Efficient Data Processing in Software-Defined UAV-Assisted Vehicular Networks: A Sequential Game Approach

Ahmed Alioua, Sidi-Mohammed Senouci, Samira Moussaoui, Hichem Sedjelmaci, Mohamed-Ayoub Messous

Efficient Data Processing in Software-Defined UAV-Assisted Vehicular Networks: A Sequential Game Approach

Ahmed Alioua[1] · Sidi-Mohammed Senouci[2] · Samira Moussaoui[1]
Hichem Sedjelmaci[3] · Mohamed-Ayoub Messous[2]

Abstract In large scale networks like Vehicular Ad-hoc Networks (VANETs), the full coverage of fixed infrastructure is hard to ensure, making network management difficult. Whether in infrastructure-less environments where the network connectivity is poor or where the infrastructure deployment is difficult, costly or not profitable. Recently, in the one side, Unmanned Aerial Vehicles (UAVs) have been used as a new flexible solution to assist infrastructure-less vehicular networks for the investigation of inaccessible areas. In the other side, several works have shown interest in the use of the emerging network paradigm of Software-Defined Networking (SDN) to facilitate the management and improve the performances of vehicular networks. In this paper, we propose a novel distributed SDN-based architecture for UAV-assisted infrastructure-less vehicular networks. The main goal is to fill the gap that no SDN-based architecture has been proposed for these networks. We focus particularly on a road safety use-case that incorporates UAVs to assist emergency vehicles in the exploration of affected zones in critical emergency situations. Moreover, we investigate how to achieve efficient data processing policy through a computation offloading/sharing decision-making problem. The main challenge is to reach the best tradeoff between computation delay and energy consumption for computation-

✉ Ahmed Alioua
aalioua@usthb.dz

Sidi-Mohammed Senouci
Sidi-Mohammed.Senouci@u-bourgogne.fr

Samira Moussaoui
smoussaoui@usthb.dz

Hichem Sedjelmaci
hichem.sedjelmaci@irt-systemx.fr

Mohamed-Ayoub Messous
ayoub.messous@u-bourgogne.fr

[1] Computer Science Department, RIIMA Laboratory, USTHB University, Algiers, Algeria

[2] DRIVE EA1859, University of Bourgogne Franche-Comte, 58000 Nevers, France

[3] Cyber Security, IRT System X, Paris, France

intensive tasks in a delay-sensitive context. We formulate this decision problem as a two-player sequential game approach and design distributed computation algorithms to solve the problem. Numerical results show that data processing policy of distributed offloading/sharing algorithms achieves efficient computation performances in terms of delay and energy whilst ensuring until 28% gain of system cost and 95% better response time, compared to native computation scenarios and related data delivery UAV-assisted VANET works, respectively.

Keywords Vehicular Ad-hoc Network · Infrastructure-less vehicular area · Unnamed aerial vehicle · Software-Defined Networking · Data processing · Sequential game

1 Introduction

Propelled by the long traffic congestions and high road accidents, Vehicular Ad-hoc Networks (VANETs) have attracted a lot of interests in the last decade for the propose to make the traveling experience more pleasant, the road more safe and transportation system more efficient. Nowadays, VANET architectures suffer from scalability issues since it is difficult to deploy services in a large-scale, dense and dynamic topology. These architectures are rigid, difficult to manage and suffer from a lack of flexibility and adaptability in control. These constraints limit system functionalities, slow down creativity and often lead to under-exploitation of network resources.

In the last few years, the emerging network architecture paradigm of Software-Defined Networking (SDN) has become one of the most important technologies to manage large scale networks. SDN is mainly based on a physical separation between control plane, i.e., network control features, and data plane, i.e., data forwarding features, and a logically centralized control and intelligence in a software controller. Several works in [1–11] have shown interest in the use of the promising SDN paradigm to address the undermentioned challenges of VANETs. SDN can be used to bring flexibility, scalability, and programmability to VANETs as it can exploit the available network resources more efficiently and, hence, introduces new services. However, the quasi-majority of the proposed SDN-based architectures are infrastructure-based. Unfortunately, the total coverage of fixed infrastructure is not yet reached in current VANET systems. Therefore, uncovered vehicular areas (i.e., infrastructure-less zones) still exist and should be considered in the design of SDN-based VANET solutions.

Recently, Unmanned Aerial Vehicles (UAVs) are used as a new flexible solution to assist ground vehicular network in the zones with poor network connectivity or network partitioning situations [12–15]. Initially, designed for military utilization, the use of UAVs and especially mini-UAVs such as quadcopters, have been democratized. Nowadays UAVs are equipped with high definition imaging sensors, embedded processors and communication modules [15].

Henceforth, UAVs through their fluidity and flexibility, can facilitate many vehicular applications such as search rescue missions. Indeed, when a critical emergency problem occurs in a vehicular environment, such as traffic accidents, it often results in a big traffic jam, which may subsequently prevent emergency vehicles from accessing the affected zone. UAVs can assist ground emergency vehicles to explore and investigate the inaccessible affected zone. Also, UAVs can collect diverse data types about the affected area, such as aerial photography (pictures, videos, etc.). This information needs to be processed

as soon as possible to retrieve vital information about the number and status of victims, the number of vehicles involved, etc. These types of information are critical to properly evaluate the damages and assess the criticality of the situation in order to anticipate further reinforcements and prepare logistics if needed. UAVs and the emergency vehicle should collaborate to achieve efficient data processing policy while ensuring optimal balance between short computation delay and low UAVs' energy consumption.

In this paper, we propose in a first stage a novel distributed architecture to enable SDN in infrastructure-less UAV-assisted vehicular environments. We project this architecture into a road safety use-case, in which we propose to equip emergency vehicles with deployable small UAVs capable of investigating and exploring the inaccessible affected zone. We investigate in a second stage how to achieve efficient data processing policy for the collected data by UAV, which includes computation offloading (from UAV to its emergency vehicle) and sharing (between nearby ground vehicles) decision-making. The main purpose is to reach the best balance between UAVs' energy consumption and processing response delay. We formulate the offloading/sharing computation decision as a two-player sequential game. We study the existence of Nash equilibrium and design algorithms to solve the problem. Further, we demonstrate the efficiency of our solution compared to others native solutions via numerical results. To the best of our knowledge, this is the first work that incorporates SDN in UAV-assisted infrastructure-less vehicular environments and investigates data processing for road safety UAV-aided vehicular scenarios.

The main contributions of this paper are summarized as follows:

- We propose a novel distributed SDN-based architecture for UAV-assisted infrastructure-less vehicular networks,
- We investigate data processing in UAV-assisted VANET road emergency scenarios as an offloading/sharing computation decision-making problem to balance the computation delay and the consumption energy,
- We formulate the offloading/sharing computation problem as a two-player sequential game and design distributed algorithms to solve the problem,
- We evaluate the performances of the proposed game-based data processing policy for different system parameters. Numerical results demonstrate the effectiveness of the proposed scheme, by ensuring the best tradeoff between the computation delay and the consumed energy.

The reminder of this paper is organized as follows. We review the related works in Sect. 2. In Sect. 3, we present the novel distributed SDN-based UAV-assisted infrastructure-less VANET architecture, describe the UAV-aided road safety use-case and discuss the motivation of the data processing offloading/sharing computation decision problem. We bring in Sect. 4, the formulation of the problem and the related sequential game approach. Section 5 presents the numerical results. Finally, the conclusion is drawn in Sect. 6.

2 Related Work

In this section, we first discuss some related SDN-based VANET architectures before summarizing the relevant works on incorporating UAVs into ground vehicular network.

2.1 SDN-Based VANET Architectures

In the last few years, researchers investigated more how to take advantage of SDN benefits to improve the performances of current vehicular networks. However, most of existing SDN-based VANET works [1–6] propose fully centralized SDN-based architecture using only one centralized controller to handle the overall network. Nonetheless, this assumption seems clearly impracticable in such dynamic, dense and large networks as VANETs. Hence, this presents a serious risk of reliability and security and can generate a high end-to-end delay. There is also a high risk of controller bottleneck and message control overhead. Given these limitations and to better deal with scalability and reliability, authors in [7–11, 16, 17] propose to use multiple SDN controllers, and each controller handles a part of the network. Much better, some of the previous works [4, 7–11, 16] propose to situate the controllers in the edge of network to guarantee a reasonable end-to-end delay and satisfy the delay-sensitive application requirements. Authors in [18] and [19], propose intelligent data offloading strategy based on a fully centralized SDN control mode for heterogeneous VANET scenarios and 5G-enabled vehicular networks, respectively. Zhang et al. in [20] propose SOVCAN, an SDN-based method to elaborate a safety-oriented vehicular controller area for the purpose of ensuring driving safety by monitoring the driver's physiological and psychological state.

Nevertheless, almost all proposed SDN-based VANET architectures are infrastructure-based and host their controllers somewhere on the fixed infrastructure. Nowadays, the full coverage of fixed infrastructure is not yet reached in vehicular networks even with the integration of new emerging heterogeneous technologies as the cellular network. Uncovered VANET areas still exist, such for example, in some places when the coverage is not available and some areas where the fixed infrastructure is absent because the deployment is very difficult, costly or not profitable. Therefore, uncovered infrastructure-less vehicular areas must be considered in the design of SDN-based VANET architectures. In this direction, we have proposed in our previous work in [21] dSDiVN, a fully distributed SDN-based architecture for infrastructure-less vehicular scenarios with SDN controllers deployed on mobile vehicles.

In this paper, we attend to take advantages of the flexibility and the capabilities of UAVs to augment the coverage of the existing infrastructure-less vehicular network and benefit from the centralized global view of SDN to enhance its performances such as the end-to-end delay, data forwarding and processing.

2.2 UAV-Assisted VANET Works

Boosted by their high mobility, agility and fast deployment, UAVs are emerged as a revolutionary technology to assist the next generation wireless communication systems, especially 5G networks, for divers issues. For instance but not limited to, in [22–24] UAVs have deployed as an aerial base station for extending the network coverage and capacity of 5G enabled-wireless communications. More concretely, Bell Labs of NOKIA demonstrated in a recent test the world's first deployment of F-Cell, a solar-powered self-configured wireless small cell installed using a drone [25]. UAVs have also used in [26], as edge-cache to offload data traffic and improve the delivery latency in ultra-dense 5G networks.

