Software-Defined heterogeneous vehicular networks: Taxonomy and architecture

Ahmed Alioua, Sidi-Mohammed Senouci, Samira Moussaoui, Hichem Sedjelmaci, Abdelwahab Boualouache

Software-Defined Heterogeneous Vehicular Networks: Taxonomy and Architecture

Ahmed Alioua*, Sidi-Mohammed Senouci**, Samira Moussaoui*, Hichem Sedjelmaci**, and Abdelwahab Boualouache*

*Department of Computer Science, RIIMA laboratory, USTHB University, Bab Ezzouar, Algiers, Algeria
**DRIVE EA1859, Univ. Bourgogne Franche-Comte, 58000, Nevers, France
Email: *aalioua@usthb.dz, Sidi-Mohammed.Senouci@u-bourgogne.fr, smoussaoui@usthb.dz, sid-ahmed-hichem.sedjelmaci@u-bourgogne.fr, aboualouache@usthb.dz*

***Abstract*— Heterogeneous vehicular networks (HetVNets) are a promising approach to meet the various communication requirements of vehicular networks' services using a variety of available access networks. However, due to their inherited characteristics, HetVNets are rigid, difficult to manage and suffer from a lack of programmability, flexibility, and scalability. In this paper, we first highlight the limits of current HetVNet architectures and show what the emerging software-defined networking (SDN) paradigm can bring to overcome these limitations with a focus on some use cases. We also provide a taxonomy of existing software-defined heterogeneous vehicular networks architectures and discuss their limits. Based on that, we continue by proposing a new SDN-based architecture for HetVNet. It is a cluster-based architecture with a semi-centralized hierarchical control and an efficient fall back recovery mechanism, that functions in both infrastructure-less and infrastructure-based covered vehicular scenarios. Finally, we demonstrate the feasibility and the efficiency of the proposed architecture compared to the related SDN-based HetVNet architectures through simulation results.**

Keywords—Heterogeneous VANET; Software Defined Networking; Distributed Controllers; Mobile Multi-Controllers.

I. INTRODUCTION

Heterogeneous vehicular network (HetVNet) [1] is a promising approach that integrates different access networks such as DSRC and cellular networks to reduce the deployment cost, support the large coverage, and meet the various quality-of-service (QoS) communication requirements of vehicular services. However, the efficient management of HetVNets is still a hard task to perform due to the inherited characteristics of vehicular networks such as high mobility, dynamic topology and the intermittent nature of wireless connections. Indeed, current HetVNet architectures suffer from a lack of programmability, flexibility and scalability [2] to efficiently deal with these increasing challenges. Thus, a new open and flexible HetVNet architecture becomes an absolute requirement to allow an efficient utilization of vehicular resources.

Software-Defined Networking (SDN) is among the emerging network architecture paradigms in the last few years. Originally designed for wired networks, SDN has known a phenomenal success both in academia and industry. SDN is based on: (i) a physical separation between the control plane and the data plane, and (ii) a logically centralized control and intelligence in a software controller. OpenFlow [3] is the most utilized standard for the communication between the control plane and data plane. It defines two kinds of networks equipment: OpenFlow controllers, which centralize all the network control functions and OpenFlow vStwichs, which only perform data packet forwarding functions. Each vSwitch has a flow table that contains controller flow entries. The controller handles the vSwitchs by formulating flow rules and installing corresponding entries in the flow table. A secure channel is used to connect the controller and corresponding vSwitchs.

Given the growing popularity of SDN, researchers are increasingly exploring the possibility of integrating SDN in mobile wireless networks and more massively into cellular mobile networks (*e.g.,* 4G/5G). Cellular networks are largely utilized in HetVNet to provide large coverage and high throughput to vehicular users. This brings new challenges as the heterogeneity of wireless infrastructures. With unified SDN network virtualization and abstraction, the integration of heterogeneous network technologies in vehicular networks became simple and transparent. Indeed, SDN promises to bring flexibility, scalability, and programmability for vehicular architectures, which can simplify the network management, reduce the cost and improve the performance of HetVNet. In this paper, we first highlight the limits of current HetVNet architectures and show how SDN paradigm can bring as a new solution to overcome these limitations with a focus on some use-cases. We also provide a taxonomy of existing software-defined vehicular architectures and discuss their limits. Based on this discussion, we continue by proposing a new SDN-based architecture for HetVNet. It is an agile cluster-based architecture adapted to both infrastructure-less and infrastructure-based scenarios. Finally, we demonstrate the feasibility and the efficiency of the proposed architecture compared to the related SDN-based HetVNet architectures via the simulation results.

II. SOFTWARE-DEFINED IN HETEROGENEOUS VEHICULAR NETWORK: OVERVIEW AND TAXONOMY

In this section, we briefly review the limits of existing HetVNets architectures, present benefits of integrating SDN paradigm in vehicular networks, describe some use cases and discuss a taxonomy of software-defined vehicular solutions.

