoidance decision-making for autonomous vehicles in multi-scenarios. *Transp. Res. Part C Emerg. Technol.* **2021**, *122*, 102820. [\[CrossRef\]](#)
39. Erel-Özçevik, M.; Canberk, B. Road to 5G Reduced-Latency: A Software Defined Handover Model for eMBB Services. *IEEE Trans. Veh. Technol.* **2019**, *68*, 8133–8144. [\[CrossRef\]](#)
40. Drożdżel, P.; Tarkowski, S.; Rybicka, I.; Wrona, R. Drivers’ reaction time research in the conditions in the real traffic. *Open Eng.* **2020**, *10*, 35–47. [\[CrossRef\]](#)

41. Taleb, T.; Benslimane, A.; Letaief, K.B. Toward an effective risk-conscious and collaborative vehicular collision avoidance system. *IEEE Trans. Veh. Technol.* **2010**, *59*, 1474–1486. [\[CrossRef\]](#)
42. Al-Saadeh, O.; Wikstrom, G.; Sachs, J.; Thibault, I.; Lister, D. End-to-end latency and reliability performance of 5G in London. In Proceedings of the 2018 IEEE Global Communications Conference (GLOBECOM), Abu Dhabi, United Arab Emirates, 9–13 December 2018; pp. 1–6.
43. Li, J.; Chen, L.; Chen, J. Enabling technologies for low-latency service migration in 5G transport networks. *J. Opt. Commun. Netw.* **2021**, *13*, A200–A210. [\[CrossRef\]](#)
44. Tinkhede, P.; Ingole, P. Survey of handover decision for next generation. In Proceedings of the International Conference on Information Communication and Embedded Systems (ICICES2014), Chennai, India, 27–28 February 2014; pp. 1–5.
45. Kovatsch, M.; Lanter, M.; Shelby, Z. Californium: Scalable cloud services for the internet of things with coap. In Proceedings of the 2014 International Conference on the Internet of Things (IOT), Cambridge, MA, USA, 6–8 October 2014; pp. 1–6.
46. Docker: Empowering App Development for Developers. 2019. Available online: <https://www.docker.com/> (accessed on 6 April 2021).
47. Kubernetes: Production-Grade Container Orchestration. 2019. Available online: <https://kubernetes.io/> (accessed on 6 April 2021).
48. SumoTraciConnection. 2019. Available online: <https://sumo.dlr.de/javadoc/traas/it/polito/appeal/traci/SumoTraciConnection.html> (accessed on 6 April 2021).
49. Lim, K.G.; Lee, C.H.; Chin, R.K.Y.; Yeo, K.B.; Teo, K.T.K. SUMO enhancement for vehicular ad hoc network (VANET) simulation. In Proceedings of the 2017 IEEE 2nd International Conference on Automatic Control and Intelligent Systems (I2CACIS), Kota Kinabalu, Malaysia, 21 October 2017; pp. 86–91.
50. SUMO—Simulation of Urban Mobility. 2019. Available online: <http://sumo.sourceforge.net/> (accessed on 6 April 2021).
51. TraCI/TraaS—SUMO Documentation. 2019. Available online: <https://sumo.dlr.de/docs/TraCI/TraaS.html> (accessed on 6 April 2021).
52. Artal-Villa, L.; Hussein, A.; Olaverri-Monreal, C. Extension of the 3DCoAutoSim to Simulate Vehicle and Pedestrian Interaction based on SUMO and Unity 3D. In Proceedings of the 2019 IEEE Intelligent Vehicles Symposium (IV), Paris, France, 9–12 June 2019; pp. 885–890.
53. Puangnak, K.; Puisamlee, W.; Puangnak, K.; Rachsiriwatcharabul, N. Evaluation of MQTT and CoAP for Vehicle Traffic Monitoring. In Proceedings of the 2019 16th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology (ECTI-CON), Pattaya, Thailand, 10–13 July 2019; pp. 915–918.
54. Eclipse Californium (Cf) CoAP Framework. 2019. Available online: <https://www.eclipse.org/californium/> (accessed on 6 April 2021).
55. Gesbert, D.; Hanly, S.; Huang, H.; Shitz, S.S.; Simeone, O.; Yu, W. Multi-cell MIMO cooperative networks: A new look at interference. *IEEE J. Sel. Areas Commun.* **2010**, *28*, 1380–1408. [\[CrossRef\]](#)
56. Makridis, M.; Mattas, K.; Borio, D.; Giuliani, R.; Ciuffo, B. Estimating reaction time in Adaptive Cruise Control System. In Proceedings of the 2018 IEEE Intelligent Vehicles Symposium (IV), Changshu, China, 26–30 June 2018; pp. 1312–1317.