Recently, incorporating UAVs to improve vehicular networks performance has attracted more attention from the researchers and different UAV-assisted VANET works are

proposed. In [12], authors propose VDNet, an infrastructure-less sparse UAV-assisted VANET system to improve data message transmission. Oubatti et al. in [13] and [27], propose UVAR, a routing protocol for UAV-assisted VANET. Authors in [14] use UAV as a relay to connect several isolated vehicle's segments. In [15], multi-UAV-aided VANET system is intended to improve the VANET performance in harsh environments. UAVs are used in [28] and [29] as store-carry-forward enabled airborne nodes to assist ground vehicles for reducing packet delivery delay and improving path connectivity in the presence of un-cooperative vehicles, respectively. In [30], Sliem et al. propose a routing protocol for sparse VANET area and analyze the minimum UAV number able to satisfy the vehicle-to-UAV packet delivery delay constraint. Sharma et al. propose in [31] an infrastructure-less multi-UAV coordinated VANETs as an efficient solution for tracking vehicles, analyzing the driver behavior and accurately detecting the faulty drivers for the purpose of improving the road traffic-safety management. In [32], an SDN-based architecture is proposed for supporting heterogeneous services in a vehicular network that integrate terrestrial vehicular infrastructure, aerial UAVs/Balloons and space satellite. Furthermore, authors in [33], investigate the problem of UAV docking station placement in a UAV-assisted Intelligent Transportation Systems (ITS) for enabling UAV to reach the incident location in a reasonable time and eliminate the risk of UAV's battery failure during the mission. These works have shown that the use of UAV to assist ground vehicles can enhance the packet delivery delay with a reasonable overhead. However, almost all proposed UAV-assisted VANET works deal with information dissemination and data forwarding problem. To the best of our knowledge, no UAV-assisted VANET work treats data processing problem. Moreover, in [34], a computation-offloading problem in a UAV network is investigated, where UAVs can offload intensive tasks to the nearby base station or an edge server in order to achieve a balance between energy and delay. Recently, some other works in [35–38] propose SDN-based UAV architectures to enhance the performance of user/control plane between UAVs, where data collection/forwarding UAVs are managed through a centralized remote ground controller.

Table 1 summarizes this section by comparing the related works cited above.

In what follow, we propose a distributed SDN-based infrastructure-less VANET architecture with multiple distributed controllers. We incorporate UAVs to assist

Table 1 Summary comparison between related works

Works	SDN-based	Controller number	UAV	Data processing	Infrastructure-less VANET
[1–6, 18–20]	Yes	Single	No	–	No
[7–11, 16, 17]	Yes	Multiple	No	–	No
[12–15, 31]	No	–	Yes	No	Yes
[13, 27–30, 33]	No	–	Yes	No	No
[32]	Yes	Multiple	Yes	No	No
[35–38]	Yes	Single	Yes	–	No
[22–24, 26]	No	–	Yes	–	No
[34]	No	–	Yes	Yes	No
[7, 21]	Yes	Multiple	No	No	Yes
Current work	Yes	Multiple	Yes	Yes	Yes

emergency vehicles for the exploring of inaccessible affected zones in road safety scenarios. We investigate how achieving efficient data processing of the data collected by UAVs. Likewise, several nearly similar works such as in [39] and [40], propose emergency vehicle prioritization mechanisms to enhance the delay efficiency of emergency services without considering efficient data processing of delay-sensitive safety tasks or using UAVs to assist ground emergency vehicles.

3 Background and Motivation

In this section, we first present a novel distributed SDN-based architecture for UAV-assisted infrastructure-less vehicular networks. Then, we introduce an UAV-aided road safety emergency scenario as a use-case of this architecture, and we bring the SDN architecture for the proposed system. Last, we discuss the motivation of data processing optimization.

3.1 SDN-Based Infrastructure-Less VANET Architecture

In this sub-section, we present a distributed SDN-based architecture for infrastructure-less vehicular environments, called *distributed Software-Defined infrastructure-less Vehicular Network (dSDiVN)*. dSDiVN is based on a logically centralized, but physically distributed multi-hop control plane. By using SDN with distributed multi-controller, dSDiVN benefits from scalability and reliability of the distributed architecture while preserving the simplicity of centralized management. dSDiVN is based on our previous work in [21]. Here, we just give an abstract of the SDN-based VANET architecture and for more details refer to [21].

As illustrated in Fig. 1, to enable SDN in infrastructure-less vehicular environments, dSDiVN proceeds by: (i) organizing and partitioning the network to make it more stable, smaller and less dynamic for a vehicle, and to reduce the control overhead and the latency. In dSDiVN, the uncovered road is divided into equal size virtual segments. Each segment represents a virtual SDN domain (simplified domain) that regroups all mobile vehicles within the same domain and with the same direction. The domain size is adjusted to half of IEEE 802.11p coverage to ensure that all vehicles of adjacent domain still always be reachable to each other, (ii) assigning a dedicated local controller to handle and control

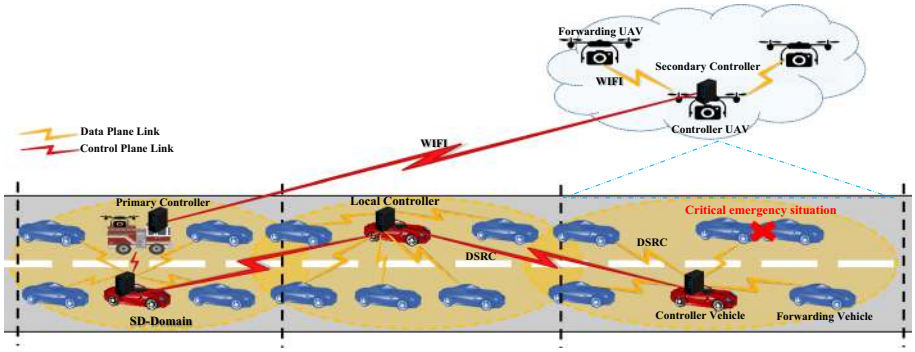

Fig. 1 UAV-assited dSDiVN system architecture for road safety use-case

each domain. Present on each vehicle, the local controller is initially in standby mode, and it is activated only when its hosting vehicle is selected as domain controller vehicle. The local controller handles and manages all the requests of forwarding vehicle members in its domain. Forwarding vehicles ensure only the monitoring of vehicle parameters (position, velocity, etc.) and the collection/forwarding of data information through a software local monitoring and collection agent, and (iii) connecting adjacent local controllers through V2V (vehicle to vehicle) links using IEEE 802.11p to build SDN-based control backhaul and enforce global policies. For the management and maintenance of the network partitioning, dSDiVN adapts the distributed clustering algorithm in [41].

3.2 SDN-Based UAV-Assisted VANET for Road Rescue Mission

Recently, UAVs have been proposed as a flexible solution to assist the vehicular network in infrastructure-less environments [12–15, 31]. By their fluidity and flexibility, UAVs can facilitate many vehicular applications such as search and rescue missions. Henceforth, when a critical emergency problem occurs in a vehicular environment, such as traffic accidents, it often results in a big traffic jam and sometimes in blocked road caused by crash remains, which may subsequently prevent the emergency vehicles from accessing the affected zone. We propose equipping the emergency vehicles with a deployable small UAVs capable of investigating and exploring the inaccessible affected area, see Fig. 1. UAVs can hover and collect different data information about the affected area through its onboard sensors, such as aerial photography (pictures, videos, etc.). This information needs to be processed as soon as possible to retrieve vital information about the number and state of victims, the number of vehicles involved, etc. These types of information are critical to properly evaluate the damages and assess the criticality of the situation in order to anticipate further reinforcements and prepare logistics if needed.

To efficiently handle the aerial/ground control plane communication between UAVs and emergency vehicle, we propose to take advantage of the centralized control of SDN as in [35–38]. This will facilitate the remote ground management of UAVs where the emergency vehicle will play the role of a centralized ground primary controller that deals with the commands installation of aerial missions, and UAVs are considered as data collection and forwarding devices. A powerful UAV will play the role of an aerial secondary controller for the swarm of forwarding UAVs, which ensure only the monitoring of UAV parameters (battery, position, etc.) and the collection and forwarding of data information, see Fig. 1. Using the aerial secondary controller allows to efficiently deal with aerial requests and can optimize the transmission energy of UAVs, reduce control overhead, and minimize response delay.

As illustrated in Fig. 2, the proposed architecture is based on the three layers SDN architecture:

1. *Data plane layer* consists of all components that only perform the collection and forwarding of data information. It includes *ground data plane sub-layer* composed of ground forwarding vehicles, and *aerial data plane sub-layer* composed of aerial forwarding UAVs.
2. *Control plane layer* consists of all components that centralize the domain/swarm intelligence and control forwarding nodes. It includes *ground control plane sub-layer* composed of the ground controller vehicles, and *aerial control plane sub-layer* composed of all aerial controller UAVs. We distinguish three kinds of SDN controllers:

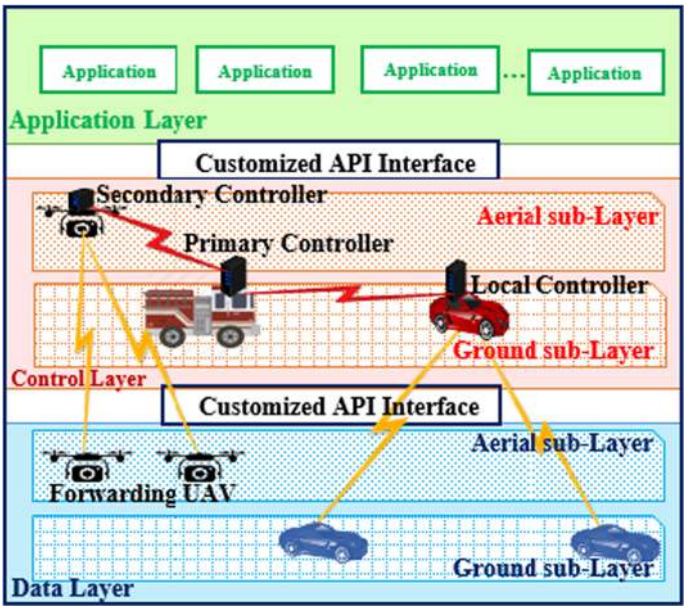

Fig. 2 3-tier SDN architecture of proposed system

(a) *Local controller* it is deployed on ground vehicles and communicates with others vehicles through V2V links using IEEE 802.11p. The local ground controller handles intra-domain vehicles requests and detains medium computation capabilities.

(b) *Primary controller* it is deployed on the ground emergency vehicle and communicates with controller vehicles through V2V links using IEEE 802.11p and with aerial controllers using WiFi. The primary controller handles the ground/aerial control requests and detains powerful computation capabilities.

(c) *Secondary controller* it is deployed on aerial UAV and communicates with ground controller vehicle and with forwarding UAVs using WiFi. The secondary controller handles aerial requests of its swarm forwarding UAVs and detains modest computation capabilities.

3. *Service and application layer* it contains all the network services and applications.

3.3 Motivation for Data Processing Optimization

In emergency road rescue missions, UAVs are used to assist ground emergency vehicles for exploring and investigating inaccessible affected road segments. UAVs can reach, hover and collect diverse data information about the affected zone such as aerial photography and videos. These data need to be processed to extract relevant information (*e.g.*, the number of crashed vehicles and its registration, the number of victims and identity, etc.) that can help rescue team to enhance the intervention quality. The processing of such kind of information is generally pattern recognition and video processing which is known to be intensive computation tasks that require complex calculations and powerful computing and energy resources. Furthermore, the size of the collected information can rapidly

increase as for high-quality long video sequences. The computation of such intensive tasks and/or the transmission of such big data by limited resources device as UAV can result in slow processing response time, long transmission delay and high energy consumption. However, in emergency road rescue scenarios, data processing response time is delay-sensitive and is of vital importance to the point of can save victims life in some situation. Also, UAV battery lifetime is determinant to the success of exploration rescue mission.

