International Journal of Computer Information Systems and Industrial Management Applications. 136
ISSN 2150-7988 Volume 13 (2021) pp. 136-148

Real-Time Databases for Better Cooperative Accident Prevention Systems

Islam Elleuch[1], Achraf Makni[2] and Rafik Bouaziz[1]

[1] University of Sfax, MIRACL Laboratory, Faculty of Economics and Management,
Airport Road, km 4, B.P. 1088, 3018, Sfax, Tunisia
islam.elleuch@fsegs.rnu.tn, rafik.bouaziz@usf.tn

[2] University of Sfax, MIRACL Laboratory, National School of Electronic and Telecommunication,
Tunis Road, km 10, B.P. 1163, 3018, Sfax, Tunisia
achraf.makni@enetcom.usf.tn

***Abstract*: The huge vehicle frequency can provoke serious accidents leading to critical consequences on human security and destruction of vehicles. Therefore, diverse systems based on *Vehicular Ad hoc Networks* (VANET) have been suggested to prevent accidents and improve users' security. Nevertheless, these systems can generate numerous problems since they do not store data in suitable databases, while VANET cannot support the transfer and the reception of all messages. To enhance road security, we propose in this paper a new system, entitled *Cooperative Safe Driving System* (CoopSafeDrivSyst). On the one hand, CoopSafeDrivSyst relies on *Vehicle-To-Vehicle* (V2V) and *Vehicle-To-Infrastructure* (V2I) communications to improve the interaction between vehicles, infrastructures, and the control center. On the other hand, it uses *Real-Time DataBases* (RT-DB) to handle data in real-time, efficiently, and precisely. It ensures road safety in almost all the dangerous situations that can be acquired on the road. Each vehicle stores its information, along with the nearby vehicles' information in its local RT-DB, and thus this information is also stored in the control center's RT-DB. Once a vehicle or the control center recognizes a road danger, the vehicle CoopSafeDrivSyst queries its database, triggers the other components to examine the situation, and determines the available operations to avoid collisions. Simulations of diverse driving situations, under the *Vehicles In Network Simulation* (VEINS) framework, confirm that CoopSafeDrivSyst provides a high level of road management thanks to VANET, the RT-DB, and the notion of Quality of Data used to decrease the number of data freshness transactions.**

***Keywords*:** Intelligent Vehicles, Accident Prevention, Real-Time Databases, Quality of Data.

I. Introduction

Driving is a difficult task demanding an important level of expertise and experience. About 1.3 million people die in road accidents each year according to the statistics of the Association for Safe International Road Travel [1]. Most of these accidents happen because of the driver's incapability to command the vehicle in unexpected or surprising In recent years, the combination of automotive manufacturing and intelligent transport systems technologies has made a notable contribution to enhancing road safety by decreasing the number of accidents on the roads.

Advanced Driver Assistance Systems (ADAS) constitute one of the most relevant fields of intelligent transportation systems [2]. They assist the driver in his driving task, by automating, partly or entirely, driving judgments and reactions. ADAS intend to avoid driver failure by supporting the driving process following information received from sensors, like lasers, radar, and cameras. Nonetheless, ADAS are autonomous and they are not effective in some circumstances (*e.g.*, bad weather conditions or a complicated situation). Thus, they remain unable to recognize static or dynamic obstacles efficiently. To improve road security, cooperative ADAS are being introduced. These systems cooperate thanks to wireless communications handled by the *Vehicular Ad Hoc Networks* (VANET) [3]. They authorize vehicles to transmit and receive information by *Vehicle-To-Vehicle* (V2V) and *Vehicle-To-Infrastructure* (V2I) communications. Vehicles and the infrastructure has to be equipped with the required devices. On the one hand, each vehicle uses a device in-stalled at the dashboard, called *On-Board Unit* (OBU), and at the infrastructure uses devices, called *Road Side Units* (RSU), on the other hand. Consequently, vehicles can share their perceptions and movement intentions to coordinate their actions, and thus better detect and avoid road dangers.

In our previous paper [4] to which we bring in this paper a major extension, we re-viewed the existing research works of the cooperative ADAS applications. We classified the cooperative ADAS into four categories based on their functionalities and application types. We called the first category as cooperative ADAS for basic functionality. It includes two functionalities that we consider prerequisites for improving road safety, namely cooperative adaptive cruise control and cooperative forward collision warning. We called the second one of the cooperative ADAS for collision prevention, which brings together the functionalities of cooperative intersection collision avoidance, cooperative left turn assistance, and cooperative highway on-ramp merging [5] [6]. As for the third category, we called it the cooperative ADAS for cooperative overtaking assistance [7]. Finally, we called the fourth category as cooperative ADAS for dangerous zones and object detection [8]. This category includes the

functionalities of cooperative dangerous zone alert and obstacle detections. However, we explained that these systems are still complex and complicated because they involve diverse sensors and functionalities. Moreover, they handle a huge data amount that needs to be updated periodically to display the current state of the environment. Furthermore, these systems can lead to dangerous problems in case of data congestion or waste, as they utilize probabilistic or recursive calculations based on the previous state of data without storage in a database. Therefore, there is a critical requirement to develop more secure and quick responding the cooperative ADAS, and hence to explain how to design these systems.

In order to reach these challenges and improve the driving system effectiveness and safety, we propose in this paper an appropriate approach for the design of an intelligent and consistent driving system as a part of cooperative ADAS, called *Cooperative Safe Driving System* (CoopSafeDrivSyst). To do so, we propose to integrate a *Real-Time DataBases Management System* (RT-DBMS), which meets exact results and the time constraints associated with data validity and operation capability. Data is not only stored and processed in the RT-DB, but also used to predict and make the appropriate decision in a timely and accurate manner. On the one hand, CoopSafeDrivSyst aims to anticipate, prevent and avoid road accidents. On the other hand, this system alert, with the cooperation of the control center about potential collisions. To correctly exhibit the current state of the environment, CoopSafeDrivSyst ought to transmit, save, update and analyze the large amount of data obtained directly via V2V or indirectly via V2I communications.

The rest of this paper is organized as follows. Section II presents the main foundation for our proposed CoopSafeDrivSyst system. Section III explains the road safety data as well as the design and management of the RT-DB in our system. Section IV de-scribes the performance metrics used to the validation of the performance of RT-DB in accident prevention. Section V presents and discusses the experimental environment and the simulation results. Finally, Section VI concludes the paper and gives the perspective of this work.

II. Building the foundation for a global cooperative ADAS

In this section, we present the basic elements of a new cooperative accident prevention system, which we propose in order to improve the level of road safety.

A. Characteristics of Real-Time Database Management Systems

Yuan defined *Real-Time Database Management Systems* (RT-DBMS) as follows: "A *Real-Time Database* (RT-DB) system combines the characteristics of real-time systems and database systems. It must not only meet the time constraints required by a real-time system, but also maintain the data consistency required by a database system" [9]. According to this definition, a RT-DBMS is identified as a database system that must be able to manage a large amount of data while respecting temporal constraints on data and transactions. Indeed, it must ensure all the functionalities of conventional DBMS, such as storage, retrieval and manipulation of data, on the one hand, and manage and verify the temporal validity of the stored data and the time constraints imposed on transactions, on the other hand.

According to [10], a real-time data item d is defined by the quadruplet $d = (d_{value}, d_{timestamp}, d_{avi}, mde)$, where d_{value} indicates the value of the recorded data, $d_{timestamp}$ describes the instant of data obtaining, d_{avi} represents to the absolute validity interval of the data, and mde represents the maximum tolerated error between the real value of the data and the value stored in the database. Therefore, the principal purpose of an RT-DBMS is not only to manage transactions on schedule but also to ensure data logical and temporal consistency and the execution of transactions before their maximum end time as a deadline. The temporal consistency leads to the requirement to keep coordination between the environment's current situation and the situation reported in the RT-DB.

In the context of an RTDB, the imprecision criterion indicates the maximum error tolerated between the current environment status and the value stored in the RT-DB. Besides, a real-time system consists of three categories of transactions which are either read-only, write-only, or update (read-write) transactions. A read-only transaction is a user query that does not change any data item. It reads both real-time data from the RT-DB and reads or writes non-real-time data. A write-only transaction captures the environment status, updates the sensor data, and writes in the RT-DB. This category of the transaction is performed periodically to accurately visualize the real environment status. An update transaction is triggered after updating sensors' data used to update derived data. This category of the transaction is executed sporadically to achieve read-write transactions.

