Proposal of an Automotive Collision Avoidance System based on Edge Computing

Nicolas Nevigato, Mauro Tropea and Floriano De Rango
Dimes Department, University of Calabria, Via P. Bucci 39/c, 87036 Rende (CS), Italy

Keywords: Automotive, Edge Computing, Multi-Access Edge Computing (MEC), Collision Avoidance.

Abstract: Multi-Access Edge Computing (MEC) is a network architecture that enables cloud computing capabilities and an IT service environment at the edge of the cellular network and, more in general at the edge of any network. It is often used in safety-critical and mission-critical communications as it is able to guarantee low latency and high bandwidth. It takes advantage of container-based virtualization and in collaboration with 5G cellular networks, it is a key enabler for the deployment of vehicular use cases, such as the improvement of road safety. This paper presents the implementation of a collision avoidance system based on MEC, with the ability to switch communication to a cloud server when possible, trying to guarantee the required constraints and balancing the communication among the servers avoiding of overloading edge layer. The simulation results have proved how in some cases the MEC-5G combination is the best one for the system in order to avoid collisions on the road.

1 INTRODUCTION

The automotive industry is constantly evolving, thanks to the numerous and important technological transformations it is going through (Giust et al., 2018). Internet of Vehicles (IoVs) (Yang et al., 2014) is a network of cars communicating with each other and with pedestrians handheld devices in a distributed manner supporting the use of data created by connected cars and Vehicular Ad-Hoc Networks (VANETs) (Sharma et al., 2019). Vehicles are becoming smarter and more aware of their surroundings through the integration of new sensors and new communication systems. They are able to exchange information with other vehicles (V2V communication), with road infrastructure (V2I communication), with pedestrians (V2P communication) and with communication networks (V2N communication). All these types of communication are incorporated in the so-called Vehicle-to-Everything communication (V2X communication), that is, communication from vehicle to everything.

The vehicular network aims to improve road safety, increase traffic efficiency and save energy giving also attention to the emission issues (Santamaria et al., 2019), (Fazio et al., 2017b). Very studied aspects for the ad-hoc network are routing issues as it is possible to view in (Zhou et al., 2006), (Fotino et al., 2007), (Socievole et al., 2011), and also for VANET is an important topic as it is possible to view in (De Rango et al., 2009), (Fazio et al., 2013). The cell permanence time and the mobile analysis for predictive services are important aspects in wireless networks and they are object of study by research communities, such as it possible to view in (De Rango et al., 2008), (Fazio et al., 2017a), (Fazio et al., 2016), (Frnda et al., 2013). A fundamental aspect to constantly consider in the automotive context is the possibility of guaranteeing an effective service in order to obtain low-latency communications. The 5G network can help in this direction, as it guarantees ultra-low latency and ultra-high reliability in highly mobile and densely connected scenarios. In addition, the Multi-access Edge Computing, also called Mo-

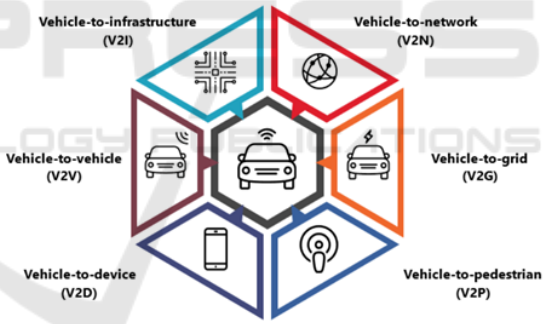


Figure 1: Communications in vehicular networks.

Nevigato, N., Tropea, M. and De Rango, F.
Proposal of an Automotive Collision Avoidance System based on Edge Computing.
DOI: 10.5220/0009971101280135
In *Proceedings of the 10th International Conference on Simulation and Modeling Methodologies, Technologies and Applications (SIMULTECH 2020)*, pages 128-135
ISBN: 978-989-758-444-2

bile Edge Computing, (MEC) network architecture (Porambage et al., 2018), (Pham et al., 2019), which brings the processing, storage and network capabilities to the edge of the network, can be considered a key factor in these vehicular networks. In fact, it meets the requirements of low latency and high bandwidth and can be implemented on the roadside and in the infrastructures of mobile network operators to support vehicle-to-infrastructure communication.