Henceforth, UAVs could collaborate with the ground emergency vehicle that has powerful computing resources and less energy constraint to achieve efficient data processing while ensuring optimal balance between computation delay and energy consumption. For example, in some scenario, it is more beneficial for an UAV in order to save its energy to compute big data tasks locally and send only the resulted small information to the emergency vehicle than sending and offloading all the data tasks to be processed on the emergency vehicle. Moreover, the emergency vehicle can optimize the computation response delay of high computation intensive tasks by sharing the computation tasks with it nearby controller vehicles.

In the next section, we investigate how achieving efficient data processing policy which includes computation offloading/sharing decision-making problem (which data offload / share and for each computation task) while considering the best tradeoff between computation delay and energy consumption.

4 Problem Formulation

In this section, we tackle the problem of achieving the best possible data processing computation in a critical emergency scenario. The model presented herewith is based on the previously presented UAV-assisted dSDiVN system. We start with presenting the system model. Then, we introduce the details related to the communication model and the computation model. Finally, we present the cost function that we use to evaluate our approach.

4.1 System Model

The system model presented in this section is based on UAV-assisted dSDiVN road safety use-case presented in the previous section. When a critical emergency situation occurs in a vehicular environment, such as traffic accidents, it often results in a big traffic jam, making the access to affected zone difficult for emergency vehicles. Facing such situation, emergency vehicles can send one or a swarm of its on board-UAVs (depending on the extent of the affected area) to explore and investigate the inaccessible affected zone. Being able of assuming some of the functionalities of the emergency vehicles (primary controller), as a secondary controller, a UAV can collect diverse types of data information about the affected zone. This information needs to be processed as soon as possible to aid rescue team to enhance the intervention quality.

In the modeling of our approach, we suppose that the emergency vehicle only needs to deploy a single UAV for exploring and investigating the affected region. This latter would always be reachable through a dedicated wireless interface. The problem can be easily extended to support multi-UAVs. Therefore, throughout this section, we only consider a simplified scenario for a UAV-assisted dSDiVN with an inaccessible affected zone as shown in Fig. 3. The emergency vehicle (EV) is equipped with one UAV and is connected

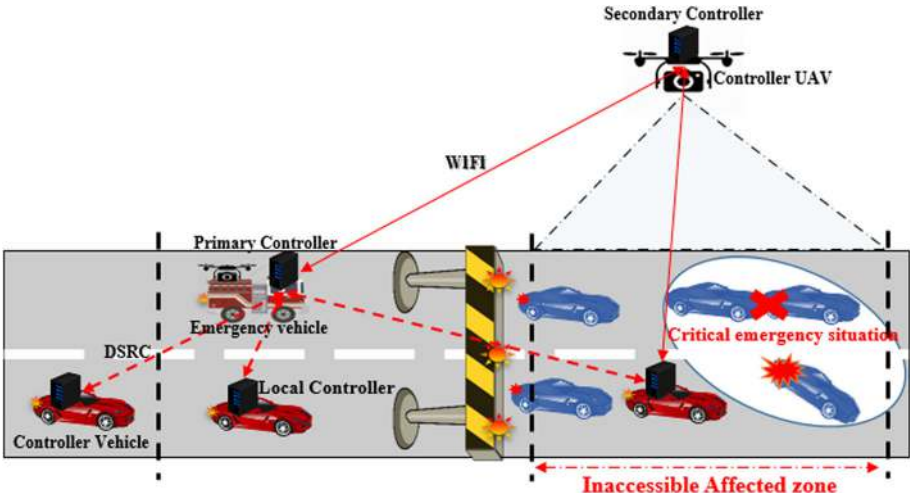

Fig. 3 UAV-assisted dSDiVN for road safety rescue scenario

to a set B of m nearby controller vehicles (CV), $B = \{1, 2, \ldots, m\}$. The UAV communicates with the emergency vehicle through WiFi. The emergency vehicle communicates with the controller vehicles through V2V link using DSRC (IEEE 802.11p).

This part of manuscript focuses on how achieving efficient data processing of the collected data by UAV. The main aim is to better balance the response computation time and consumption energy. The processing of this data and the related computation-intensive tasks can be, (i) performed locally on the UAV by the secondary controller, (ii) offloaded through WiFi in one-hop if the emergency vehicle is in the coverage range of UAV or in multi-hop using WiFi and after that using DSRC to be performed locally on the emergency vehicle by the primary controller, or (iii) shared between the ground primary controller and one of the local controller of it nearby controller vehicles using DSRC, as illustrated in Fig. 4.

We present in the following the communication and the computation models.

4.2 Communication Model

Since we consider a UAV-assisted infrastructure-less vehicular environment, all the communications are assumed to be device to device. We assume that the emergency vehicle is equipped with two communication interfaces: a WiFi (IEEE 802.11a) interface to communicate with the UAV and a DSRC (IEEE 802.11p) interface to communicate with the nearby controller vehicles.

Initially, the UAV collects data information about the inaccessible affected area. After that, the secondary controller when going to execute a task i, will take the offloading decision ($d_i^o = \{0, 1\}$) whether it chooses to process the data computation task locally ($d_i^o = 0$) or it chooses to offload the task using WiFi to be computed by its primary controller on the emergency vehicle ($d_i^o = 1$). If this is the case, the primary controller has to decide share it or not with nearby controllers after receiving the task. If it chooses to execute the computation task locally then $d_i^s = 0$, and if it chooses to share it with one of the nearby local controllers then $d_i^s = 1$.

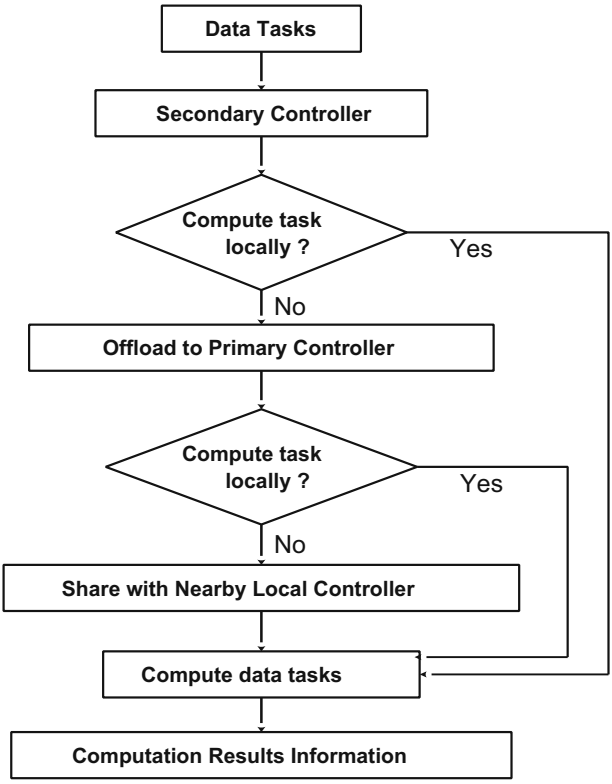

Fig. 4 Data processing policy flow chart

4.3 Computation Model

For the computation model, we assume a time-slotted system to describe the different network data processing, while at each time slot only one task is processed. $T = \{t_1, t_2, \ldots, t_{|T|}\}$ denotes the time slots under consideration. The length of each time slot is normalized to unity [3]. Furthermore and as in [42–44], since we study a traffic jam emergency situation and to enable tractable analysis, we consider a quasi-static scenario, where the number of ground vehicles and the position of UAV rest unchanged during one-time slot computation period. We also consider that each collected data to be processed can be split into several atomic intensive tasks. Each task i can be represented by a 3-tuple (C_i, Ds_i, Rs_i), $i \in N$, where, C_i represents the total computation CPU cycles required to accomplish the task i, Ds_i represents the total size of computation data (i.e., the computation input parameters and the program code to be executed) to offload / share, and Rs_i represents the size of task computation results (i.e., the computation output) to send back to the ground primary controller [43].

In our critical emergency situation with UAV-assisted vehicular for affected zone exploration scenario, the processing of collected data is a delay-sensitive task because of the emergency nature of the situation. Moreover, the UAV battery lifetime is determinant to the success of the exploration mission. Thus, the computation response delay and the consumed energy represent the principal metrics for the offloading/sharing decision-

making. The following presents in detail the different formulas of these metrics: delay and energy. Some similar formulas were proposed in [42–44].

We consider two level of decision-making with two possible computation decisions for each one: on secondary controller and on the primary controller. The details are given below.

4.3.1 UAV Computation Sub-Model

When UAV collects some data, the aerial secondary controller, based on the measured delay and energy metrics, chooses either to process the data locally or offload it either directly in one-hop to be processed by its ground primary controller using WiFi if the emergency vehicle is in its coverage zone or offload it in multi-hop, initially using WiFi to the nearest controller vehicle, after that point by point using DSRC until it arrives at the emergency vehicle.

1. *Local computation on Secondary Controller* the total processing delay $(D_{l,i}^{Sc})$ of local computation of task i, is the time of local computation (T_i^{ls}) plus the time of results sent back to primary controller (T_i^{Rs}), either in one-hop using WiFi or in multi-hop using WiFi and DSRC. The delay of local computation of task i is given as:

$$D_{l,i}^{Sc} = T_i^{ls} + T_i^{Rs} \tag{1}$$

where

$$T_i^{ls} = \frac{C_i}{F_{CPU}^{UAV}} \tag{2}$$

and

$$T_i^{Rs} = \frac{Rs_i}{R_{WiFi}^{UAV}} + \sum_{s=0}^{h} \frac{Rs_i}{R_{DSRC}} \tag{3}$$

where F_{CPU}^{UAV} denotes the computation frequency of the secondary controller in CPU cycles per second, R_{WiFi}^{UAV} denotes the data transmission rate of WiFi interface and R_{DSRC} denotes the data transmission rate of DSRC interface. h is the number of V2V hop necessary to reach the emergency vehicle.

The total energy consumption of local computation $(E_{l,i}^{Sc})$ is the energy of local computation (E_i^{ls}) plus the energy of results sending (E_i^{Rs}). We assume that vehicles are less energy-constrained, therefore, we neglect the vehicle's consumed energy for local computation and data transmission. The total energy consumption of local computation is given as:

$$E_{l,i}^{Sc} = E_i^{ls} + E_i^{Rs} \tag{4}$$

where

$$E_i^{ls} = C_i \times e_{CPU}^{UAV} \tag{5}$$

and

$$E_i^{Rs} = Rs_i \times e_{WiFi}^{UAV} \tag{6}$$

here, e_{CPU}^{UAV} denotes the local computation energy consumed by the central processing unit (CPU) of UAV for local computation and e_{WiFi}^{UAV} denotes the energy consumed by the WiFi interface of UAV to send one data unit to primary controller.