Many works dealing with *Wireless Sensor Networks* (WSN) and *Mobile Ad hoc Networks* (MANET) have used RTDBs in the field of intelligent transportation [11] [12]. These works have clearly demonstrated the relevance of the contribution of these databases to improve the performance of these networks, and thus to enhance road safety. But, to the best of our knowledge, there is no research work that has used RT-DB in the VANET framework, while we perceive that RT-DB enable new contributions to VANET safety applications. Furthermore, in her thesis work, Marouane improved the architecture of existing autonomous ADAS by integrating an RT-DB to increase the fault tolerance of the sensors [13]. She proved that her proposed system efficiently handles time constraints on a large amount of data and can even estimate the value of a faulty sensor from the data stored in the database.

B. Towards the integration of RT-DB in cooperative ADAS

The functionality of cooperative driver assistance systems is increasingly evolving, and the volume of data being manipulated is growing. However, these systems do not manage the storage of this data, while its collection, distribution, and manipulation become cumbersome and time-consuming, and the risk of transactions missing their deadline increases. Similarly, the risk of data loss during transmission increases, leading to incorrect processing and potentially serious consequences.

For the cooperative ADAS to deal with these problems, we see that the definition and integration of Real-Time Databases (RT-DB) in cooperative ADAS under VANET is crucial. The manipulation of these RT-DB requires the selection of an *RT-DB Management System* (RTDBMS) and their installation in different components of the considered ADAS. This would allow better data maintenance and management, minimizing the number of accesses and reducing processing times. The ADAS would thus become more reactive and more efficient.

C. General structure of road traffic control

We aim to propose a new global system, entitled CoopSafeDrivSyst, to improve the driving assistance system performance and safety. CoopSafeDrivSyst takes into account the most important objects of the cooperative ADAS associated to anticipate most of the critical situations that may happen on the roads. In addition, this system enhances cooperation and permits vehicles to make better decisions by synchronizing the different objects. We designed CoopSafeDrivSyst as a combined system with three sub-systems. Each of these sub-systems handles an object or a collection of objects of the concerned cooperative ADAS. Each sub-system includes two basic functionalities: (i) the first one is the object of the *Cooperative Forward Collision Warning* (CFCW) and (ii) the second one is the object of the *Cooperative Adaptive Cruise Control* (CACC). The first sub-system corresponds to the *Cooperative Road Hazard Detection Persistent Sub-system* (CRHDPS), which prevents and avoids traditional collisions on the road. CRHDPS combines the functionalities of the cooperative ADAS objects related to the cooperative obstacle detections and the cooperative dangerous zone alert [5] [8]. The second sub-system corresponds to the *Cooperative Overtaking Assistance Persistent Sub-system* (COAPS). COAPS recognizes and avoids collisions while overtaking maneuvers using the functionalities of the cooperative overtaking assistance object [7]. The third sub-system corresponds to the *Cooperative Collision Prevention Persistent Sub-system* (CCPPS). CCPPS anticipates and avoids collisions at intersections and highway on-ramp merging. It combines the functionalities of the cooperative ADAS objects associated with cooperative left turn assistance, cooperative collision avoidance in an intersection, and highway on-ramp merging [5] [6].

CoopSafeDrivSyst intends to enhance cooperation to allow vehicles to make more suitable judgments by synchronizing the distinct sub-systems. Moreover, it is designed to function in an automatic driving style to guarantee more effective vehicle command and fully avoid collisions. Furthermore, CoopSafeDrivSyst actions at two levels: intra-vehicle and inter-vehicle.

The intra-vehicle level includes only the components of the intelligent vehicle. Indeed, it consists of an embedded system that incorporates both physical components and a software system that manages these physical components. Especially, an intelligent vehicle of CoopSafeDrivSyst contains Collection Units, an OBU, an Application Unit, an RT-DBMS, and Reaction Units. The OBU of each intelligent vehicle provides driving data, received by the Collection Units (*i.e.*, by the *Global Positioning System* (GPS) and the sensors), to the Application Unit through a wireless or wired connection. The Application Unit uses the communication capabilities of the OBU and the functionalities of the modules responsible for accident prevention. The RT-DBMS of a vehicle permits handling the driving data in real-time, efficiently, and precisely.

The inter-vehicle level, (*i.e.*, VANET), includes the intelligent vehicles, the RSU, and a control center in each area of the intelligent road. On the one hand, vehicles communicate immediately with each other; a vehicle's OBU transfers its data and collects data from other OBU, via V2V communications. On the other hand, a vehicle interacts with the RSU of the most proximate infrastructure, which communicates with the control center, via V2I communications. The latter has to receive information from the different RSU, analyze the collected information to determine the alerts to send for these RSU, which have to transmit the received alerts to vehicles in their range. An RT-DB is also integrated into the control centers to efficiently manage the large amount of data transferred by V2I communications.

III. Definition of the prerequisites for road safety

In this section, we present the basic resources required for the entire subsystems of CoopSafeDrivSyst, as well as the performance metrics defined for these subsystems.

A. Road safety data

To deal with the problem posed by the frequency of transferring a large amount of data in cooperative ADAS, we adopt CAM's proposal of not sending messages in a periodic manner, and only sending them when the difference between the current value of any data and its previous value is greater than a certain gap. In addition, we define a specific message for each type of data. We detail, in the following, our proposals for data processing using RT-DB in a global ADAS.

1) Data collection

We limit the types of relevant data to a cooperative ADAS to three: position, speed and direction of vehicles. This information must be exchanged between vehicles in order to improve road safety. For this purpose, we symbolize each vehicle *V* by a node characterized by the position (X, Y), the speed ($V_{x,y}$) and the direction (Direction) of *V*. Knowing that these three types of data have different rates of evolution, we decided to specialize the messages exchanged in CoopSafeDrivSyst, by defining a message type for each type of data. This avoids redundancy in the sent data; the position of a vehicle can very well change while its speed and direction remain the same. Therefore, the OBU of a vehicle will transmit the values of its position, speed and direction not in the same message, but in different types of messages and with different rhythms. Each message starts with the identification number (V_ID) of the sending vehicle and ends with the time (Instant) when the data is acquired. In addition, (1) the position message contains the coordinates (X, Y) of the position, obtained from the GPS, (2) the speed message contains the speed of the vehicle and (3) the direction message contains the direction of movement of the vehicle.

2) Data model

Adopting the model proposed by [10], we define the data structure by quadruplets d = (d_{value}, $d_{timestamp}$, $d_{duration}$, mde) where d_{value} represents the real value of the considered data, $d_{timestamp}$ is the instant when this value is updated, $d_{duration}$ designates the duration of validity of any value of this data, and mde corresponds to the maximum tolerated error between the real value of the data and the value stored in the RT-DB. For example, the datum d = (90 km/h, 5 h 34 min 23 s, 1 s, 5 km/h) represents the value of the speed of a vehicle which is equal to 90 km/h, which is extracted at 5 h 34 min 23 s, which has a validity time of 1 s, and which tolerates an inaccuracy of 5 km/h. However, we consider that derived data (e.g., acceleration) should not be characterized by the mde field, since they are computed from the sensory data; their update is triggered when one of the sensory data is updated, and therefore, there is no margin error to consider for this type of data. With these quadruplets, a RT-DBMS can manage logical data constraints (d_{value}), temporal constraints ($d_{timestamp}$, $d_{duration}$) and data quality constraints (mde). Indeed, the timestamp and the validity time of each data element are mandatory to verify that a data is valid and not obsolete ($d_{timestamp}$ + $d_{duration}$ ≤current time).

3) Data freshness

As mentioned previously, each vehicle's OBU sends successive values of its position, speed and direction in isolated messages because these pieces of information do not have identical validity times. This is because the data collection process depends on the notion of the validity time of each data. In fact, the Monitor must send a new value for each data type to the Information Selector just before the end of the validity time of its previous value in the RT-DB, in order to keep the data freshness. Note that, the freshness of the data means that their values are not obsolete. It is then a prerequisite for the data values to reflect the current state of the environment. There-fore, the Monitor periodically sends the measured values according to the validity time of each type of data: position, speed and direction. Thus, the time constraints of the data can be respected. However, the Information Selector does not transmit these values to the Data Manager and OBU periodically due to the consideration of data quality.

4) Data Quality

The notion of *Quality of Data* (QoD) is defined to ensure that the values of the data in a RDB correspond to their values in the real world, while accepting a margin of error considered negligible in relation to these values. These data are thus qualified with good quality. For this purpose, we define an *Inaccuracy Threshold* (InacThresh) for each data element, also called the maximum tolerated error, as the difference between the current value of this element in the real world and its last value stored in the RT-DB. This threshold determines if the eventual variation of the value of this element is significant or negligible. If the difference between the acquired value and the value stored in the RDB is greater than the InacThresh, the local update transaction must be executed and the OBU of the vehicle concerned must transmit the value of the data in an appropriate message to the OBUs of the neighboring vehicles, via V2V communications, and to the control center, via a V2I communication through the nearest RSU. Otherwise, the system does not trigger the local update transaction, nor the creation and broadcast of a message for this data. As a result, the number of local transactions and the number of messages sent by a vehicle are reduced. Systematically, a vehicle only receives messages from other vehicles that carry data values with significant variations. The number of messages received by a vehicle and the number of transactions to update external data, i.e., data related to other vehicles, are also reduced. The same applies to the control centers, which only receive messages from the vehicles with significant variations.