Several applications represent a concrete use case of MEC technology in collaboration with the latest generation networks. A possible use case in the automotive sector is to make a driver's driving safer, through assisted driving systems to improve the reliability, efficiency and quality of road transport. For example, it is possible to think of a system capable of predicting potential collisions by anticipating detection of a road hazard. In aid of this, it is possible to use *virtualization* (Jain and Choudhary, 2016) with Docker containers (Bernstein, 2014) and the container orchestrator Kubernetes (Shah and Dubaria, 2019).

The rest of this paper is organized as follows: Section 2 presents a brief overview on the vehicular environment; Section 3 presents the proposal of this work on the considered research topic. In Section 4, a description of the developed system is given with a description of the simulation environment created in java and the simulator implementation details. The numerical results are presented in Section 5. Finally, Section 6 concludes the paper.

2 RELATED WORK

A lot of work exist in literature on topics that regard the automative envirnoment and Vehicular Ad-Hoc Networks (VANETs). In general, they face with different issues such as communication protocols, new technologies used in particular scenarios, virtualization capabilities and so on. Many works have been written on the new paradigm called Internet of Vehicles (IoVs).

In (Song and Jiang, 2015) the authors aim at discovering the related characteristics between the active safety strategy and velocity through the visual analysis of Internet of Vehicles (IoV) warning data. They propose a warning velocity association model (WVAM) able to visually validate the relation between warnings and velocity. Experiments using Gephi show meaningful results.

In (Xiong et al., 2019) the authors propose an elastic wave equation model to reveal the relation between safety distance and road congestion. It can be found that the propagation speed of road congestion is largely affected by the safety distance. To recoup the efficient road safety and alleviate road congestion, an optimization problem is formulated with cooperative communication and computing via platoons that aims to minimize the total safety distance. Simulation experiments show that the proposed algorithm leads to near-optimal results with low complexity but no overhead of vehicular information exchange.

In (Khelil and Soldani, 2014) the authors explore recent Device-to-Device (D2D) research efforts and review their suitability to safety-critical Internet of Vehicles (IoV) applications, such as cooperative or autonomous driving. Their review work shows that current approaches ignore high relative node mobility and accurate proximity measurements, and the crucial research challenges mainly are those related to maintaining the required QoS level in highly fluctuating D2D communications. In this work the authors want to provide a roadmap towards adopting the emerging D2D technique for critical IoV communications.

In (Santamaria et al., 2018) the attention is focused on the design of a new approach for vehicular environments able to gather information during mobile node trips, for advising dangerous or emergency situations by exploiting on-board sensors. It is assumed that each vehicle has an integrated on-board unit composed of several sensors GPS device, able to spread alerting messages around the network, regarding warning and dangerous situations/conditions and it is assumed that each vehicle can communicate with on roadside devices called RSU able to spread warning messages. In this way, if an accident occurs, the arriving cars will, probably, avoid delay and danger situations.

3 PROPOSED SOLUTION

The system implemented for the use case shown above has the following operating principle: the vehicles periodically, by a device installed on board, send messages that contain information about their status. In particular, the message contains the *ID*, the position and speed of the vehicle in question. These messages are received by a MEC server connected to a roadside base radio station, running on a Docker (doc, 2019) container managed by Kubernetes (kub, 2019). The information is saved and processed by a software that can understand when an emergency braking is taking place and therefore anticipate any collisions. In this case 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, automatically, by means of suit-

able devices, the braking system is activated to stop the vehicle safely and avoid a collision.

A dynamic switching algorithm has also been implemented that allows the vehicle to instantly decide whether to send the message to the MEC server or to a cloud located in Europe, so as not to overload the device on the roadside when it is not necessary.

3.1 Docker and Kubernetes

A container is an isolated environment that shares the same kernel of the operating system. It contains the code and all the dependencies to be able to run an application quickly and easily.

Docker is an open source project that provides a systematic way to automate the faster deployment of Linux applications within portable containers (Bernstein, 2014; Ahmed and Pierre, 2018). It uses a client-server architecture model and is based on the concept of image. A Docker image can include only the fundamentals of the operating system, or it can consist of a sophisticated stack of predefined applications ready for launch. When creating images with Docker, each command executed forms a new layer above the previous one. To create a new image you use the Dockerfile, a script composed of various instructions and arguments to automatically perform actions on a basic image. Starting from the same image it is then possible to create multiple containers.

Kubernetes is an open source container orchestrator used to automate the distribution, scaling and orchestration of container-based applications (Shah and Dubaria, 2019). In Kubernetes, the *pod* is the fundamental element. It consists of a group of tightly coupled containers located on the same host that share the same IP address. Kubernetes guarantees reliability as it can automatically restart containers that fail during execution and terminate those that do not respond, always guaranteeing a certain number of containers in execution.