2. *Offload Computation on Primary Controller* the delay of offloading $(D_{o,i}^{Sc})$ is the time of data sent to the ground emergency vehicle either in one-hop using WiFi or in multi-hop using WiFi and DSRC. Offloading delay of task i is given as:

$$D_{o,i}^{Sc} = \frac{Ds_i}{R_{WiFi}^{UAV}} + \sum_{s=0}^{h} \frac{Ds_i}{R_{DSRC}} \tag{7}$$

and, the total energy of offloading $(E_{o,i}^{Sc})$ is represented by the energy consumed by UAV to offload (send) the data to emergency vehicle. The consumed energy is given as:

$$E_{o,i}^{Sc} = Ds_i \times e_{WiFi}^{UAV} \tag{8}$$

4.3.2 Emergency Vehicle Computation Sub-Model

When the emergency vehicle receives an offloaded data task from the UAV, its primary controller chooses either to compute all the tasks locally or share the computation tasks with one of the local controllers of it nearby controller vehicles. Before that, the primary controller acts in two steps: selects the best sharing nearby controller vehicle (local controller) and after that negotiate with it the sharing percentage. Based on this percentage, the primary controller can take the sharing decision. We model the potential delay (waiting time) needed to achieve this two steps by d_i:

1. *Selection of the best sharing nearby controller* to optimize the response computation time, primary controller can share the computing task with one of the local controllers of its nearby controller vehicles. We assume that all the nearby vehicles are homogeneous in term of computing capability and it is unsuitable to use a link that has reached its maximum capability. The primary controller selects the nearby sharing vehicle that has the best quality of link (QoL), which is expressed by the probability of a successful packet transmission in a small time interval [45]. Therefore, a link with faster transmission rate and lower delay latency will have high quality of link and the related controller vehicle obtains a large chance to be selected as best sharing computation vehicle [45]. The primary controller assigns a choice probability to each nearby vehicle based on its QoL. The quality of link that connects emergency vehicle e to nearby controller vehicle j in the time slot i, denoted by $\phi_{e,j}$, is modeled through the softmax action selection method by using a Boltzmann distribution as proposed in [45], here a link (e, j) is selected with probability $\omega_{e,k}$ as shown in Eq. (9) (see[45]).

$$\phi_{e,j} = \frac{\sigma_2}{\sigma_1 + \omega_{e,j}} \tag{9}$$

where

$$\omega_{e,j} = \frac{e^{\frac{q_{e,j}}{\xi}}}{\sum_{k=1}^{m} e^{\frac{q_{e,k}}{\xi}}} \tag{10}$$

Here, $\omega_{e,j}$ represents the probability that the nearby controller vehicle j can be chosen to share the computation task among the k ($k = [1, m]$) nearby controller vehicles, i.e., the probability that the controller vehicle j detains the best QoL. σ_1 and σ_2 denote the parameters that determine the influence from $\omega_{e,k}$. $q_{e,j}$ denotes the actual throughput of link e,j. ξ denotes the Boltzman temperature parameter. We model by ξ the actual transmission delay of link (e, j) . A high value of ξ results on an equal choice probability of all links and a small value results a high choice probability of links that have the best QoL. The best link corresponds to the link with the shortest delay and highest throughput. The link delay value is adjusted through the parameter ξ. A link with a short delay is represented by low ξ values and a link with high delay is represented by high ξ values. We assume that the primary controller manages and periodically updates a neighbor table that contains the actual delay and throughput of each link of it nearby controller vehicles.

2. *Negotiation of the sharing percentage* after the selection of best sharing vehicle that has the best quality of the link, the primary controller will negotiate with the local controller how much computation data it can accept. The primary controller starts by sending to the local controller the number of necessary computation CPU cycles of task i and this latter after evaluating it resources replies by sharing acceptance ratio. We represent this ratio by the variable γ, where $\gamma = [0, 1]$: 0 if the local controller does not accept to share any data computation tasks, 1 if it accepts to compute all the data computation tasks and between 0 and 1 when it accepts to compute only a part of the data computation tasks.
The different computation time formulas of local computation tasks and shared computation tasks of the primary controller are detailed as follow:

   (a) *Local computation on primary controller* the total delay of local computation ($D_{l,i}^{Pr}$) on primary controller is equal to the waiting time plus the processing time of local computation of task i (T_i^{lp}), is given as:

   $$D_{l,i}^{Pr} = d_i + T_i^{lp} \tag{11}$$

   and

   $$T_i^{lp} = \frac{C_i}{F_{CPU}^{EV}} \tag{12}$$

   where F_{CPU}^{EV} denotes the local computation frequency of the primary controller in CPU cycles per second. d_i is assumed very small compared to T_i^{lp}.

   (b) *Shared computation with local controller* the processing time of shared computation is equal to the time of local computation of partial data tasks on primary controller (T_i^{ls}) plus the time of partial data transfer to the chosen nearby controller vehicle T_i^t plus the time of local computation of partial data tasks on local controller (T_i^{lc}) plus the time to send results back (T_i^{Rs}) to the primary controller. The time of local computation of partial data tasks (T_i^{ls}) on the primary controller is given as:

$$T_i^{ls} = (1-\gamma) \times \frac{C_i}{F_{CPU}^{EV}} \tag{13}$$

The transfer time of partial shared data (T_i^t) is given as:

$$T_i^t = \gamma \times \frac{Ds_i}{R_{DSRC}} \tag{14}$$

The local computation time of shared data execution of local controller j (T_i^{lc}) is given as:

$$T_i^{lc} = \gamma \times \frac{C_i}{F_{CPU}^{CV_j}} \tag{15}$$

The time to send results back (T_i^{rs}) is given as:

$$T_i^{rs} = \gamma \times \frac{Rs_i}{R_{DSRC}} \tag{16}$$

From (13), (14), (15) and (16), we get:

$$T_{i,j}^{sh} = (1-\gamma) \times \frac{C_i}{F_{CPU}^{EV}} + \gamma \times \phi_{e,j} \times \left(\frac{Ds_i}{R_{DSRC}} + \frac{Rs_i}{R_{DSRC}} + \frac{C_i}{F_{CPU}^{CV_j}} \right) \tag{17}$$

The total time of shared computation ($D_{sh,i}^{Pr}$) is equal to the waiting time plus the processing time of shared computing is given as:

$$D_{sh,i}^{Pr} = d_i + T_{i,j}^{sh} \tag{18}$$

where $F_{CPU}^{cv_j}$ denotes the local computation frequency of the nearby local controller j in CPU cycles per second.

4.4 System Cost Function

Since we consider a critical emergency situation of a road safety scenario, the collected information needs to be processed urgently. Thus, the computation response time is a delay-sensitive computation task. Moreover, the UAV plays an important role in the exploration of affected zone and its battery lifetime is a determinant factor to the success of the rescue mission. Thus, the computation delay and the consumption energy represent the principal factors for the offloading/sharing decision-making. Consequently, the system payoff (cost) function is represented as a combination of the energy (the total consumed energy by UAV) and the delay (the total computational time) and is given as:

$$P_i = \alpha \times \frac{D_i - minD}{maxD - minD} + \beta \times \frac{E_i - minE}{maxE - minE} \tag{19}$$

where: α and β ($\alpha, \beta \in [0,1], \alpha + \beta = 1$), represent the weighting factors for computation time and consumption energy, respectively. *maxE* (*resp. minE*) represent the maximum (*resp.* minimum) energy and *maxD* (*resp. minD*) maximum (*resp.* minimum) delay necessary to compute the maximum (*resp.* minimum) size of the task's data, respectively.

In the next section, we present a sequential game approach for the previously presented data processing policy of the offloading/sharing decision-making problem.

5 Data Processing Offloading/Sharing Policy for Emergency Scenario in UAV-Assisted Infrastructure-Less VANETs

In this section, we descript a dynamic sequential game approach for data processing in a UAV-assisted VANET rescue scenario, which includes offloading/sharing decisions making. The main aim is to achieve the best balance between computation delay and consumption energy. We study the existence of the Nash equilibrium (NE) and design distributed offloading/sharing algorithms to solve the problem.

Game theory is considered as a powerful mathematical modeling tool to analyze the strategic decision-making problems of the interactions among multiple rational players that act toward achieving their own interests. Some recent works have used game theory as an enabling tool for their design of computation offloading approaches such as in [42–44]. Game theory has also been used to model different problems in SDN-based VANET scenarios such as resources sharing in [45] and control plane optimization in [3].

5.1 Offloading/Sharing Computation Sequential Game

We formulate the distributed data processing decision-making problem as a two-player perfect information, finite, non-zero-sum sequential game where all the players know the system parameters. In perfect information sequential game, the players act sequentially, and each player can observe the moves of the other players that move before it and make strategic choices accordingly. Because data are initially collected by the UAV, the secondary controller acts first, and the primary controller acts after observing the decision of the secondary controller and only moves if the secondary controller chooses to offload the data computation task. The number of data computation tasks is finite and equals to the number of computation tasks of collected data, and the game finishes after a finite number of time-slot when the equilibrium is achieved (all the parameters are optimized for each collected data), or the critical UAV's energy level is reached.

Definition 1 The sequential game is defined by the 3-tuple $G(N, A, \mu)$:

- $N = \{1, 2\}$, represents the non-empty finite set of players: 1 represents the aerial secondary controller on UAV and 2 represents the ground primary controller of the emergency vehicle.
- $A = \{a_1, a_2\}$, represents the non-empty finite set of players actions or strategies. Where the strategy of player 1 is $a_1 = \{s_j, \forall j \in (0\,\textit{Local computation}, 1\,\textit{Offload computation})\}$ and the strategy of player 2 is $a_2 = \{s_i, \forall i \in (0\,\textit{Local computation}, 1\,\textit{Shared computation})\}$.
- $\mu = (\mu_1, \mu_2)$, represents the utility function or the payoff function of each player. μ_1 is the payoff function of player 1 and μ_2 is the payoff function of player 2. μ_1 and μ_2 are given as:

$$\mu_{1,i}(a_{1,i}) = \theta_i \times \left(\alpha \times \frac{D_{l,i}^{Sc} - minD}{maxD - minD} + \beta \times \frac{E_{l,i}^{Sc} - minE}{maxE - minE} \right) + (1 - \theta_i) \times \left(\alpha \times \frac{D_{o,i}^{Sc} - minD}{maxD - minD} + \beta \times \frac{E_{o,i}^{Sc} - minE}{maxE - minE} \right) \tag{20}$$

where

$$\theta_i = \begin{cases} 1, & \text{if } s_{1,i} = 0 \\ 0, & \text{if } s_{1,i} = 1 \end{cases} \tag{21}$$

and

$$\mu_{2,i}(a_{2,i}) = \alpha \times \quad \vartheta_i \times \frac{D_{l,i}^{Pr} - minD}{maxD - minD} + (1 - \vartheta_i) \times \frac{D_{sh,i}^{Pr} - minD}{maxD - minD} \Bigg) \tag{22}$$

where

$$\vartheta_i = \begin{cases} 1, & \text{if } s_{2,i} = 0 \\ 0, & \text{if } s_{2,i} = 1 \end{cases} \tag{23}$$

The global system payoff (system cost) of a data computation task i at a time slot t is the sum of the payoff of each player, primary and secondary controller:

$$P_i(t) = \mu_i(t) = \mu_{1,i}(a_{1,i}) + \mu_{2,i}(a_{2,i}) \tag{24}$$

In this manuscript, we aim to minimize the total system cost by jointly optimizing the task offloading and sharing decisions.
Figure 5 illustrates the extensive representation of our proposed sequential offloading/sharing game and the correspondent strategic representation of two-player payoff matrix.