B. Design and management of RT-DB

In the following, we first present the structure of the RT-DB to be installed inside each CoopSafeDrivSyst intelligent vehicle and control center, and then, the relational RDBMS that we use to manage these RT-DB. Figures 1 and 2 show the UML[1] class diagrams of our proposed CoopSafeDrivSyst for the vehicles and the control centers, respectively. We describe below the semantics of the first diagram, relating to the "Intelligent Vehicle" component through the presentation of the two classes Vehicle and Speed only, knowing that the classes Position and Direction have structures and behaviors very similar to those of the Speed class.

- **"Vehicle" class:** The vehicles are classified in two categories: concerned vehicles and other vehicles to be controlled. Thus, we defined the two subclasses *ControlledVehicle* and *OtherVehicleToBeControlled* for the generic class *Vehicle* through an inheritance relationship. This class is characterized by the following five attributes: *V_ID* (vehicle identification number), *Brand*, *Model*, *Serial_Num* (serial number) and *Regist_Num* (registration number). Note that we can have only one object (*i.e.*, occurrence), for the subclass *ControlledVehicle* related to the concerned vehicle, whereas we can have several objects, those related to the other vehicles circulating in the surroundings of the concerned vehicle. Each vehicle stores in its local RT-DB the speed, position and direction data that characterize the evolution of its traffic through these methods: *getCurrentSpeed()*, *getCurrentPosition()* and *getCurrentDirection()*. Similarly, it sends this data through the methods *sendSpeed()*, *sendPosition()* and *sendDirection()*, respectively, to the other vehicles (neighboring) and to the control centers. In addition, it records the data received from other vehicles by the methods *getOtherSpeed()*, *getOtherPosition()* and *getOtherDirection()*. Moreover, it receives the commands and the orders from the control centers via getCommand() et getAlert() methods.
- **"Speed" class:** This class represents the speed data stored in the RT-DB. It includes the following attributes: (i) V_Value which represents the value of the obtained data, (ii) V_Timestamp which represents the time of acquisition of this data value, and (iii) V_Duration which indicates its validity time. The Partition constraint (also called exclusive OR) indicates that an occurrence of the Speed class is relative either to an occurrence of the class *ControlledVehicle* or to an occurrence of the class *OtherVehicleToBeControlled*, but not to these two classes at the same time.

[1] Unified Modeling Language

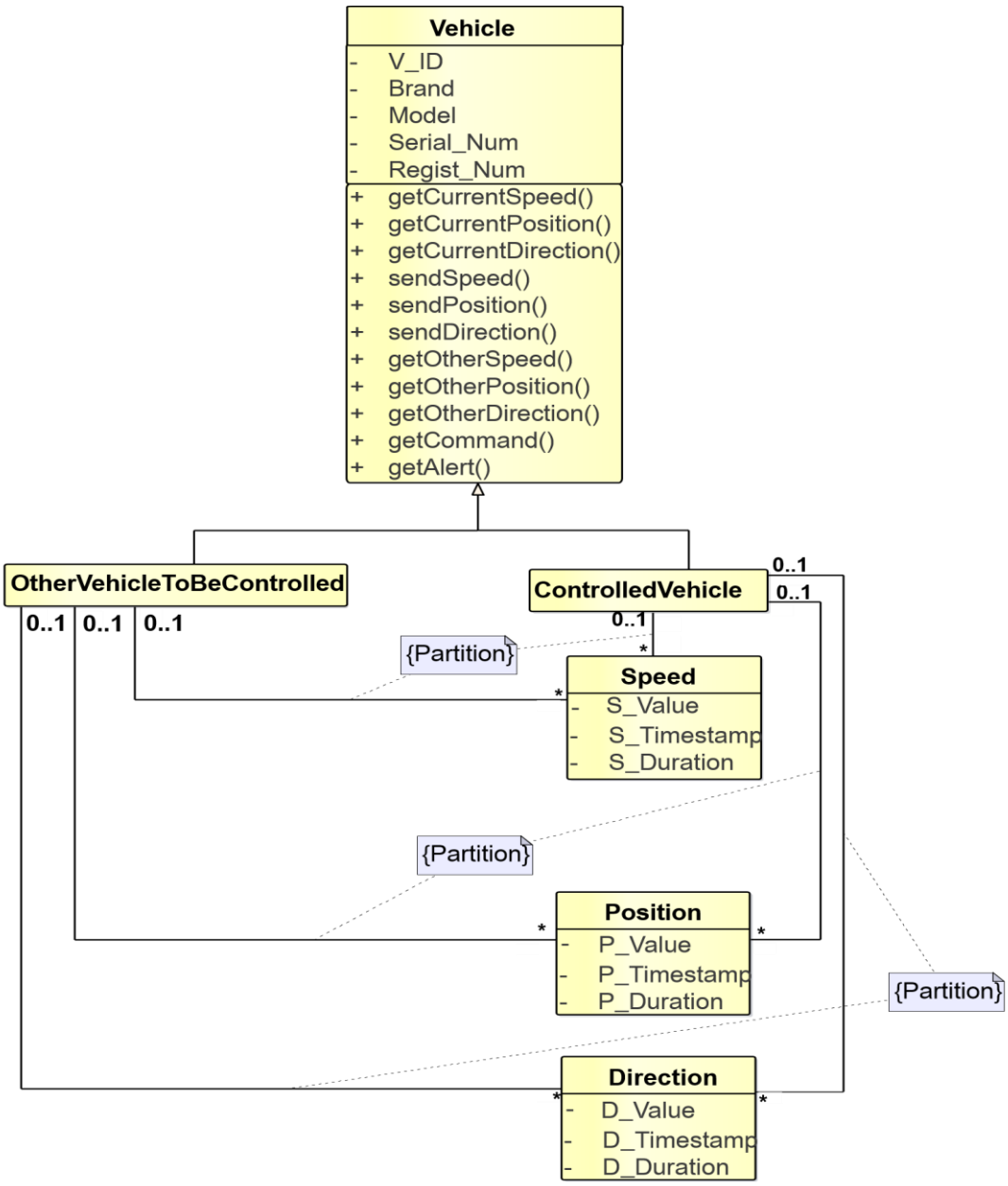

Figure 1. Class diagram of an intelligent vehicle

As for the semantics of the second-class diagram, relating to a control center (*cf.* figure 2), it is described through the presentation of the following four classes, knowing again that the classes *Position* and *Direction* have similar structures and behaviors to the class *Speed*.

- **"ControlCenter" class:** A control center is characterized by three attributes: CC_ID (identification number), Num_Region (number of the region in which it is located), and Area_Region (area of that region). It communicates with one or more RSU to (i) receive data from the vehicles to be stored, and (ii) transmit commands and alerts to the vehicles. Indeed, it receives the data of any vehicle circulating in the region it controls, through an RSU, by the *getSpeed()*, *getPosition()* and *getDirection()* methods. In the same way, it sends, if necessary, commands and alerts to the vehicles of its region, always through an RSU, by the methods *sendCommand()* and *sendAlert()*.
- **"RSU" class:** An RSU has three attributes: RSU_ID that represents its identification number, Num_Zone that represents the number of the zone where it is installed, and Length_Zone that represents the length of that zone. It communicates with several vehicles and with a unique control center. It receives data from the vehicles through the *getSpeed()*, *getPosition()* and *getDirection()* methods, and transfers them to the control center through the *sendSpeed()*, *sendPosition()* and *sendDirection()* methods.
- **"Vehicle" class:** A vehicle is characterized by its identification number, brand, model, serial number and registration number represented by the attributes *V_ID*, *Brand*, *Model*, *Serial_Num* and *Regist_Num*, respectively. While the vehicle is moving, it communicates with the nearest RSU installed within its range. It sends its own data to this RSU via the methods *sendSpeed()*, *sendPosition()* and *sendDirection()*.