3.2 Design and Study of Critical Parameters

The design of the system is characterized primarily by a study carried out in order to understand the latency and reaction times necessary to avoid collisions between vehicles, based on the examined traffic scenarios.

Two vehicles identified by an identifier number (*ID*) are considered. In particular, the vehicle that is ahead has $ID = 0$, while the one that follows has $ID = 1$. For simplicity, in hourly laws, these *IDs* are put as superscripts. Two vehicles travelling a stretch of road at constant speed have the same deceleration capacity "a" and maintain a distance "d" from each other. It is assumed that the first vehicle for any reason is forced to brake abruptly. With t_{cr} is indicated the sum of the latency time and the reaction time necessary for the second vehicle to start braking. The hourly law with instant started $t_0 = 0$ for the vehicle with $ID = 0$ is as follows:

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

The hourly law with instant start $t_0 = 0$ for the vehicle with $ID = 1$ is as follows:

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

Let suppose that due to braking vehicle 0 stopped its run, it is possible to calculate the instant of collision between vehicle 0 and vehicle 1 that follows, with the assumption of consider equal the two paths traveled by the vehicles.

$$x_0^0 + \frac{1}{2}\frac{(v_0^0)^2}{a} = x_0^1 + \left(v_0^1 + at_{cr}\right)t - \frac{1}{2}a\left(t^2 + t_{cr}^2\right) \quad (3)$$

Carrying out the calculations we obtain the following second degree equation:

$$\frac{1}{2}at^2 - \left(v_0^1 + at_{cr}\right)t + \frac{1}{2}at_{cr}^2 + d + \frac{1}{2}\frac{(v_0^0)^2}{a} = 0 \quad (4)$$

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

$$\begin{aligned} \Delta = & \\ & \left(v_0^1 + at_{cr}\right)^2 + \\ & -4\left(\tfrac{1}{2}a\right)\left(\tfrac{1}{2}at_{cr}^2 + d + \tfrac{1}{2}\tfrac{(v_0^0)^2}{a}\right) > 0 \\ & \Longrightarrow t_{cr} > \tfrac{d}{v} \end{aligned} \quad (5)$$

Therefore, to occur a collision between the two vehicles, t_{cr} must be greater than the ratio between distance and speed of the considered vehicles.

As regards the tools used in the realization, the open source simulator Sumo (sum, 2019a) was chosen to generate realistic vehicular traffic in the urban environment, so as to have data from vehicles and to be able to process them and the Eclipse development environment to interface with the simulator and to be able to program it using code written in Java language, with the library called *SumoTraciConnection* (sum, 2019b).

To carry out the communication between vehicle and MEC server and between vehicle and Cloud server, the CoAP protocol (Puangnak et al., 2019) was chosen using the Californium library written in Java (cal, 2019). Each vehicle was created as a CoAP client, and periodically makes requests to the MEC or Cloud server, as appropriate, which acts as a CoAP server.

While the simulator and, therefore, the various "vehicle" clients are running on a PC, the MEC server was hosted in a Docker container in order to have all the necessary dependencies and to be able to run it on another PC. In order to guarantee reliability and greater ease of use, the Docker container for the server was managed by the Kubernetes orchestrator.

4 SYSTEM DEVELOPMENT

The development of the system consists first of all in the implementation of vehicular traffic using the SUMO simulator (Lim et al., 2017). Subsequently, the implementation of the clients, that is, the communication from vehicles to server has been developed. Finally, the implementation of the MEC server and, therefore, the communication from server to vehicle has been created. The final system is able to simulate a traffic of vehicle that communicate between them and with the server in order to manage a collision avoiding mechanism and, so guaranteeing security on the vehicular environment.

4.1 Vehicular Traffic with Sumo

In the realization of the system, in order to use Sumo, a section of the road route related to the SS107 Silana-Crotonese in Italy was initially downloaded and processed with the command *netConvert* from the command line on windows, to obtain in output a *xml* type file compatible with Sumo.

This file contains all the attributes relating to the road portion. For example, there are the *IDs* related to the two lanes of the roadway with the two travel directions, the name of the considered road portion and the maximum allowed speed expressed in m/s.

Subsequently, an *XML* type file was created to define the vehicles travelling on the previously chosen road portion. For each vehicle it is possible to indicate its acceleration and deceleration value, id, length, maximum speed in m/s and minimum gap with the previous vehicle, and so on.