5.2 Nash Equilibrium

In this subsection, we investigate the existence of the Nash Equilibrium (NE) and solve the distributed two-player finite sequential offloading/sharing computation game by backward induction.

Definition 2 A pure-strategy Nash equilibrium of the two-player offloading/sharing computation game is a decision profile $d^* = (d_{1,1}^*, \ldots, d_{1,i}^*, d_{2,1}^*, \ldots, d_{2,i}^*)$ such that $\forall n \in N$ we have the following [43]:

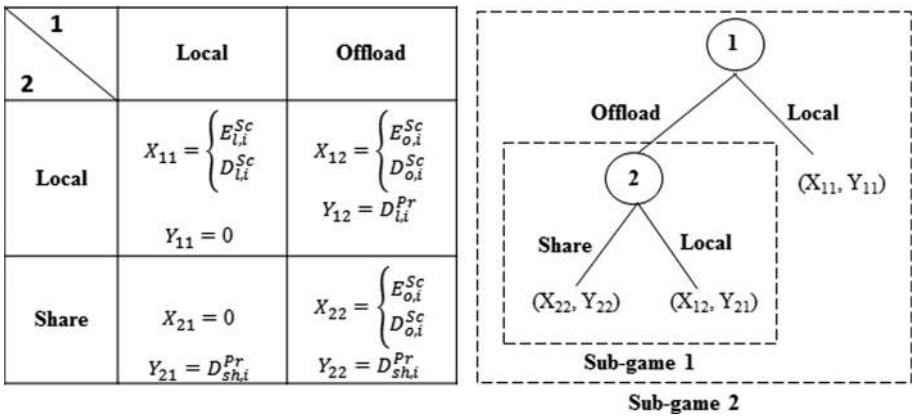

Fig. 5 Strategic/extensive representation of one stage of sequential offloading/sharing game

$$P_{n,i}(d^*_{n,i}) \preceq P_{n,i}(d_{n,i}), \forall d_{n,i} \in A \tag{25}$$

At Nash equilibrium, the primary and secondary controller can achieve a mutually satisfactory solution for that no controller has the incentive to unilaterally deviate [43]. This satisfactory solution is defined as an optimal strategy that should be selected by the two-player to get optimal energy and delay while treating each computation task.

Theorem 1 **(Kuhn)** "*Sequential games where the players have finite action sets and act only a finite number of times are equivalent, and every finite extensive-form game with perfect information has a pure-strategy Nash equilibrium*" [43].

Proof The proof of Theorem 1 is given in [46]. □

Lemma 1 $G(N, A, \mu)$ *is a finite sequential game with perfect information.*

Proof see the previous Sect. 5.1. □

Theorem 1 implies that for the distributed two-player sequential finite offloading/sharing computation game with perfect information (see Sect. 5.1), at least a Nash equilibrium exists and it can be converged in a finite number of decision slots [43].

Theorem 2 (Zermelo) "*Every finite game of perfect information has a pure strategy Nash equilibrium that can be derived by backward induction*" [47].

Proof The proof of Theorem 2 is given in [47]. □

As illustrated in Fig. 5, a stage of our distributed sequential finite two-player offloading/sharing computation game with perfect information in a decision time slot t contains two sub-games: (i) the local/sharing primary controller sub-game and (ii) the global game of local/offloading secondary controller sub-game. We solve the game by backward induction. First, we study the primary controller's local/sharing computation best strategy and then, we study the secondary controller's local /offloading computation best strategy.

5.3 Best Local/Sharing Computation Decision of Primary Controller

Given the secondary controller decision. Primary controller can derive the optimal sharing strategy S_i by solving the problem:

$$\phi_{2,i}(a_i) = \arg\min_{D^{Pr}} \mu_{2,i}(a_{2,i}) = \vartheta_i \times \frac{D^{Pr}_{l,i} - minD}{maxD - minD} + (1 - \vartheta_i) \times \frac{D^{Pr}_{sh,i} - minD}{maxD - minD} \tag{26}$$

where

$$\vartheta_i = \begin{cases} 1, & \text{if } s_{2,i} = 0 \\ 0, & \text{if } s_{2,i} = 1 \end{cases} \tag{27}$$

Here, D^{Pr} represents the computation delay response of the primary controller.

It is easy to check that the primary controller utility function (24) is convex.

$$\frac{\partial^2 \mu_{2,i}(D_{l,i}^{Pr}, D_{sh,i}^{Pr})}{\partial^2 D_{l,i}^{Pr}} = 0, \frac{\partial^2 \mu_{2,i}(D_{l,i}^{Pr}, D_{sh,i}^{Pr})}{\partial^2 D_{sh,i}^{Pr}} = 0 \tag{28}$$

From the previous equations, we conclude that the payoff function of the primary controller is a convex function. Hence, it has optimal solutions, and Nash equilibriums exist for the primary controller sub-game. However, since the number of players is small (two players), the payoff function is simple and easy to solve. Thus, the best decision in a single-slot strategy of the primary controller (payoff) is given by solving (24).

The optimal solution of $d_{2,i}^*$ for the single-slot strategy is given by:

$$d_{2,i}^* = \phi_{2,i}(a\_i) = \begin{cases} \arg\min_{D_{l,i}^{Pr}} \mu_{2,i}, & \text{if } s_{2,i} = 0 \\ \arg\min_{D_{sh,i}^{Pr}} \mu_{2,i}, & \text{if } s_{2,i} = 1 \end{cases} \tag{29}$$

Distributed Computation Sharing Algorithm in Algorithm 1, we design a distributed computation sharing policy algorithm to compute the best decision of primary controller. At each decision time slot, the primary controller treats only one computation task and acts in three steps (see Sect. 4.3.2) when it receives a computation data. First, it selects the best sharing controller vehicle based on it QoL. In a second step, it negotiates the sharing percentage with the local controller of selected controller vehicle, and in the third step, it computes the local and sharing computation delay and makes the best response decision based on the previously measured parameters. Since the problem is formulated as a perfect information sequential game, the primary controller informs the secondary controller of its decision. The Nash equilibrium strategies are reached when the delay and energy are optimized for each task i and no player has incentive to deviate this equilibrium. When the equilibrium is achieved or the game finish, the primary controller sends an end message to the secondary controller. The proposed algorithm is inspired from [42, 43].

Algorithm 1 Distributed Computation Sharing Algorithm

Input: A set of offloaded intensive-computation tasks
Output: Optimal sharing decision profile

Initialization
Primary controller selects its first strategy using the local computation: $d_{2,i}(0) \longleftarrow 0$
Initialize and update its neighboring table
End initialization

Begin
repeat for each decision slot t:
if **receive** a data computation task i **then**
Select the best sharing local controller
Negotiate the sharing percentage
Measure the sharing parameters
Select the new primary controller best response $\phi_{2,i}(a_{-i})(t+1)$ using (25)
Compute the new value for P_{i+1} using (22)
if $(\phi_{2,i}(a_{-i})(t+1) < \phi_{2,i}(a_{-i})(t))$ **then**
Make the decision $d_{2,i}(t+1) \leftarrow 1 - d_{2,i}(t)$
if **receive** energy alert message **then**
break
else **send** the decision information to secondary controller
until an equilibrium is achieved
Send end message to secondary controller
End

5.4 Best Local/Offload Computation Decision of the Secondary Controller

The secondary controller can derive the optimal offloading strategy by solving the following problem:

$$\begin{aligned}\phi_{1,i}(a_i) = \underset{D_{Sc},E_{Sc}}{\arg\min}\mu_{1,i}(a_{1,i}) = \theta_i \times \quad & \alpha \times \frac{D_{l,i}^{Sc} - minD}{maxD - minD} + \beta \times \frac{E_{l,i}^{Sc} - minE}{maxE - minE}\Bigg) \\ + (1-\theta_i) \times \quad & \alpha \times \frac{D_{o,i}^{Sc} - minD}{maxD - minD} + \beta \times \frac{E_{o,i}^{Sc} - minE}{maxE - minE}\Bigg)\end{aligned} \tag{30}$$

where

$$\theta_i = \begin{cases} 1, & \text{if } s_{1,i} = 0 \\ 0, & \text{if } s_{1,i} = 1 \end{cases} \tag{31}$$

Here, D^{Sc} and E^{Sc} represent the computation response delay and energy of the secondary controller, respectively.

It is easy to check that the secondary controller utility function (26) is convex.

$$\frac{\partial^2 \mu_{1,i}(D_{l,i}^{Sc}, D_{o,i}^{Sc}, E_{l,i}^{Sc}, E_{o,i}^{Sc})}{\partial^2 D_{l,i}^{Sc}} = 0, \frac{\partial^2 \mu_{1,i}(D_{l,i}^{Sc}, D_{o,i}^{Sc}, E_{l,i}^{Sc}, E_{o,i}^{Sc})}{\partial^2 D_{o,i}^{Sc}} = 0,$$
$$\frac{\partial^2 \mu_{1,i}(D_{l,i}^{Sc}, D_{o,i}^{Sc}, E_{l,i}^{Sc}, E_{o,i}^{Sc})}{\partial^2 E_{l,i}^{Sc}} = 0, \frac{\partial^2 \mu_{1,i}(D_{l,i}^{Sc}, D_{o,i}^{Sc}, E_{l,i}^{Sc}, E_{o,i}^{Sc})}{\partial^2 E_{o,i}^{Sc}} = 0 \tag{32}$$

From the previous equations, we conclude that the payoff function of the secondary controller is a convex function. Hence, it has optimal solutions, and Nash equilibriums exist for the secondary controller sub-game. However, when the number of players is small (two players), the payoff function is simple and easy to solve. Thus, the best decision in a single-slot strategy of the primary controller (payoff) is given by solving (26).

The optimal solution of $d_{1,i}^*$ for the strategy is:

$$d_{1,i}^* = \phi_{1,i}(a\_i) = \begin{cases} \arg\min_{D_{l,i}^{Sc}, E_{l,i}^{Sc}} \mu_{1,i}, & \text{if } s_{1,i} = 0 \\ \arg\min_{D_{o,i}^{Sc}, E_{o,i}^{Sc}} \mu_{1,i}, & \text{if } s_{1,i} = 1 \end{cases} \tag{33}$$

Distributed computation offloading algorithm in Algorithm 2, we design a distributed computation offloading policy algorithm to compute the best decision of secondary controller. The logic of Algorithm 2 is similar to Algorithm 1. The secondary controller monitors the energy of its UAV continuously. If it observes that the critical energy level (the energy needed by the UAV to return to the emergency vehicle) is reached, the secondary controller send a low energy alert message to the primary controller. The algorithm finishes when the end message is received (the equilibrium is reached). Since the secondary controller acts first, Algorithm 2 is executed before Algorithm 1.