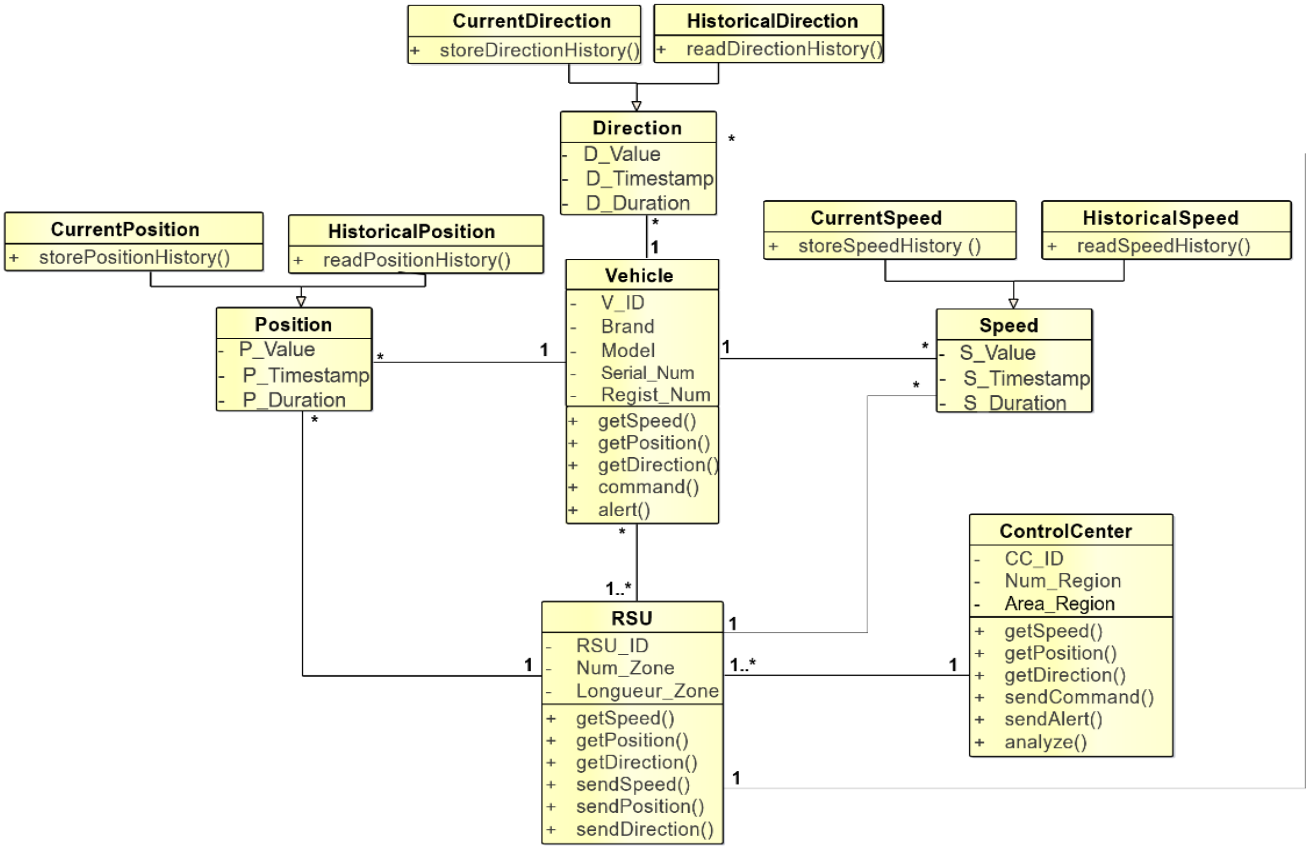

Figure 2. Class diagram of a control center

- **"Speed" class:** This class represents the speed data stored in the control center's RT-DB. It includes the attributes V_Value, V_Timestamp and V_Duration. A vehicle speed stored in the control center can be either a current speed or a historical speed; the CurrentSpeed subclass concerns the real time (i.e., current) speed data of the different controlled vehicles, with a maximum of one object (i.e., one occurrence) per vehicle. It transfers the speed data as soon as it becomes passed, via the *storeHistoricalSpeed()* method, to the *HistoricalSpeed* subclass reserved for past states of speed data.

In order to manage the data from the vehicles' RT-DB and the control center's RT-DB, we need a suitable RT-DBMS. To do so, we choose the SQlite3 RT-DB to implement these RT-DB, as explained in the following.SQLite3 is a relational database engine, embedded, open source and written in C language. It produces lightweight databases that do not adopt the client-server model, but are integrated into the application. Thanks to its lightness, its great control capacity and its speed, SQLite3 is well adapted to real time systems [14]. SQLite3 provides support for the SQL-92 query language standard and ACID (Atomicity, Coherence, Insulation and Durability) properties of transactions. It allows managing many databases with limited resources. Each of them is stored in its own file. It reads and writes directly to the hardware in a different way than traditional SQL databases. Consequently, the data access is faster, easier and completely independent of the platform used, unlike other SQL databases based on the client-server scheme. Since SQLite3 is proven to be an integrated system [14], it is well suited for driver assistance systems, which are constrained by limited resources, including size. To this end, we see that SQLite3 is a good choice for creating databases on many embedded systems, such as those in intelligent vehicles [15]. The local RT-DB created for a vehicle contains some information of the concerned vehicle and the information of adjacent vehicles. As a result, each vehicle has not only its own information, but also those of neighboring vehicles.

IV. Performance metrics for accident prevention systems

In order to evaluate the effectiveness and efficiency of CoopSafeDrivSyst, we introduce the following performance metrics. To do this, we add a specific function to CoopSafeDrivSyst, that we called *Indicator Calculator* (IndicCalc()), to calculate the different metrics.

- **Number of sent messages:** The number of messages sent by a vehicle to neighboring vehicles and the control centers is equal to the number of local update transactions executed in its RT-DB, since the OBU sends data sporadically considering the notion of QoD.

- **Number of exchanged messages:** The number of messages exchanged in the VANET network, with V2V and/or V2I communications, is the total of the number of messages sent by all vehicles and the number of messages received by all vehicles and control centers.

$$NbrExchang = NbrMsgSent + NbrMsgReceived \quad (1)$$

where:

- *NbrMsgSent* is the number of the sent messages,
- *NbrMsgReceived* is the number of the received messages.

• **Number of transactions:** In the RT-DB of an intelligent vehicle (distributed processing), the number of transactions is the sum of the number of local update transactions (*i.e.*, data specific to the vehicle to be controlled) and the number of external data update transactions (*i.e.*, data related to other vehicles to be controlled).

$$NbrTrans = NbrTransLocal + NbrTransExter \quad (2)$$

where:

- *NbrTransLocal* is the number of local transactions,
- *NbrTransExter* is the number of the external transactions.

Note that in the control center's RT-DB (centralized processing), the number of transactions is equal to the total number of data updates received by the vehicles moving in its area.

• **Ratio of lost messages:** To better evaluate and consider the problem of lost messages, we determine the ratio of lost messages. This ratio is the proportion between the number of lost messages and the total number of messages exchanged in the network, as shown in formula 3.

$$LossRatio = \frac{LostMsg}{TotalMsg} \quad (3)$$

where:

- *LostMsg* is the number of lost messages,
- *TotalMsg* is the total number of messages exchanged in the network, *i.e.*, those sent, whether they are received or not.

V. Simulation and results

Due to the fact that testing our system in a real environment requires appropriate infrastructure, complex resources, and high costs, we can only evaluate the performance of our system through simulation, as is the case with the majority of state-of-the-art works. However, we tried to choose the best simulator that could conform to the real environment. In the following, we first present the architecture of the CoopSafeDrivSyst simulation environment. Then, we detail the simulation parameters and scenarios. Finally, we present and analyses the simulation results.

A. Architecture of the simulation environment

To approve the capabilities of our proposed CoopSafeDrivSyst, we simulated road traffic with the VEINS (*Vehicles In Network Simulation*) framework [16]. It inte-grates the road traffic simulator SUMO (*Simulation of Urban MObility*) [17] and the discrete event-based network simulator OMNET++ [18]. Furthermore, we propose to use SQLite3 in order to evaluate the benefits of integrating an RT-DBMS into the cooperative ADAS. Figure 3 shows the architecture of the simulation environment.

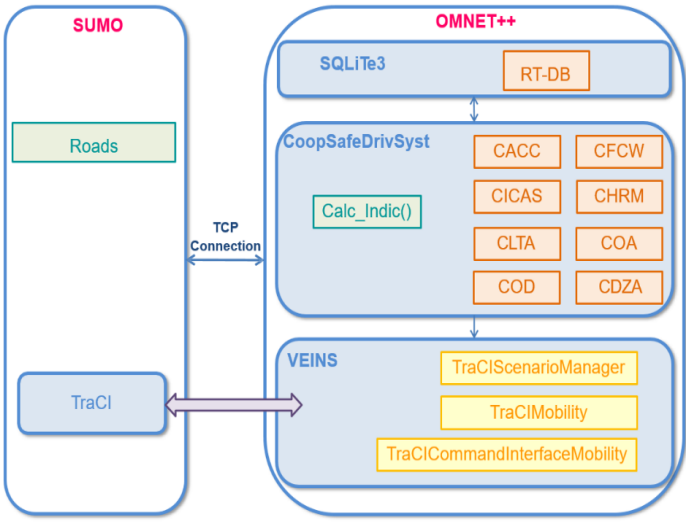

Figure 3. Architecture of the simulation environment

The main advantage of using VEINS is its ability to generate real-time interaction between the network simulation and the road traffic simulation. In addition, it sup-ports the simulation of wireless communication protocols in VANET and supports the 802.11p standard. Moreover, it relies on the DSRC/WAVE technology [19] which encompasses both V2I and V2V communication modes.