Finally, in order to start the simulation, it was necessary to generate the *XML* configuration file with the .config extension. As input it takes 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. There is a time section where the duration of the simulation with the step is specified. It is also possible to output a file with the simulation logs.

The implemented system follows a client-server structure with CoAP communication. The Sumo simulator connected to Eclipse is used to manage client-side communication, that is, it treats the vehicles that are travelling in the map. This communication can be only with a MEC server, only with a Cloud server or characterized by a dynamic switching between MEC Server and Cloud.

4.2 Communication Towards Edge or Cloud

As regards the communication of the vehicle to the server, in general, each one represents a CoAP client, able to send information to the server through the POST method of the Californium library (Kovatsch et al., 2014). The information is enclosed in a message and relates to the status of the vehicle, such as id, speed and position. Furthermore, every command that the vehicles receive from the server is managed to pass it to the simulator. In order to realize this process, a Java class has been implemented which contains the logic related to the vehicle.

Briefly, the behavior of the individual vehicle is shown in the following flow chart, figure 2.

4.3 Dynamic Switching Edge - Cloud

The implementation of the dynamic switching algorithm has the aim of not overloading the Edge server when it is not essential, deviating part of the traffic to a Cloud server. In particular, each vehicle can decide whether to communicate with the Cloud server or with the MEC server, so that the roadside device will receive fewer packets number since part of these will be managed by the Cloud. The decision made by the vehicle depends on the current vehicle traffic conditions, with the aim of keeping the collisions number as low as possible. Once the previous and the next vehicle of a generic vehicle have been identified, this performs a calculation in order to understand what communication to establish, if it is better communicate with the edge or cloud server. Since the instant of collision between two vehicles depends on the total time between latency and reaction, and the reaction, as shown in the study done previously, for the same amount of deceleration and speed between the vehicles must be greater of the distance ratio between two

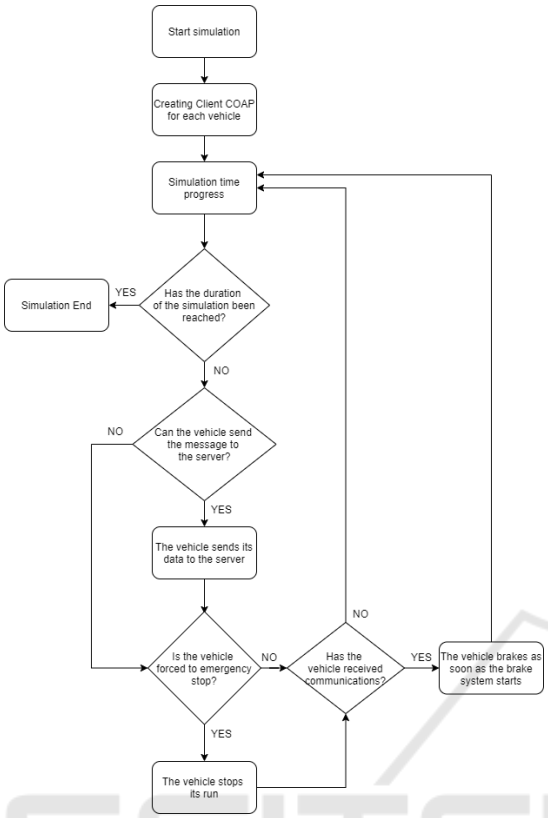


Figure 2: Flow chart of the behavior of a vehicle.

vehicles and the speed for collision, this ratio is considered in the calculation made by the vehicle with respect to the following and previous vehicle in order to choose the server. If in at least one of the two cases, the ratio is less than the total time between the latency of the Cloud and the reaction of the vehicle, it means that communication to the Edge is necessary otherwise there would be a collision choosing the Cloud. Otherwise, communication with the Cloud is chosen. In general, the algorithm is described by the following flow chart, figure 3.

As regards incoming communications, each vehicle must check whether it has received warnings from both the Cloud server and the MEC server. Therefore, two *CoapHandlers* procedure were needed to manage both communications asynchronously.