Algorithm 2 Distributed Computation Offloading Algorithm

Input: A set of intensive-computation tasks
Output: Optimal offloading decision profile

Initialization
Secondary controller selects its first strategy using the local computation: $d_{1,i}(0) \longleftarrow 0$
Compute initial value of cost function P_i
End initialization

Begin
repeat for each data computation task i and each decision slot t:
Measure the offloading parameters
Select the new best strategy of secondary controller $\phi_{1,i}(a_{-i})(t+1)$ using (27)
Compute the new value for P_{i+1} using (22)
if $\phi_{1,i}(a_{-i})(t+1) \prec \phi_{1,i}(a_{-i})(t)$ **then**
Make the decision $d_{1,i}(t+1) \longleftarrow 1 - d_{1,i}(t)$
Send results/data to primary controller
if return energy level is reached **then**
Send alert message to primary controller
until receive end message from primary controller
End

6 Numerical Results

In this section, we use Matlab to evaluate the performance of the proposed distributed offload/sharing computation algorithms. The simulated scenario is based on the system model described in Sect. 4.1. For the different experimentations, we consider that the CPU computation capability of the emergency vehicle (primary controller) F_{CPU}^{EV} is ten times more powerful than that of UAV (secondary controller) F_{CPU}^{UAV} and the CPU computation capability of nearby controller vehicle (local controller) F_{CPU}^{CV} is three times more powerful of that of UAV (secondary controller). We also consider than the sending energy of one data unit through the wireless WiFi interface of UAV e_{WiFi}^{UAV} consumes one thousand times more than the local computation energy cycle e_{CPU}^{UAV} of the same data unit on the UAV [48]. The transmission rate of WiFi interface ($R^{WiFi} = 12Mbps$) is considered two times the transmission rate of DSRC interface ($R^{DSRC} = 6Mbps$) [49]. Since we investigate a critical emergency scenario where the result of data processing is delay-sensitive and we care about the computation time, we set a high weight ($\alpha = 0.7$) to computation delay in the decision making compared to the weight of computation energy consumption ($\beta = 0.3$).

The main simulation parameters are summarized in Table 2.

In the rest of this section, we compare the performances of the proposed distributed offload/sharing computation (DOSC) algorithm based on the two-player data processing sequential game with three other scenarios: (i) local computation (LC), where the secondary controller on UAV decides to compute all the tasks locally, (ii) offload computation (OC), where the secondary controller of UAV decides to offload all the tasks to the emergency vehicle and the primary controller decides to compute all the tasks locally and, (iii) shared computation (SC), where the secondary controller of UAV decides to offload all the tasks to the emergency vehicle and the primary controller decides to share a percentage of computation tasks with one of its nearby local controllers.

Table 2 Simulation parameters

Parameters	Value
C_i	$[100, \ldots, 100000](\times 10^3)$
Ds_i	$[10, \ldots, 2000](\times 10^3)$
Rs_i	$[5, \ldots, 15]$
F_{CPU}^{UAV}	1 GHZ
F_{CPU}^{EV}	10 GHZ
F_{CPU}^{CV}	3 GHZ
R^{WiFi}	12 Mbps
R^{DSRC}	6 Mbps
e_{CPU}^{UAV}	1 u
e_{WiFi}^{UAV}	100 u
M	[1, 4]
h	[0, 4]
α	0.7
β	0.3
σ_1	0.5
σ_2	0.5
γ	0.5

In the evaluation illustrated in Fig. 6, we study the average system cost for different data processing scenarios. As we can clearly remark from the results in Fig. 6, our DOSC algorithm is effective and outperforms all the other three scenarios whilst achieving 28, 25 and 17% gain in terms of the average system cost compared to SC, LC and OC scheme, respectively. This is principally because at each decision time slot, the proposed DOSC policy smartly balances between different system metrics (computation response time and UAV's energy) according to the actual available network resources (*e.g.*, CPU, UAV energy, QoL) in order to make better well-informed decisions that allow the emergency vehicle to get the fastest as possible, relevant safety information result of the intensive-tasks computation while better saving the UAV energy. This up-to-date view on network resource state is guaranteed via the collaboration among the SDN controllers. Contrarily, other computation decision-making scenarios make naive decisions without any consideration of the network resource state.

In Fig. 7, we investigate the impact of the size of collected data by UAV on the system cost. In this evaluation, we fix the data computation complexity in terms of CPU cycles and we change each time the size of data tasks. From Fig. 7, we can see that the system cost for OC/SC increases as the data size increases while this latter has a very slight impact on LC. Moreover, SC is better than OC for relatively small data size but not for big data size. This is because offloading /sharing big data via wireless interface resulting in a degradation of system performance caused by a long transmission delay and high UAV's energy consumption, contrarily to LC where only small size computation results are sent. As result, we conclude that LC is more suitable for big data size where OC and SC are more appropriate for medium and small data sizes respectively. We can also clearly remark the effectiveness of our DOSC policy that already chooses the best possible strategy corresponding to the optimal system cost that allow the system to perform intensive-computation tasks within acceptable response time and reasonable UAV energy consumption whatever the task's data size, thanks to the decision profile achieved by the intelligent SDN-based DOSC algorithm in NE.

In the evaluation illustrated in Fig. 8, we study the impact of the computation complexity of the collected data by UAV in terms of CPU cycles on the system cost. In this experimentation, we fix the data size of the computation tasks and we change at each time the number of CPU cycles. As Fig. 8 shows, the system cost of LC/OC increases with the increase of the number of CPU cycles while that of SC experiments a slight impact. This is

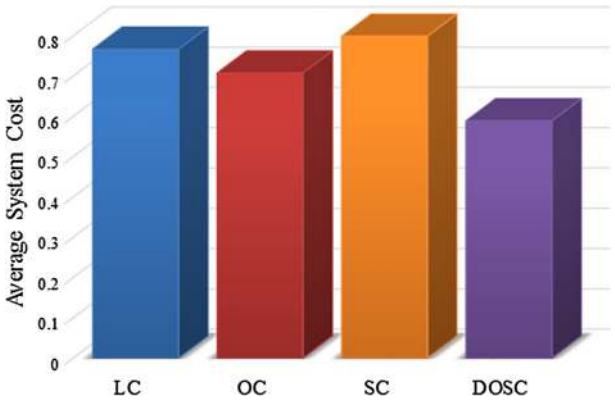

Fig. 6 Average system cost for different data processing scenario

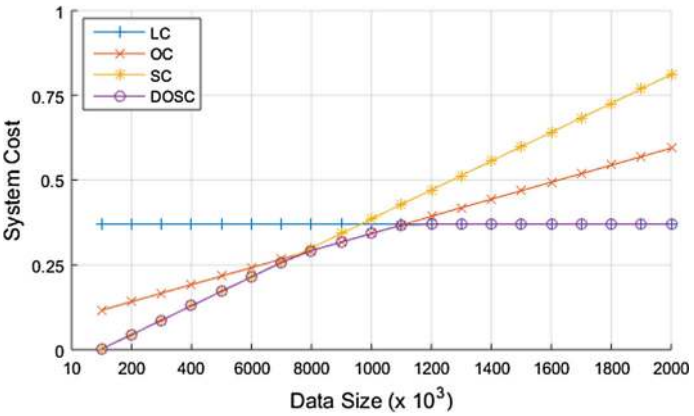

Fig. 7 The Impact of data size on system cost

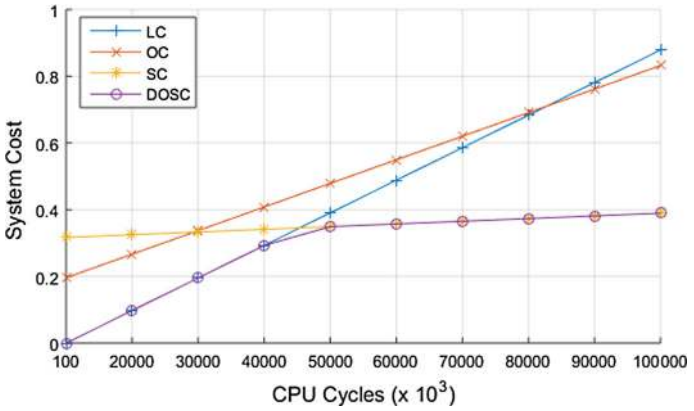

Fig. 8 The impact of CPU cycles on system cost

principally due to the fact that local computation by secondary controller or offload computation by primary controller of complex tasks with high CPU cycles requires more energy and/or delay resources depending on CPU rate which increases the system cost. This is in contrast to SC where several distributed SDN controllers of nearby vehicles collaborate by sharing the tasks computation. By consequence, we can conclude that the SC is more suitable for high intensive tasks where LC is more appropriate to less intensive tasks. Moreover, we remark that our proposed DOSC policy by balancing the system metrics under different decision computation strategy using the SDN-based optimization algorithms already adopts the most efficient strategy corresponding to the optimal system cost that ensures the best possible tradeoff between fast computation response time and less energy consumption.

In Fig. 9, we investigate the impact of the size of data computation tasks on the UAV's average energy consumption. We change the data size and we compute the average energy consumption for different CPU cycles for each size of data computation task. The results illustrated in Fig. 9, show that the OC and SC energy consumption increases linearly with the increase of the size of computation data task inversely to LC that seems not much be

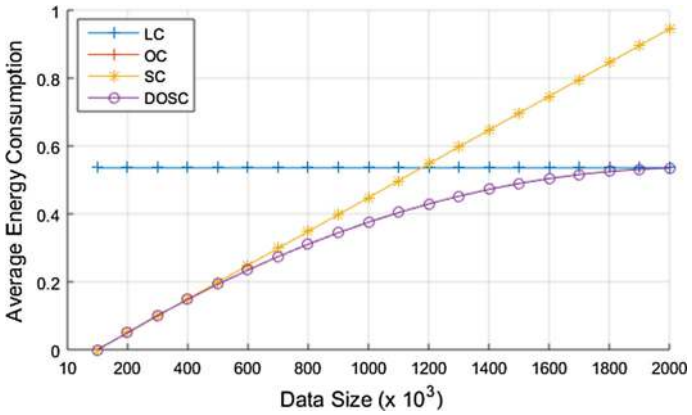

Fig. 9 The impact of data size on average energy consumption

affected by the increase in data size. Also, because we consider that ground vehicles have no energy constraint, OC and SC energy consumption are identical and equal to the energy of data transmission by UAV. This can be justified by the fact that the wireless transmission of a data size unit is largely more energy-gourmand than if locally computed [48]. Indeed, in OC/SC, the tasks computation goes through the transmission of overall big data size of tasks through the wireless interface. This results in high consumption of the limited energy resource of UAV before be computed on emergency or/and controller vehicles, refer to Eq. (8). Whereas LC consumes only small energy of data computation by secondary controller on UAV plus the transmission energy of small size result, refer to equation in (4). As a conclusion, we can say that OC is more suitable for tasks with small data size while LC is more appropriate for tasks with big data size. Moreover, it's clear that our DOSC policy based on the achieved decision profile in NE and the collaboration between the SDN controllers efficiently saves the UAV's energy by already selecting the most efficient optimal strategy that minimizes the energy consumption.