For traffic simulation with VEINS, SUMO provides very powerful simulations of massive networks with roads composed of multiple lanes. Each vehicle is defined by an identifier and described by physical properties. In addition, the properties characterizing the traffic of a vehicle (*i.e.*, position, speed, lane to use, etc.) can also be de-fined with SUMO. VEINS works as follows: first, SUMO generates and then exports vehicles to the OMNET++ network simulator, which considers each vehicle as a node. OMNET++ and SUMO, connected via TCP connections, operate in parallel. The *Traffic Control Interface* (TraCI) [20] is used as the communication protocol. Following each change in vehicle movement, OMNET++ sends the information to SUMO. The latter modifies the vehicle movement and sends the position and other information back to OMNET++. The modules TraCIMobility and TraCIScenarioManager of VEINS use API calls to interact directly with the traffic simulator (SUMO). TraCIScenarioManager connects OMNET++ to the TraCI server running SUMO simulations. Mobile nodes (*i.e.*, vehicles) are configured and controlled using the TraCIMobility module.

Therefore, the simulation environment that we built integrates CoopSafeDrivSyst into VEINS. We have extended the VEINS framework with modules to manage data collection, freshness and distribution, while ensuring road safety. We have adopted an integral development method without resorting to a prior generation of code parts. In fact, we have developed the code of our global system CoopSafeDrivSyst and its different components using object-oriented programming, with the objective of verifying the validity of their functionality under the VEINS simulation framework. Indeed, CoopSafeDrivSyst and the three subsystems it integrates (*i.e.*, CCPPS, COAPS, and CRHDPS) are implemented by (i) the C++ classes we

developed to deploy the eight cooperative ADAS modules: CACC, CFCW, COA, CICAPS, CLTA, CHRM, CDZA and COD, (ii) the C++ methods we have developed to implement formulas, conditions and rules, (iii) the Calc-indic() function that calculates performance metrics, and (iv) SQLite3 that we integrated into the OMNET++ IDE to create a local RT-DB in each vehicle and an RT-DB in each control center. CoopSafeDrivSyst is implemented with relational RT-DB defined by transforming the classes and relations of the class diagrams in Figures 1 and 2.

Relational diagram of the class diagram Vehicle:
Vehicle (V_ID, Brand, Model, Num_Regist, Regist_Num, Type)
Speed (#V_ID, S_Timestamp, S_Value, S_Duration)
Position (#V_ID, P_Timestamp, P_Value, P_Duration)
Direction (#V_ID, D_Timestamp, D_Value, D_Duration)

Relational diagram of the class digram Control Center:
Vehicle (V_ID, Brand, Model, Num_Regist, Regist_Num)
Speed (#V_ID, S_Timestamp, S_Value, S_Duration, #RSU_ID)
Position (#V_ID, Timestamp, P_Value, P_Duration, #RSU_ID)
Direction (#V_ID, D_Timestamp, D_Value, D_Duration, #RSU_ID)
RSU (RSU_ID, Num_Area, Area_Length, #CC_ID)
Control_Center (CC_ID, Num_Region, Area_Region)
RSU-Vehicle (#RSU_ID, #V_ID)
ControlCenter-RSU (#CC_ID, #RSU_ID)

For each node in OMNET ++ (*i.e.*, vehicle), we extract its index and create a local RT-DB for the vehicle (*i.e.,* the RT-DB *vehicle1.db* for the vehicle identified by ID-1, for example). Similarly, we group the roads by region and identify each control center by its region. For example, *center1.db* corresponds to the RT-DB of the control center in region 1.

As mentioned earlier, VEINS has many predetermined methods that meet our needs. These methods are mentioned below with *italic style*, while CoopSafeDrivSyst methods are mentioned with normal style. Along with TraCIMobility and TraCIScenarioManager modules, the TraCICommandInterface module uses various methods to obtain vehicles, RSU and road information. For example, we use the *getSpeed()*, *getCurrentPosition().x* and *getCurrentPosition().y* methods to get the speed, the coordinate X and the coordinate Y, respectively. However, there is no existing method that determines the direction of the vehicle. Thus, we propose a new method, called getDirection(), to determine the direction of the vehicle based on the lane identification according to the *getLaneId()* method. Each vehicle saves its initial data obtained with the previous methods, while it sends speed, position and direction messages to the neighboring vehicles by CoopSafeDrivSyst's methods sendSpeed(), sendPosition() and sendDirection, respectively. In addition, a vehicle to be controlled receives driving data from neighboring vehicles through CoopSafeDrivSyst's methods getOtherSpeed(), getOtherPosition() and getOtherDirection().

VEINS uses WSM (*Wave Short Message*) to transfer the messages. The data are transmitted by the method *setWsmData(message)* and sent by the method *sendDown()*.When receiving a message, CoopSafeDrivSyst of a vehicle extracts this data and stores it in the appropriate table, both for the vehicle's own information and for the information of surrounding vehicles. All these methods are called through the VEINS method *handlePositionUpdate (cObject* obj)*.

B. Basic considerations for the simulations

The simulations we have considered adopt the following basic considerations. The vehicles are arranged on the roads of the map. Each vehicle moves randomly along the road in a certain direction. It leaves the simulation when it reaches its destination. Table 1 lists the parameters employed by SUMO to represent vehicles and some mobility characteristics in the scenarios of our simulations. The most used parameters of the network simulator are shown in Table 2. We experimented the different subsystems in a sparse network and varied the number of vehicles from two to ten. The vehicles must be within the same communication range to exchange messages. We have considered that the transmission range is 500 meters for all vehicles.

Parameters	**Values**
Maximum speed of a vehicle	30 m/s
Acceleration of a vehicle	[0.3 m/s2, 2 m/s2]
Deceleration of a vehicle	[1 m/s2, 4.5 m/s2]
Length of a vehicle	5 m

Table 1. SUMO parameters.

Parameters	**Values**
Number of vehicles	[2..10]
Range of the radio transmission	500 m
Type of vehicle mobility	TraCIMobility
Typical application for vehicles	TraCIDemo11p

Table 2. Configuration of simulations.

C. Basic considerations for the simulations

The Monitor sends the data to the Information Selector, respecting the validity interval of each data. We set the validity time interval of each data by relying on a mobility file generated by the SUMO simulator. This file shows the variations of the position and speed, along with other information, of each vehicle in the network at each time step (*e.g.*, every 100 ms). Faced with this rate of new data supply, we define the validity time of each data item as follows: (i) 1 s for the vehicle's speed, (ii) 100 ms for its position (*i.e.*, coordinate X and coordinate Y), and (iii) 5 s for the vehicle's direction. If the data stored in the RT-DB must respect their validity interval, the notion of QoD comes to alleviate this constraint and thus reduce the frequency of updates. In fact, cooperative ADAS, like any real-time system, can tolerate a certain degree of inaccuracy. The data are updated only if the difference between the value of the data stored in the database and the acquired one is higher than an inaccuracy threshold that we determine by simulations according to the driving conditions. As an example, we note that the values of the speed of a vehicle do not vary much under normal conditions. Therefore, it is unnecessary to periodically update the speed value every period (*i.e.*, every second). In this purpose, we have set for each type of data (velocity and position) values for the *Inaccuracy Threshold* (InacThr) that

alleviate the frequency of updates without affecting the security of the system. We set the speed InacThr ($InacThr_{Speed}$) at either 1 m/s or 2 m/s, while we set the position InacThr ($InacThr_{Position}$) at either 1 m, 2 m or 3 m. Indeed, we have performed multiple simulations of our system under these different values of $InacThr_{Speed}$ and $InacThr_{Position}$.

To better demonstrate the effects of applying the notion of QoD on reducing the number of update transactions (executed by the Information Selector), let us examine the example of three vehicles V1, V2 and V3 that circulate on the road. Table 3 shows the values of the collected speed data for these three vehicles at time t = 47 s, while Table 4 shows the data following the update performed by an RT-DBMS adopting a speed data validity interval equal to 1 s and an $InacThr_{Speed}$ equal to 1 m/s. In this example, we distinguish two cases:

- The case of the vehicle V1, for which the difference between the speed value acquired at time t = 47 s and the value recorded in the RT-BD is greater than the $InacThr_{Speed}$ value. Thus, the value at t = 46 s is replaced by the value acquired at t = 47 s (*cf.* Table 4).
- The case of the vehicles V2 and V3 for which the speed update is not necessary. Indeed, (i) the speed values of V2 are identical at time t = 46 s and at time t = 47 s, and (ii) the difference between the speed value measured at t = 47 s and the value stored in the RT-DB of the vehicle V3 is less than the $InacThr_{Speed}$ value. Consequently, the system may well proceed to use the values stored in the RT-DB at time t = 24 s.