4.4 Server MEC

The role of the MEC server on the roadside, in the implemented application, is to receive data from travelling vehicles, process it to understand if there is a danger or emergency braking by a vehicle, and immediately notify all vehicles that follow in order to avoid collisions. Everything is summarized in the *handlePOST* method which manages every single commu-

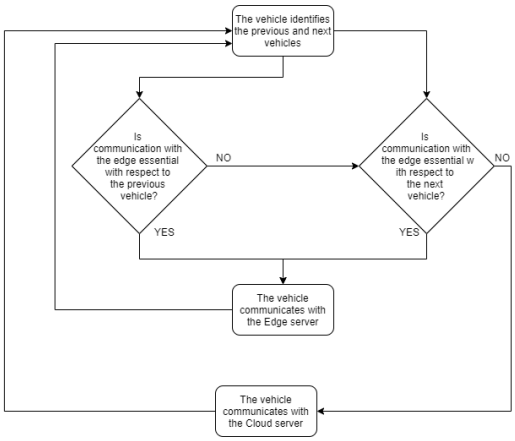


Figure 3: Flow chart relating to dynamic switching.

nication with the client. After accepting the communication, the server decodes the received message and performs calculations to understand if the vehicle is braking abruptly. In particular, it makes the difference between two consecutive vehicle speeds, and if this difference is greater than a certain threshold, it is identified as a dangerous condition. Therefore in this case, the server identifies all the vehicles that are arriving within a certain range, in order to alert them in time and avoid collisions. The operating principle is described briefly in the following figure 4.

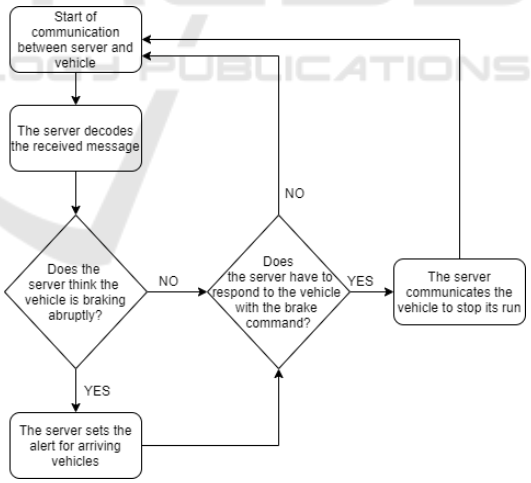
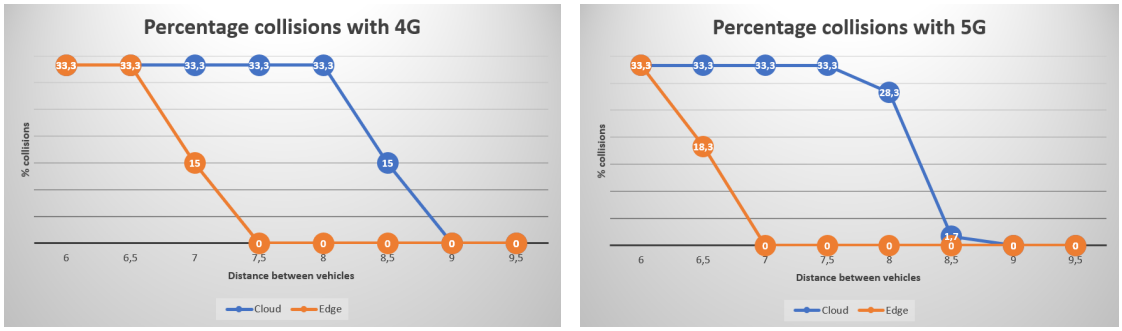


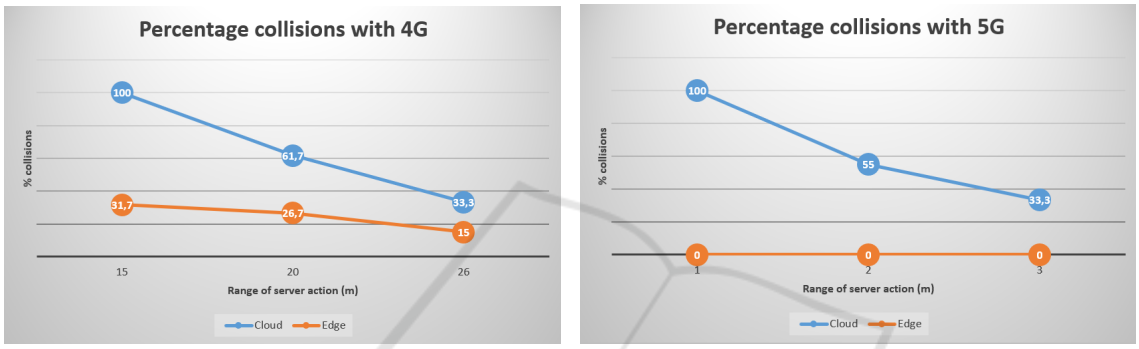
Figure 4: Flow chart relating to the behavior of the MEC server.