In the evaluation illustrated in Fig. 10, we study the impact of the data size of computation tasks on the average computation delay. In this evaluation, we change the data

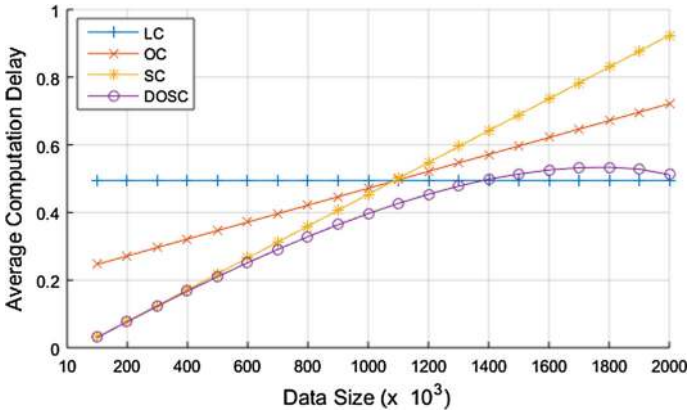

Fig. 10 The impact of data size on average computation delay

size and we compute the average computation delay for different CPU cycles for each size of data computation task. The results in Fig. 10 show that the computation delay of OC and SC increases linearly with the increase of the size of the computation data task while that of LC experiments a very slight effect and by consequence, it clearly seems that for small data sizes OC/SC is more suitable while for big data LC is more appropriate. The main justification of these results is principally related to computation logic of OC and SC that consists in transmitting through UAV's wireless interface of all the task's big data to be processed on ground vehicles resulting in supplementary high transmission time in addition to the computation time, refer to Eqs. (7), (11) and (18), respectively. Whereas in LC the collected data tasks are initially processed in relatively smaller time by secondary controller on UAV before transmitting only relevant small size results, refer to Eq. (1). Furthermore, it is clear that our proposed DOSC policy achieves the optimal computation delay by balancing and selecting the best computation strategy that minimizes the task's computation response time.

The results illustrated in Figs. 9 and 10, demonstrate that our proposed DOSC achieves a flexible tradeoff between the energy consumption and the computation delay.

In Fig. 11, we investigate the impact of sharing decision on average computation delay of primary controller in function of task's data size as illustrated in sub-figure (a) and task's complexity in terms of CPU cycles as illustrated in sub-figure (b). From sub-figures (a) and (b), we remark that average computation delay increases when sharing computation of tasks with big data and also when locally compute high intensive tasks. As result, we can conclude that the shared computation is more suitable for small data size and high intensive computation tasks thanks to the benefits of distributed collaborative computation among nearby ground controllers. In the contrary, the local computation at the primary controller is more appropriate for less intensive computation tasks and big data size tasks fault of the costly wireless transmission of tasks with voluminous data.

In the evaluation illustrated in Fig. 12, we evaluate the system cost for different system metrics (delay and energy) weight factors settings (α and β ($\beta = 1 - \alpha$)). In this experimentation, we fix the task's data size and CPU cycles and we vary the system metrics weight factors (α and β) settings. From the result in Fig. 12, we remark that the variation of the system metric weight factors has a significant impact on the system cost value, also, setting a higher value of one weight factor compared to the other, allow to enhance and decrease the system cost. More particularly, by setting delay weight factor (α) higher than the energy weight factor (β), our DOSC policy achieves until 50% improvement of the system cost compared to equal value metric parameters. Results that confirm our initial choice of the weight factors value, which can be justified by the fact that we attribute a

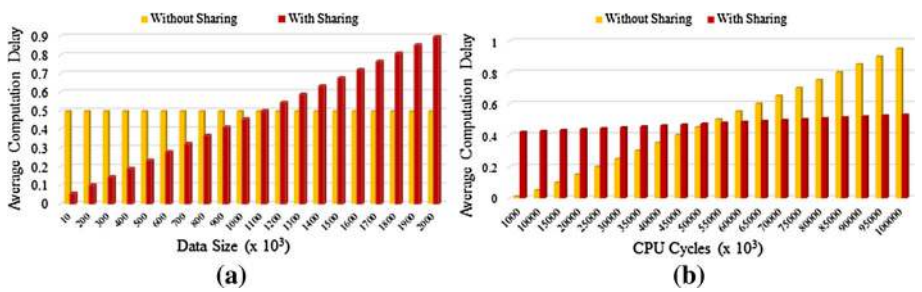

Fig. 11 The effect of sharing decision on primary controller (PC): average computation delay

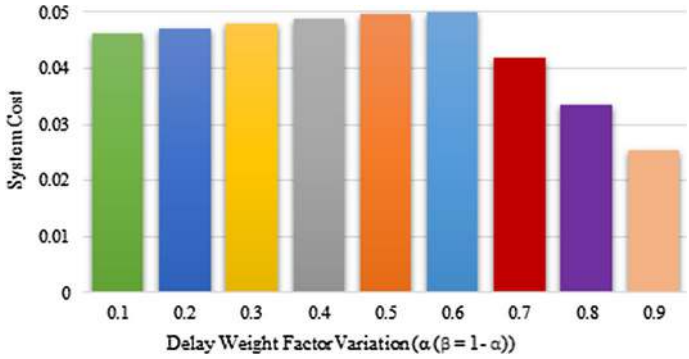

Fig. 12 The impact of system metrics weight factors variation on system cost

more important weight to the computation response time by optimizing the delay at each system computation point and considering only the UAV energy, given the critical delay-sensitive nature of the under-study road safety rescue use-case. Likewise, our proposed SDN-based game model, use these system weight factor parameters as a flexible way to balance the system performance metrics according to the actual context and to efficiently support the requirements of various application use-cases.

In the experimentation illustrated in Fig. 13, we simulate a simple multi-hop data transmission scenario from an aerial source to a ground destination far in 4 hop vehicles. We compare the performance of our SDN-based DOSC algorithm in terms of average delay, i.e., the average response time until the destination get the data information, with two UAV-assisted VANET data delivery protocols as reference works with a distributed path selection and without SDN logic, for different data size: UVAR-S in [27] and VDNet in [12]. The results illustrated in Fig. 13 show that the proposed SDN-based DOSC algorithm is efficient and outperforms the related works whilst achieving from 35 to 95% better average delay. This can be justified by the fact that our DOSC algorithm using the achieved decision profile in NE is already able to smartly choose the most efficient scenario with the smallest response time. Whereas, other related works focus mainly on the path selection and availability and don't care about what is better to reduce the delay: local

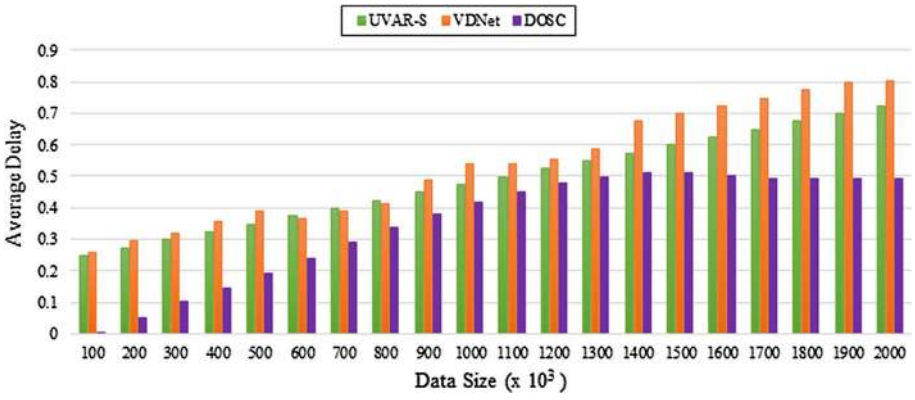

Fig. 13 Performance comparison of our SDN-based distributed offloading/sharing scheme with related data delivery UAV-assisted works in terms of average delay

computation of tasks with big data by UAV and only sending small computation results or the offload computation on the ground vehicle after the transmission of all task's data whatever it size. Moreover, both related works and specially VDNet, present a supplementary delay in addition to the delivery delay caused by the distributed periodic phases of destination location prediction and end-to-end path selection. Whereas, in our solution, all this information are beforehand available at each virtual SDN domain, thanks to the well-informed collaborative up-to-date network knowledge of the SDN controllers. Furthermore, UVAR-S by equipping both the UAVs and ground vehicles with low latency DSRC interface, benefits from relatively small advantage compared to VDNet. Finally, it is obvious that the delay increase with the increase of data size because the wireless transmission of big data size results in high delays.

7 Conclusion

In this paper, we first present a novel distributed SDN-based architecture for UAV-assisted infrastructure-less vehicular environments. We focus mainly on a road safety scenario as a use-case of the proposed architecture. In such scenario, UAVs are incorporated in order to assist emergency rescue vehicles to explore and investigate inaccessible affected zones. We investigate how to achieve an efficient data processing policy of the data collected by UAV which includes computation offloading and sharing decision-making problem. The main propose is to find the best balance between computation delay and consumption energy. We formulate the offloading/sharing decision problem via a theoretical game approach as a two-player sequential game and then we study the existence of Nash equilibrium. After that, we design distributed computation algorithms to solve the problem. Numerical results show that the proposed SDN-based data processing policy achieves efficient computation performance in terms of both computation delay and energy consumption whilst achieving until 28% gain of average system cost and between 35% and 95% better response time compared to other related native data processing scenarios and data delivery UAV-assisted VANET works without SDN, respectively.

The deployment complexity and cost rests open challenge in front of integrating SDN in infrastructure-less VANET areas. As future work, we plan to extend the theoretical game model to a general case with N players, multi UAVs-assisted VANET by using offloading to adjacent UAVs and sharing with multiple nearby controllers.

References

1. Duan, X., & Wang, Y. (2017). SDN enabled 5G-VANET: Adaptive vehicle clustering and beamformed transmission for aggregated traffic. *IEEE Communications Magazine*, *55*(7), 120–127.
2. Secinti, G., Canberk, B., Duong, T. Q., & Shu, L. (2017). Software defined architecture for VANET: A testbed implementation with wireless access management. *IEEE Communications Magazine*, *55*(7), 135–141.
3. Li, H., Dong, M., & Ota, K. (2016). Control plane optimization in Software-Defined Vehicular Ad hoc Networkss. *IEEE Transactions on Vehicular Technology*, *65*(10), 7895–7904.
4. Wang, X., Wang, C., Zhang, J., Zhou, M., & Jiang, C. (2016). Improved rule installation for real-time query service in Software-Defined internet of vehicles. *IEEE Transactions on Intelligent Transportation Systems*, *99*, 1–11.
5. Venkatramana, D. K. N., Srikantaiah, S, B., & Moodabidri, J. (2017). SCGRP: SDN-enabled connectivity-aware geographical routing protocol of VANETs for urban environment. *IET Networks*, *6*(5), 102–111.