Data	Values
Speed_V1	16 m/s
Speed_V2	12 m/s
Speed_V3	21 m/s

Table 3. The values of the data acquired at t = 47 s.

Data	Value	Timestamp	Duration
Speed_V1	~~16 m/s~~ 17.2 m/s	~~46 s~~ 47 s	One second
Speed_V2	12 m/s	46 s	One second
Speed_V3	21.6 m/s	46 s	One second

Table 4. Example of data update using an RT-DB at t = 47 s.

D. Simulation scenarios

We performed several simulations of various scenarios to show the advantage of integrating the RT-DB and applying the notion of QoD. For each scenario, we perform three experimental cases:

- **Case of periodic data sending:** we force CoopSafeDrivSyst to send periodic messages with a period of 100 ms, without taking into account the notions of data validity and QoD, as is the case of cooperative ADAS in the literature.
- **Case of sending in accordance with the data validity intervals:** we let our system use a data validity interval for each data item (100 ms for position, 1 s for velocity and 4 s for direction).
- **Case of sending according to validity intervals and data quality:** we let CoopSafeDrivSyst use the notion of QoD; with different InacThr for position and speed data.

E. Simulation results

1) Evaluation of the effectiveness of the system

We begin by evaluating CoopSafeDrivSyst's performance in terms of accident avoidance, following its three sub-systems. So, we effectuated three sets of simulations.

In the first set of simulations, we evaluated the *Cooperative Road Hazard Detection Persistent Sub-system* (CRHDPS). We simulated a scenario where an intelligent vehicle identifies a static obstacle in its trajectory and overtakes it securely. Later, it detects a work zone, therefore it changes its path to avoid crashes with other vehicles. We confirmed through the simulation results of this scenario that CRHDPS accurately examined these two dangerous situations, executed the suitable decisions, and triggered the proper component for each situation to avoid accidents. In the second set of simulations, we evaluated the performance of the *Cooperative Overtaking Assistance Persistent Sub-system* (COAPS). To do so, we have simulated scenarios, where vehicles are confronted with different overtaking situations. In some cases, overtaking is allowed, but in other cases, overtaking is not allowed because of the accident danger. We proved that COAPS could interpret both situations and take the appropriate decisions while overtaking maneuvers. In the third set of simulations, we evaluate the *Cooperative Collision Prevention Persistent Sub-system* (CCPPS) sub-system. We tested a situation where an intelligent vehicle crosses an intersection, then a T-intersection, and lastly a highway on-ramp merging. We concluded that CCPPS has always completed correctly the analysis of the different situations, the triggering of the right component for each situation, and the taking of the appropriate decisions to guarantee secure passage.

The simulation proofs are shown in the annex accessible in: [https://drive.google.com/file/d/1GZhtpVEzeXYXPINRg4GZFy35yOFd9C4D/view]

2) Evaluation of the RT-DB integration

In the following, we evaluate and analyze the effect of integrating RT-DB into our overall CoopSafeDrivSyst system, as well as the effects of applying data validity and QoD concepts. For space constraints, we demonstrate in the following the results concerning only the CRHDPS sub-system. Note that we obtained for the CCPPS and COAPS sub-systems very similar results.

(a) Variation in the number of messages sent by a vehicle

We begin by evaluating the contribution of RT-DB integration as well as the notions of data validity and QoD on the number of messages sent by an intelligent vehicle, labelled HostVeh. To this end, we simulate a scenario where the HostVeh detects a static obstacle on the road, with another vehicle following it. In this scenario, the IndicCalc function calculates the number of messages sent by the HostVeh that transfers its speed, position and direction data, in separate messages, to the other vehicle. We present in Figure 4 the results related to the variation of the number of messages sent by the HostVeh, for the three types of experiments mentioned above.

- **First experiment:** CRHDPS transmits data periodically with a period of 100 ms. We notice that the value of the

number of messages sent is high (red graph). This number is equal to 490 messages.

- **Second experiment:** CRHDPS considers the duration of validity of each type of data (*i.e.*, 100 ms for position, 1 s for speed and 4 s for direction). We observe that this number has decreased (blue graph). It is reduced from 490 to 215 messages.
- **Third experiment:** COD additionally considers the notion of QoD by applying different values of the inaccuracy thresholds for the position (SI-P) and speed (SI-V) data. We note that the number of messages sent by the VehHot has been reduced more and more (graphs in other colors). It is reduced from 215 to 55 messages, with SI-P equal to 3 m and SI-V equal to 2 m/s (yellow graph).

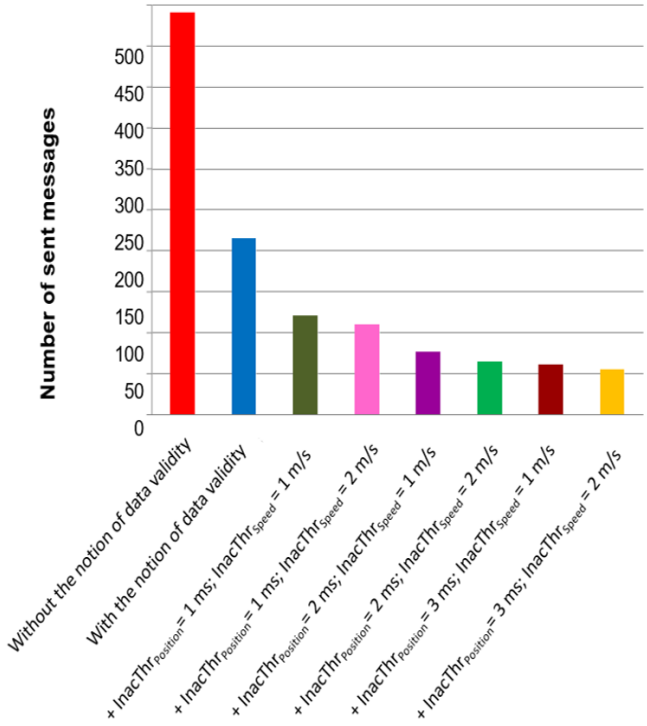

Figure 4. Variation in the number of sent messages by a vehicle, in CRHDPS

Moreover, the Figure 4 shows that the number of messages sent by the HostVeh decreased lightly when we fixed the InacThr$_{Position}$ but we varied the InacThr$_{Speed}$ from 1 m/s to 2 m/s. For example, when InacThr$_{Position}$ is equal to 3 m, it is reduced from 61 with InacThr$_{Speed}$ equal to 1m/s (seventh graph) to 55 messages with InacThr$_{Speed}$ equal to 2m/s (last graph). In contrast, we observe that this number decreased significantly when we fixed InacThr$_{Speed}$ but varied the values of InacThr$_{Position}$ from 1 m to 2 m or 3 m. When InacThr$_{Speed}$ is set to 2 m/s, for example, it is reduced from 110 with an InacThr$_{Position}$ equal to 1 m (fourth graph) to 55 messages with an InacThr$_{Position}$ equal to 3 m (last graph). This variation in the amplitude of the reductions is explained by the fact that the vehicle maintains a speed that normally varies only slowly; this speed varies only in particular situations, such as when the vehicle detects a static object on the road that requires its deceleration or stopping. But, on the contrary, its position varies in a rapid way. The improvements obtained by the second and third experiments can be explained by the fact that the OBU of the HostVeh avoids sending a message when the variations of the position and speed data, in two successive moments, are minimal. In fact, it transfers messages according to the notions of data validity and QoD.

(b) Variation in the number of messages exchanged

Let us now evaluate the effect of introducing the notions of data validity and QoD on the variation of the overall number of messages exchanged in VANET, *i.e.*, between vehicles between themselves, on the one hand, and between vehicles and the control center via the nearest RSU deployed in the portion of road containing the danger, on the other. The importance of the variation in the overall number of messages exchanged is measured according to the number of vehicles. Figure 5 illustrates the results for the number of messages exchanged, for all scenarios with the three types of experiments, also by varying the number of vehicles running from 2 to 10.

- **First experiment:** When CRHDPS does not consider the notion of data validity (periodic sending every 100 ms, as it is the case in the related works), we notice that the number of messages exchanged is high (red graphs), and increases remarkably with the increase of the number of vehicles. Such a number reaches about 3000 messages for ten vehicles on the road.
- **Second experiment:** When CRHDPS considers a validity time for each type of data, we notice that these numbers are significantly reduced (blue graphs). Such a number is reduced to about 2220 messages in the case of ten vehicles.
- **Third experiment:** When CRHDPS considers the notion of data validity and QoD with different InacThr$_{Position}$ and InacThr$_{Speed}$ inaccuracy thresholds, we notice that the numbers of messages exchanged become smaller and smaller (graphs in other colors). Such a number decreased to about 980 messages for the case of ten vehicles.