Once the Java Server code was realized, a Docker container could be created. This latter was then published on a private repository created in the Docker Hub registry so that it can be managed with Kubernetes. In Kubernetes a deployment has been created where a *pod* characterized by the single container is declared. Finally, a service has been created to make the *pod* accessible.



(a) (b)

Figure 5: Percentage of Collisions with 4G and 5G network varying vehicles distance.



(a) (b)

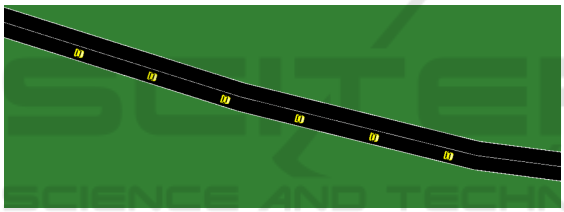
Figure 6: Percentage of Collisions with 4G and 5G network varying server range action.

5 PERFORMANCE ANALYSIS

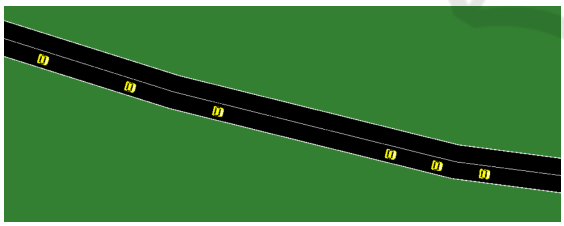
To understand the performance of the implemented system, several simulation tests were carried out. Firstly, the collisions percentage between vehicles varying the distance between them, using first communication with MEC Server and then with Cloud Server, considering 4G and 5G technology was assessed. In particular, both technologies have been used from a simulation point of view, considering the latency times that characterize each of them. The scenario considered to make the measurements is as follows: four vehicles at a distance ”d” from each other, with constant speed equal to 50 km/h, same deceleration values, random reaction times between 450 ms and 500 ms (Makridis et al., 2018) and a range of action of the server equal to 50 m from the point where the hazard event was generated. The first vehicle at a certain moment is forced to brake suddenly. The distances ”d” expressed in meters and taken into consideration are: 6, 6.5, 7, 7.5, 8, 8.5, 9, 9.5. Twenty runs were carried out for each configuration considering first the communication with the MEC server (4G and 5G) and then with the Cloud server (4G and 5G). The results obtained are shown in the following graphs, see figure 5.

As it is possible to appreciate, considering the same distance between two vehicles, the percentage of collisions that characterizes communication with the Edge is lower than communication with the Cloud. Furthermore, considering first 4G and then 5G, in some cases the number of collisions for the same distance between two vehicles decreases. Subsequently, the percentage of collisions between vehicles was evaluated, no longer changing the distance between two, but varying the range of action of the server with respect to the point where the hazard event was generated. The scenario considered is as follows: four vehicles at a distance of 7 m from each other, with a constant speed of 50 km/h, same deceleration values, random reaction times between 450 ms and 500 ms (Makridis et al., 2018) and a range of action of the server equal to ”r” meters with respect to the point where the hazard event was generated. The first vehicle at a certain moment is forced to brake suddenly. The ranges used expressed in meters are: 15, 20, 26. These values have been chosen respectively to ensure that the server once only warns the vehicle following the one that suddenly brakes, then the next two and finally all three vehicles that follow. Twenty runs were

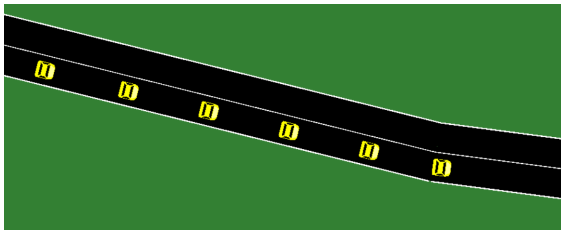
carried out for each configuration considering first the communication with the MEC server (4G and 5G) and then with the Cloud server (4G and 5G). The results obtained are those shown in the figure 6. As you can see, both using 4G and 5G technology, and considering communication with the Cloud server and with the MEC server, as the range of action of the server increases, the percentage of collisions decreases. This happens because a vehicle warned in time of the danger will start braking earlier avoiding collision with the previous vehicle. Therefore the more vehicles are warned by the server, the lower the collision rate. Another parameter that has been evaluated is the percentage of use of the Cloud and MEC servers in dynamic switching. This algorithm allows, especially in certain vehicular traffic conditions, not to overload the Edge and communicate with the Cloud server which has higher performance. Three scenarios are considered in the evaluation, starting from one that is not congested up to the congested one, as shown in the figures 7a, 7b and 7c and the communication is made towards cloud or edge server on the basis of latency issues.