6. He, Z., Cao, J., & Liu, X. (2016). SDVN: Enabling rapid network innovation for heterogeneous vehicular communication. *IEEE Network*, *30*(4), 10–15.
7. Alioua, A., Senouci, S. M., Moussaoui, S., Sedjelmaci, H., & Boualouache, A., (2017). Software-Defined heterogeneous vehicular networks: Taxonomy and architecture. In *The proceeding of the 2017 Global Information Infrastructure and Networking Symposium (GIIS)* (pp. 50–55). St. Pierre, France.
8. Kazmi, A., Khan, M. A., & Akram, M. U. (2016). DeVANET: Decentralized Software-Defined VANET architecture. In *The proceeding of the IEEE international conference on cloud engineering workshop (IC2EW)*.
9. Zheng, Q., Zheng, K., Zhang, H., & Leung, V. C. M. (2016). Delay-optimal virtualized radio resource scheduling in Software-Defined vehicular networks via stochastic learning. *IEEE Transactions on Vehicular Technology*, *65*(10), 7857–7867.
10. Correia, S., Boukerche, A., & Meneguette, R, I. (2017). An architecture for hierarchical Software-Defined vehicular networks. *IEEE Communications Magazine*, *55*(7), 80–86.
11. Chen, J., Zhou, H., Zhang, N., Xu, W., Yu, Q., Gui, L., et al. (2017). Service-oriented dynamic connection management for Software-Defined internet of vehicles. *IEEE Transactions on Intelligent Transportation Systems*, *18*(10), 2826–2837.
12. Wang, X., Fu, L., Zhang, Y., Gan, X., & Wang, X. (2016). VDNet: An infrastructure-less UAV-assisted sparse VANET system with vehicle location prediction. *Wireless Communications and Mobile Computing*, *16*, 2991–3003.
13. Oubbati, O. S., Lakas, A., Lagraa, N., & Yagoubi, M. B. (2016). VConnectivity of VANET segments using UAVs. In *The proceeding of the internet of things, smart spaces, and next generation networks and systems*.
14. Shilin, P., Kirichek, R., Paramonov, A., & Koucheryavy, A. (2016). UVAR: An intersection UAV-assisted VANET routing protocol. In *Proceeding of the. (2016) IEEE wireless communications and networking conference*. Qatar: Doha.
15. Zhou, Y., Cheng, N., Lu, N., & Shen, X, S. (2015). Multi-UAV-aided networks: Aerial-ground cooperative vehicular networking architecture. *IEEE Vehicular Technology Magazine*, *10*(4), 36–44.
16. Zheng, k, Hou, L., Meng, H., Zheng Lu, N., & Lei, L. (2015). Soft-defined heterogeneous vehicular network: Architecture and challenges. *IEEE Network*, *30*(4), 72–80.
17. Ghafoor, H., & Koo, I. (2018). CR-SDVN: A cognitive routing protocol for Software-Defined vehicular networks. *IEEE Sensors Journal*, *18*(4), 1761–1772.
18. Huang, C. M., Chiang, M, S., Dao, D, T., Pai, H, M., Xu, S., & Zhou, H. (2017). Vehicle-to-infrastructure (V2I) offloading from cellular network to 802.11p Wi-Fi network based on the Software-Defined Network (SDN) architecture. *Vehicular Communications*, *9*, 288–300.
19. Aujla, G. S., Chaudhary, R., Kumar, N., Rodrigues, J, J, P, C., & Vinel, A. (2017). Data offloading in 5G-enabled Software-Defined vehicular networks: A stackelberg-game-based approach. *IEEE Communications Magazine*, *55*(8), 100–108.
20. Zhang, Y., Chen, M., Kumar, N., Guizani, N., Wu, D., & Leung, V. C. M. (2017). SOVCAN: Safety-oriented vehicular controller area network. *IEEE Communications Magazine*, *55*(8), 94–99.
21. Alioua, A., Senouci, S., M., & Moussaoui, S. (2017). dSDiVN: A distributed Software-Defined Networking architecture for infrastructure-less vehicular networks. *The proceeding of I4CS* (pp. 56–67). Darmstadt, Germany.
22. Sharma, V., Srinivasan, K., Chao, H, C., Hua, K, L., & Cheng, W, H. (2017). Intelligent deployment of UAVs in 5G heterogeneous communication environment for improved coverage. *Journal of Network and Computer Applications*, *85*, 94–105.
23. Kalantari, E., Shakir, M. Z., Yanikomeroglu, H., & Yongacoglu, A. (2017). Backhaul-aware robust 3D drone placement in 5G+ wireless networks. *The proceeding of 2017 IEEE international conference on communications workshops (ICC Workshops)* (pp. 109–114). Paris, France.
24. Iellamo, S., Lehtomaki, J. J., & Khan, Z. (2017). Placement of 5G Drone Base Stations by Data Field Clustering. *The proceeding of IEEE 85th vehicular technology conference (VTC Spring)* (pp. 1–5). Sydney, NSW.
25. Bell labs future cell project by using massive MIMO for efficient small cell deployment. https://www.nokia.com/en_int/news/releases/2016/10/03/f-cell-technology-from-nokia-bell-labs-revolutionizes-small-cell-deployment-by-cutting-wires-costs-and-time.
26. Zhao, N., Cheng, F., Richard Yu., F., Tang, J., Chen, Y., Gui, G., & Sari, H. (2018). Caching UAV assisted secure transmission in hyper-dense networks based on interference alignment. *IEEE Transactions on Communications,* Early access. https://ieeexplore.ieee.org/document/8254370/.
27. Oubbati, O., Lakas, L., Zhou, F., Gunes, M., Lagraa, L., & Yagoubi, M. B. (2017). Intelligent UAV-assisted routing protocol for urban VANETs. *Computer Communications*, *107*, 93–111.

28. Fawaz, W., Atallah, R., Assi, C., & Khabbaz, M. (2017). Unmanned aerial vehicles as store-carry-forward nodes for vehicular networks. *IEEE Access*, *5*, 23710–23718.
29. Fawaz, W. (2018). Effect of non-cooperative vehicles on path connectivity in vehicular networks: A theoretical analysis and UAV-based remedy. *Vehicular Communications*, *11*, 12–19.
30. Seliem, H., Ahmed, M. H., Shahidi, R., & Shehata, M. S. (2017). Delay analysis for drone-based Vehicular Ad-hoc Networks. In *The proceeding of IEEE 28th annual international symposium on personal, indoor, and mobile radio communications (PIMRC)*, Montreal, Canada.
31. Sharma, V., Chen, H., & Kumar, R. (2017). Driver behaviour detection and vehicle rating using multi-UAV coordinated vehicular networks. *Journal of Computer and System Sciences*, *86*, 3–32.
32. Zhang, N., Zhang, S., Yang, P., Alhussein, O., Zhuang, W., & Shen, X. S. (2017). Software defined space-air-ground integrated vehicular networks: Challenges and solutions. *IEEE Communications Magazine*, *55*(7), 101–109.
33. Ghazzai, H., Menouar, H., & Kadri, A. (2017). On the placement of UAV docking stations for future intelligent transportation systems. In *The proceeding of IEEE 85th vehicular technology conference (VTC Spring)*, Sydney, NSW.
34. Messous, M., A., Arfaoui, A., Alioua, A., & Senouci, S. M. (2017). A sequential game approach for computation-offloading in an UAV network. In *The proceeding of GLOBECOM 2017, Singapore*.
35. Yuan, Z., Huang, X., Sun, L. & Jin, J. (2016). Software defined mobile sensor network for micro UAV swarm. In *The proceeding of the 2016 IEEE international conference on control and robotics engineering (ICCRE)*, Singapore, Malysia.
36. Gupta, L., Jain, R., & Vaszkun, G. (2016). Survey of important issues in UAV communication networks. *IEEE Communications Surveys and Tutorials*, *18*(2), 1123–1152.
37. Sara, M., Jawhar, I., & Nader, M. (2016). A softwarization architecture for UAVs and WSNs as part of the cloud environment. In *The proceeding of the 2016 IEEE international conference on cloud engineering workshop (IC2EW)*, Berlin, Germany.
38. Ramaprasath, A., Srinivasan, A., Lung, C. H., & St-Hilaire, M. (2017). Intelligent wireless ad hoc routing protocol and controller for UAV networks. In Y. Zhou & T. Kunz (Eds.), *Ad Hoc Networks*. Social informatics and telecommunications engineering: Lecture notes of the institute for computer sciences.
39. Masoud, M., & Belkasim, S. (2018). WSN-EVP: A novel special purpose protocol for emergency vehicle preemption system. *IEEE Transactions on Vehicular Technology, 67*(4), 3695–3700.
40. Nellore, K., & Hancke, G. P. (2016). Traffic management for emergency vehicle priority based on visual sensing. *Sensors*, *16*(11), 1892.
41. Remy, G., Cherif, M., Senouci, S., M., Jan, F. & Gourhant, Y. (2012) Lte4v2x-collection, dissemination and multi-hop forwarding. In *The proceeding of the IEEE ICC2012*.
42. Chen, X., Jiao, L., Li, W., & Fu, X. (2016). Efficient multi-user computation offloading for mobile-edge cloud computing. *IEEE/ACM Transactions on Networking*, *24*(5), 2795–2808.
43. Deng, M., Deng, H., & Lyu, X. (2016). Adaptive sequential offloading game for multi-cell mobile edge computing. In *The proceeding of the 2016 23rd international conference on telecommunications (ICT)*, Thessaloniki.
44. Chen, X. (2014). Decentralized computation ooading game for mobile cloud computing. *IEEE Transactions on Parallel and Distributed Systems*, *26*(54), 974–983.
45. Yu, R., Ding, J., Huang, X., Zhou, M. T., Gjessing, S., & Zhang, Y. (2016). Optimal resource sharing in 5G-enabled vehicular networks: A matrix game approach. *IEEE Transactions on Vehicular Technology*, *65*(10), 7844–7856.
46. Kuhn, H. W. (1953). Extensive games and the problem of information. *Annals of Mathematical Studies*, *2*(28), 193–216.
47. Schwalbe, U., & Walker, P. (2001). Zermelo and the early history of game theory. *Games and Economic Behavior*, *34*, 123–137.
48. Stetsko, A., Folkman, L., & Matay, V. (2010). Neighbor-based intrusion detection for wireless sensor network. In *The proceeding of the 6th international conference on wireless and mobile communications*, Valencia, Spain.
49. Li, Y. J., Deng, H., & Lyu, X. (2012). An overview of the DSRC/WAVE technology. In *The proceeding of the international conference of on heterogeneous networking for quality, reliability, security and robustness*.