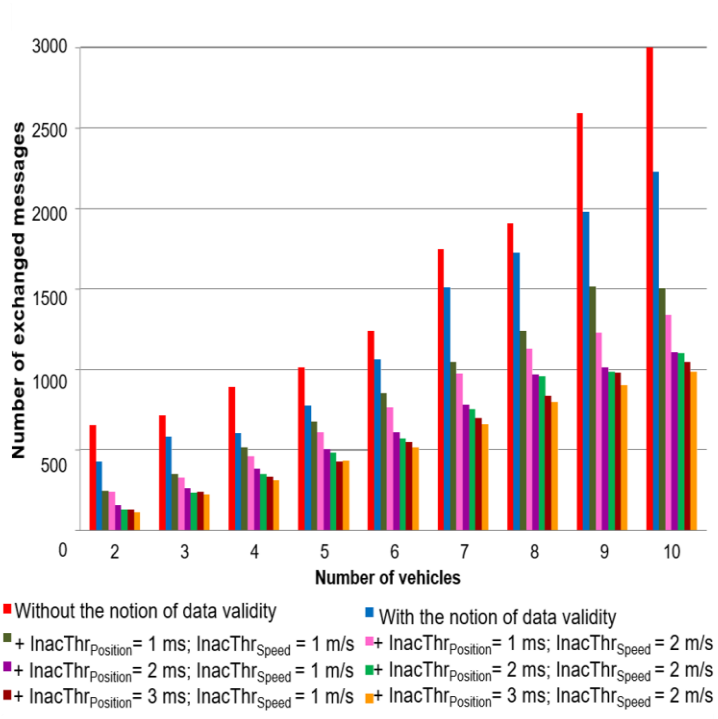

Figure 5. Variation of the number of messages exchanged between vehicles, in the CRHDPS sub-system

Thus, these results confirm the fact that the use of RT-DB with the notions of data validity and QoD considerably reduces the number of messages exchanged for the CRHDPS sub-system in VANET, both V2V, *i.e.*, between the vehicles themselves, and V2I, *i.e.*, between the vehicles and the control center.

(c) Variation in the number of transactions

We present now the contribution of the notions of data validity and QoD in terms of reducing the total number of update transactions in the TR-DB of the vehicles and control centers. The simulations presented here are carried out in the framework of the treatments realized in a HostVeh by the CRHDPS sub-system. Note that the number of update transactions is calculated by formula 2 using the IndicCalc () function. Figure 6 visualizes the results of the same scenarios and the same three types of experiments, discussed previously, in terms of total numbers of update transactions.

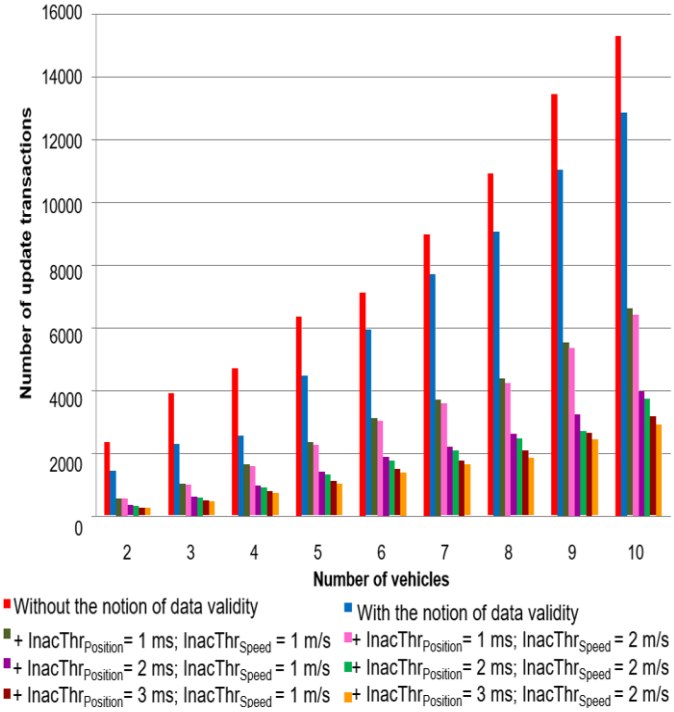

Figure 6. Comparison of the number of transactions, in the CRHDPS sub-system

- **First experiment:** We can see from the red graphs that the number of transactions is high, especially when the number of vehicles increases, since each vehicle collects and sends the three data of position, speed and direction, in a periodic way every 100 ms. Thus, these three data are periodically updated every 100 ms in the corresponding RT-DB. The number of update transactions exceeds 15000 for the case of ten vehicles on the road.
- **Second experiment:** We observe from the blue graphs that the number of transactions is quite reduced, since the data stored in the RT-DB do not need to be updated when they remain valid during their validity intervals. Such a number is reduced from about 15000 transactions to about 12800 transactions, for the case of ten vehicles.
- **Third experiment:** We see from the graphs of the other colors in Figure 6, the interest of introducing the notion of QoD, using different values of the InacThr for position (InacThr$_{Position}$) and speed (InacThr$_{Speed}$), in addition to the notion of data validity. We observe that the number of update transactions has been considerably reduced since a position or speed data is updated only if the difference between its current value and the one stored in the RT-DB is higher than the InacThr. Such a number is reduced to about 2910 transactions for the same case of ten vehicles, with an InacThr$_{Position}$ equal to 3 m and an InacThr$_{Speed}$ equal to 2 m/s (yellow graph). In fact, in this third experiment, the position and speed data stored in the RT-DB were not updated periodically, but rather sporadically according to the validity times of the data and the values of the InacThr$_{Position}$ and InacThr$_{Speed}$.

It is important to mention that the considerable reduction in the number of transactions certainly leads to a beneficial reduction in the processing time of the various operations.

(d) Variation in the number of lost messages ratio

We now study by simulation the effect of the use of the RT-DB and the notions of data validity and QoD on the ratio of lost messages in the CRHDPS sub-system. Note that the IndicCalc() function calculates this ratio according to the formula 3. Figure 7 shows the successive improvements obtained from the results of the simulations conducted according to the three types of experiments.

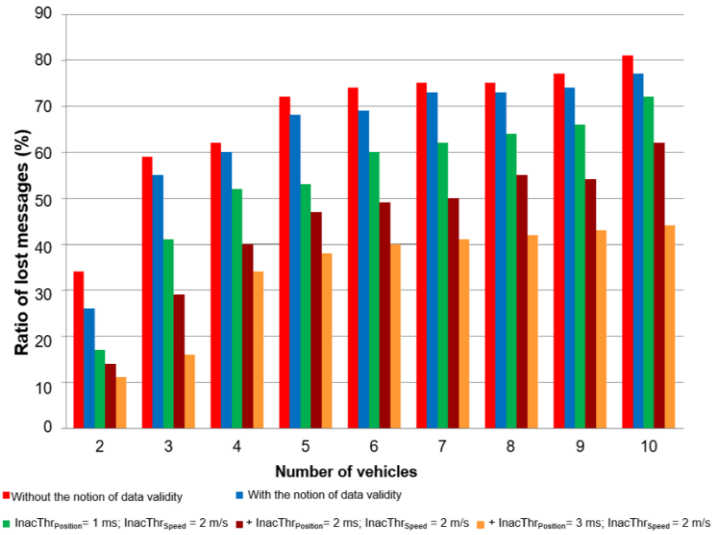

Figure 7. Variation of the ratio of lost messages, in the CRHDPS sub-system

Without using the notion of data validity for the three types of data, we see that the ratio of lost messages is equal to 81% (red graph) for the ten-vehicle case. Using the notion of data validity, we see a slight improvement, with a value of 77% (blue graph) for the case of ten vehicles. With the use of the notion of QoD, applying different values of the InacThr for position and speed data, we see that these improvements continue. With an InacThr$_{Speed}$ equal to 2 m/s and an InacThr$_{Position}$ equal to 3 m, for example, this ratio decreased, quite considerably (a little less than half), to 44% for the same case of ten vehicles. Note also that we obtained quite similar improvements in this ratio when we used an InacThr$_{Speed}$ equal to 1 m/s (instead of 2 m/s). As a result, we have contributed to the improvement of the message loss ratio, and thus the improvement of the performance of the COD component, by implementing the notions of data validity and QoD with adequate InacThr for position and speed. While this improvement seems not sufficient, as this ratio is reduced at best to only 44% for the case of ten vehicles (and 11% for the

case of two vehicles), there are other issues to address the message loss problem in VANET and reduce the loss rate [21] [22].