(a)



(b)



(c)

Figure 7: (a) Uncongested Vehicular Traffic; (b) Low Traffic Congestion; (c) Congested Vehicular Traffic.

6 CONCLUSIONS

In the paper, the attention was focused to the service delivery paradigm called Edge Computing, realizing a specific use case that demonstrates its importance. In particular, an assisted guidance system for collision prediction has been developed and it has been shown how the use of Edge technology is essential in some circumstances where the latency parameter that characterizes communications must be very low. This is because, with Edge Computing, each Cloud functionality moves close to the end user and therefore to the edge of the network. Therefore in the system it was necessary to implement an MEC server to manage communications with vehicles, run on a Docker container managed by the Kubernetes orchestrator. In order to show the improvements of Edge Computing compared to Cloud Computing, a comparison has been made between the communication of a vehicle to the Egde server and to the Cloud server, evaluating the percentage of collisions of the vehicles. Obviously, communication with the Edge allows to significantly reduce the number of collisions as it guarantees minimum latency. It has also been demonstrated from a simulation point of view how in some cases the MEC-5G combination is essential for this type of system.

REFERENCES

(2019). Docker: Empowering app development for developers. https://www.docker.com/.

(2019). Eclipse californium (cf) coap framework. https://www.eclipse.org/californium/.

(2019). Kubernetes: Production-grade container orchestration. https://kubernetes.io/.

(2019a). Sumo - simulation of urban mobility. http://sumo.sourceforge.net/.

(2019b). Sumotraciconnection. https://sumo.dlr.de/javadoc/traas/it/polito/appeal/traci\\/SumoTraciConnection.html.

Ahmed, A. and Pierre, G. (2018). Docker container deployment in fog computing infrastructures. In *2018 IEEE International Conference on Edge Computing (EDGE)*, pages 1–8. IEEE.

Bernstein, D. (2014). Containers and cloud: From lxc to docker to kubernetes. *IEEE Cloud Computing*, 1(3):81–84.

De Rango, F., Fazio, P., and Marano, S. (2008). Utility-based predictive services for adaptive wireless networks with mobile hosts. *IEEE Transactions on Vehicular Technology*, 58(3):1415–1428.

De Rango, F., Veltri, F., Fazio, P., and Marano, S. (2009). Two-level trajectory-based routing protocol for vehicular ad hoc networks in freeway and manhattan environments. *Journal of Networks*, 4(9):866–880.

Fazio, P., De Rango, F., Tropea, M., and Voznak, M. (2017a). Cell permanence time and mobility analysis in infrastructure networks: Analytical/statistical approaches and their applications. *Computers & Electrical Engineering*, 64:506–523.

Fazio, P., Sottile, C., Santamaria, A. F., and Tropea, M. (2013). Vehicular networking enhancement and multi-channel routing optimization, based on multi-objective metric and minimum spanning tree. *Advances in Electrical and Electronic Engineering*, 11(5):349–356.

Fazio, P., Tropea, M., De Rango, F., and Voznak, M. (2016). Pattern prediction and passive bandwidth management for hand-over optimization in qos cellular networks with vehicular mobility. *IEEE Transactions on Mobile Computing*, 15(11):2809–2824.

Fazio, P., Tropea, M., and Marano, S. (2017b). Node re-routing and congestion reduction scheme for wireless vehicular networks. *Wireless Personal Communications*, 96(4):5203–5219.

Fotino, M., Gozzi, A., Cano, J.-C., Calafate, C., De Rango, F., Manzoni, P., and Marano, S. (2007). Evaluating energy consumption of proactive and reactive routing protocols in a manet. In *IFIP Conference on Wireless Sensor and Actor Networks*, pages 119–130. Springer.

Frnda, J., Voznak, M., Rozhon, J., and Mehic, M. (2013). Prediction model of qos for triple play services. In *2013 21st Telecommunications Forum Telfor (TELFOR)*, pages 733–736. IEEE.