Despite this limitation, the results of the conducted simulations confirm that the CRHDPS sub-system manages to ensure accident avoidance even with lost messages.

3) Summary

On the one hand, the results of the different simulations performed confirm to us (i) the robustness of our CoopSafeDrivSyst in avoiding the different type of road accidents, without the intervention of the driver, and (ii) the ability of its Reaction Units to act with much reduced reaction times. This is due to the simplification and acceleration of the calculation and processing operations using the RT-DB.

In addition to validating CoopSafeDrivSyst in terms of effectiveness and efficiency, we have shown the usefulness of applying the concepts of data validity and QoD. The results of the different simulations carried out showed us that the number of messages sent by each vehicle and those exchanged between vehicles and between vehicles and control centers, within the VANET, were remarkably reduced. This correlates with the decrease in the number of update transactions in both the vehicle and the control center databases. Furthermore, we observed that the RT-DB, through the notions of data validity and QoD, mitigated the message loss problem under VANET, reducing the loss rate and ensuring the robustness of our system operation despite some message losses. Therefore, the simulation results confirmed that the integration of an RT-DBMS in all the sub-systems of CoopSafeDrivSyst allowed, on the one hand, improving the reliability of this system, and, on the other hand, to manage well the large amount of data, collected by the sensors and distributed via the V2V and V2I communications, and their temporal constraints.

VI. Conclusion

Road accidents are the principal problem producing deaths, injuries, and material destruction. Therefore, numerous researches have proposed systems based on Vehicular Ad hoc NETworks (VANET) to improve road safety. Nevertheless, these systems have weaknesses and problems: high computation time, and data loss, etc. To anticipate road accidents, we have proposed a new system, called Cooperative Safe Driving System (CoopSafeDrivSyst) based on VANET. CoopSafeDrivSyst combines the different functionalities of the cooperative ADAS objects and synchronizes them to make more reliable decisions and actions. Moreover, CoopSafeDrivSyst can automatically deal with all situations along the vehicle trajectory, without the requirement of driver interaction. Furthermore, CoopSafeDrivSyst integrates RealTime DataBases, in each vehicle and the in the control center. Therefore, our system can efficiently handle a huge amount of data, decrease the number of transferred messages and accelerate decision-making and reaction.

In our future work, we aim to enhance CoopSafeDrivSyst performance, on the one hand, and implement a Temporal DataBases in the control center to manage historical data and avoid the loss of the old data values with non-destructive updates, on the other hand. Therefore, it will become possible to develop data mining processes about road risks and their outcomes, to make great decisions to improve the road safety.

References

[1] Association for safe international road travel. http://www.asirt.org/, last accessed 2021/04/25.

[2] A. Koesdwiady, R. Soua, F. Karray, M. Kamel. “Recent Trends in Driver Safety Monitoring Systems: State of the Art and Challenges”, *IEEE Transactions on Vehicular Technology*, 66 (6), pp. 4550–4563, 2016.

[3] M. Dongre, G. Narendra “Effective Road Model for Congestion Control in VANETs”, *International Journal of Wireless & Mobile Networks (IJWMN)*, 8 (2), pp. 11-21, 2016.

[4] I. Elleuch, A. Makni, R. Bouaziz. “Cooperative Advanced Driver Assistance Systems: A Survey and Recent Trends”, *20th International Conference on Intelligent Systems Design and Applications (ISDA)*, pp. 1-11, 2020.

[5] I. Elleuch, A. Makni, R. Bouaziz. “Towards a Cooperative Intelligent System for Unpredictable and Predictable Road Hazard Detection”, *International Journal of Computer Information Systems and Industrial Management Applications*, 11(2019), pp. 101-114, 2019.

[6] I. Elleuch, A. Makni, R. Bouaziz. “Cooperative Intersection Collision Avoidance Persistent System Based on V2V Communication and Real-Time Databases”, *In the proceedings* of *IEEE/ACS 14th International Conference on Computer Systems and Applications (AICCSA)*, pp. 1082-1089, 2016.

[7] I. Elleuch, A. Makni, R. Bouaziz. “Cooperative Overtaking Assistance Persistent System Based on V2V Communications and RTDB”, *The Computer Journal*, 62(10), pp. 1426-1449, 2019.

[8] I. Elleuch, A. Makni, R. Bouaziz. “Design of an Intelligent Cooperative Road Hazard Detection Persistent System”, *In Proceedings of the International Conference on Intelligent Systems Design and Applications (ISDA)*, pp. 696-707, 2018.

[9] Y. Yong. “A memory-resident object store”. PhD thesis, University of Rhode Island, 1997.

[10] N. Idoudi, C. Duvallet, B. Sadeg, R. Bouaziz, F. Gargouri. “Structural Model of Real-Time Databases: An Illustration”, *11th IEEE International Symposium on Object Oriented Real-Time Distributed Computing (ISORC)*, pp. 58–65, 2008

[11] M. Franceschinis, L. Gioanola, M. Messere, M. Tomasi, M. Spirito, P. Civera. “Wireless Sensor Networks for Intelligent Transportation Systems”, *In Vehicular Technology Conference (VTC Spring 2009)*, pp. 1–5. IEEE, 2009.

[12] M. Friesen, R. Jacob, P. Grestoni, T. Mailey, M. Friesen, M. McLeod. “Vehicular Traffic Monitoring Using Bluetooth Scanning over a Wireless Sensor Network”, *Canadian Journal of Electrical and Computer Engineering*, 37(3), pp. 135–144, 2014.

[13] H. Marouane. “Contribution à la modélisation des applications temps réel d’aide à la conduit”. PhD thesis, Université du Havre, 2015.

[14] W. Shuai, L. Jian, J. Shuming, W. Zhiqiang, Z. Jianfeng, X. Shijie. “Study of Real-time Data Collection and Release with In-vehicle System”, *In International Conference on Computer Application and System Modeling (ICCASM)*, pp. 1472–1475, 2012.

[15] C. Bi “Research and application of SQLite embedded database technology”, *WSEAS Transactions on Computers*, 1(8), pp. 83–92, 2009.

[16] C. Sommer, R. German, F. Dressler. “Bidirectionally coupled network and road traffic simulation for improved IVC analysis”, *IEEE Transaction on Mobile Computing*, 10(3), pp. 3-15, 2011.

[17] M. Behrisch, L. Bieker, J. Erdmann, D. Krajzewicz. “Sumo-Simulation of Urban Mobility”, Third International Conference Advances in System Simulation (SIMUL), pp. 55–60, 2011.

[18] A. Varga, R. Hornig. “An Overview of the OMNeT ++ Simulation Environment”, International Conference on Simulation Tools and Techniques for Communications, Networks and Systems & workshops, pp. 1–10. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2008.

[19] Y.J. Li. “An Overview of the DSRC/WAVE Technology”, *International Conference Heterogeneous Networking for Quality, Reliability, Security and Robustness*, pp. 544–558, 2010.

[20] A. Wegener, M. Piórkowski, M. Raya, H. Hellbrück, S. Fischer, J.P. Hubaux. “TraCI: An Interface for Coupling Road Traffic and Network Simulators”, *Communications and Networking Simulation Symposium*, pp. 155–163, 2008.

[21] J.P. Hespanha, P. Naghshtabrizi, Y. Xu. “A survey of recent results in networked control systems”, *Proceedings of the IEEE*, 95(1), pp. 138–162, 2007.

[22] W. Lai, W. Ni, H. Wang, R.P. Liu. “Analysis of average packet loss rate in multi-hop broadcast for VANETs”, *IEEE Communication Letter*, 22, 157–160, 2018.

Author Biographies

Islam Elleuch received her PhD degree in Computer Science from the Faculty of Economics and Management of Sfax, Tunisia in 2019. She also obtained her Engineering degree in Computer Science from the National Engineering School of Sfax, Tunisia in 2014. She is currently a researcher and a member of Multimedia, InfoRmation systems and Advanced Computing Laboratory (MIRACL). Her current research interests include intelligent transportation systems, cooperative advanced driver assistance systems and real-time databases.

Achraf Makni is associate professor on computer science at the National School of Electronics and Telecommunications of Sfax, Sfax University, Tunisia. His PhD has dealt with concurrency control in temporal databases. Currently, his main research topics of interest are temporal databases, real-time databases and intelligent transportation systems.

Rafik Bouaziz is full professor on computer science at the Faculty of Economic Sciences and Management of Sfax University, Tunisia. He was the president of this University during August 2014 – December 2017, and the director of its doctoral school of economy, management and computer science during December 2011 – July 2014. His PhD has dealt with temporal data management and historical record of data in Information Systems. The subject of his accreditation to supervise research was “A contribution for the control of versioning of data and schema in advanced information systems”. Currently, his main research topics of interest are temporal databases, real-time databases, information systems engineering, ontologies, data warehousing and workflows.