Giust, F., Sciancalepore, V., Sabella, D., Filippou, M. C., Mangiante, S., Featherstone, W., and Munaretto, D. (2018). Multi-access edge computing: The driver behind the wheel of 5g-connected cars. *IEEE Communications Standards Magazine*, 2(3):66–73.

Jain, N. and Choudhary, S. (2016). Overview of virtualization in cloud computing. In *2016 Symposium on Colossal Data Analysis and Networking (CDAN)*, pages 1–4. IEEE.

Khelil, A. and Soldani, D. (2014). On the suitability of device-to-device communications for road traffic safety. In *2014 IEEE World Forum on Internet of Things (WF-IoT)*, pages 224–229. IEEE.

Kovatsch, M., Lanter, M., and Shelby, Z. (2014). Californium: Scalable cloud services for the internet of things with coap. In *2014 International Conference on the Internet of Things (IOT)*, pages 1–6. IEEE.

Lim, K. G., Lee, C. H., Chin, R. K. Y., Yeo, K. B., and Teo, K. T. K. (2017). Sumo enhancement for vehicular ad hoc network (vanet) simulation. In *2017 IEEE 2nd International Conference on Automatic Control and Intelligent Systems (I2CACIS)*, pages 86–91. IEEE.

Makridis, M., Mattas, K., Borio, D., Giuliani, R., and Ciuffo, B. (2018). Estimating reaction time in adaptive cruise control system. In *2018 IEEE Intelligent Vehicles Symposium (IV)*, pages 1312–1317. IEEE.

Pham, Q.-V., Fang, F., Ha, V. N., Le, M., Ding, Z., Le, L. B., and Hwang, W.-J. (2019). A survey of multi-access edge computing in 5g and beyond: Fundamentals, technology integration, and state-of-the-art. *arXiv preprint arXiv:1906.08452*.

Porambage, P., Okwuibe, J., Liyanage, M., Ylianttila, M., and Taleb, T. (2018). Survey on multi-access edge computing for internet of things realization. *IEEE Communications Surveys & Tutorials*, 20(4):2961–2991.

Puangnak, K., Puisamlee, W., Puangnak, K., and Rachsiriwatcharabul, N. (2019). Evaluation of mqtt and coap for vehicle traffic monitoring. In *2019 16th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology (ECTI-CON)*, pages 915–918. IEEE.

Santamaria, A. F., Fazio, P., Raimondo, P., Tropea, M., and De Rango, F. (2019). A new distributed predictive congestion aware re-routing algorithm for co 2 emissions reduction. *IEEE Transactions on Vehicular Technology*, 68(5):4419–4433.

Santamaria, A. F., Tropea, M., Fazio, P., and De Rango, F. (2018). Managing emergency situations in vanet through heterogeneous technologies cooperation. *Sensors*, 18(5):1461.

Shah, J. and Dubaria, D. (2019). Building modern clouds: using docker, kubernetes & google cloud platform. In *2019 IEEE 9th Annual Computing and Communication Workshop and Conference (CCWC)*, pages 0184–0189. IEEE.

Sharma, S. et al. (2019). Vehicular ad-hoc network: An overview. In *2019 International Conference on Computing, Communication, and Intelligent Systems (ICCCIS)*, pages 131–134. IEEE.

Socievole, A., De Rango, F., and Coscarella, C. (2011). Routing approaches and performance evaluation in delay tolerant networks. In *2011 Wireless Telecommunications Symposium (WTS)*, pages 1–6. IEEE.

Song, W. and Jiang, B. (2015). Visualization of iov warning data based on warning velocity association model. In *2015 8th International Symposium on Computational Intelligence and Design (ISCID)*, volume 2, pages 30–33. IEEE.

Xiong, K., Leng, S., He, J., Wu, F., and Wang, Q. (2019). Recouping efficient safety distance in iov-enhanced transportation systems. In *ICC 2019-2019 IEEE International Conference on Communications (ICC)*, pages 1–6. IEEE.

Yang, F., Wang, S., Li, J., Liu, Z., and Sun, Q. (2014). An overview of internet of vehicles. *China communications*, 11(10):1–15.

Zhou, B., Lee, Y.-Z., Gerla, M., and De Rango, F. (2006). Geo-lanmar: a scalable routing protocol for ad hoc networks with group motion. *Wireless Communications and Mobile Computing*, 6(7):989–1002.