A. Current Heterogeneous Vehicular Architecture Limitations

Although the popularity of current HetVNet architectures [1], several limitations still exist. In addition to the heterogeneity of existing network infrastructures which causes challenges in network management and integration [4], we can cite for example, *(i)* the lack of scalability: it is very difficult to deploy services on a large scale, very dense and highly dynamic topology such as vehicular networks [2], *(ii)* the lack of intelligence: the heterogeneity of the vehicular equipment and their characteristics such as no programmability and the dependency to providers in development make these architectures rigid and difficult to manage, and *(iii)* the lack of flexibility and adaptability: given the diversity of deployment environments and the heterogeneity communication technologies, it is difficult to select the adequate technology to use according to the actual context and the fast changing of network parameters. These constraints limit the system functionality, slow down the creativity and often lead to the under-exploitation of network resources. Indeed, the solutions offered by these architectures are generally adapted only for a specific context or particular situation.

B. SDN Benefits for Heterogeneous Vehicular Networks

SDN has emerged as an attractive approach to improve the flexibility, programmability, efficiency and scalability of current network architectures. In addition, to the generic advantages that it brings to both wired and wireless networks, SDN keeps some unique benefits especially adapted to vehicular networks. Indeed, with the global awareness of network topology provided by the logically centralized control, it becomes easy to efficiently allocate all types of network resources (*e.g.,* bandwidth, spectrum, power transmission, etc.). In one hand, the performance of existing vehicular architectures can be enhanced and better exploited by adapting the deployment of the most suitable solutions for each specific situation and adjusting the system parameters according to the actual context. In the other hand, with virtualization and abstraction that SDN brings, the integration of heterogeneous network technologies in vehicular networks becomes simple and transparent. In the following, we describe some use-cases that can be enhanced by software-defined HetVNets:

- **SDN-based global dynamic map:** in traditional vehicular networks, vehicles collaborate with each other and exchange different information collected by their sensors to build a *local dynamic map* (LDM) on network conditions. This LDM contains static information such as road signs and dynamic information such as traffic and weather conditions. Integrating SDN into vehicular networks allows to the controller to centrally manage and maintain a *global dynamic map* (GDM) by collecting and filtering distributed vehicle LDMs. This allows the controller to get and construct a global dynamic view over the whole (or a part) of the network state, which can be used to make more informed and refined decisions as a reply to vehicle requests.
- **SDN-based offloading:** Offloading consists of the transfer of a task to an external entity when it cannot be performed locally due to the lack of necessary resources. In traditional vehicular networks, the offloading decision is made based on a limited knowledge of the networks. Indeed, this decision is based, for example, on historical network parameters [5], which cannot reflect the current network state and even results in a non-beneficial decision. Besides, in the SDN-based vehicular networks, using the SDN controller centralized global view can dynamically make better profitable offloading decisions based on real-time networks state, which is more suitable to user's requirements and adapted to current network conditions.

C. Taxonomy Discussion on Software-defined Heterogeneous Vehicular Networks

Several software-defined heterogeneous vehicular networks (SDVN) architectures have recently been proposed. Table 1 gives a short overview of these architectures considering main characteristics such as the number of used controllers and the support from the infrastructure. Based on the number of used SDN controllers, we can classify these architectures into two categories: *(i)* Uni-controller architectures [2, 4, 6, 7, 8]: These architectures use a unique controller that can be installed somewhere in the fixed infrastructure to handle all the network demands, and *(ii)* Multi-controllers architectures [9-12]: These architectures use multiple controllers installed over the fixed infrastructure, where each one handles a part of the network.

The use of a unique controller in such dynamic, large, and dense networks as vehicular networks is challenging for several reasons we can cite for example *(i)* a serious risk of controller bottleneck confronting to the huge number of requests, *(ii)* a high installation delay of flow rules is generated especially when the distance between vehicles and the controller surpasses hundreds of kilometers, and *(iii)* omnipresent risks of reliability and security are raised especially with such intermittent and unstable nature of wireless connections. Therefore, the use of multiple controllers is more appropriate with heterogeneous vehicular networks. Indeed, some architectures [8-12] of the second category propose to install controllers as nearest as possible to vehicles in order to obtain an acceptable end-to-end delay and better, some works in [2, 4,

TABLE 1
COMPARISON BETWEEN SOFTWARE-DEFINED HETVNET ARCHITECTURES.

Existing architectures	Controller number	Centralized \ Decentralized	Fall back recovery mechanism	Delay sensitive applications	Infrastructure-less compatibility
[2]	Uni Controller	Centralized	Yes	No	No
[4]	Uni Controller	Centralized	Yes	No	No
[6]	Uni Controller	Centralized	No	No	No
[7]	Uni Controller	Centralized	No	No	No
[8]	Uni Controller	Centralized	No	Yes	No
[9]	Multi Controllers	Centralized	No	Yes	No
[10]	Multi Controller	Centralized	Yes	Yes	No
[11]	Multi Controllers	Centralized	No	Yes	No
[12]	Multi Controllers	Decentralized	Yes	Yes	No

10, 12] use fall back recovery system to ensure the availability of service if one of the controllers fails.

However, as we can notice in Table 1, all the available solutions are infrastructure-based architectures, *i.e.,* they can only operate on the infrastructure-covered zones. This assumption seems strong one, especially with the first steps deployment of HetVNets, where the total coverage is far to be achieved. Motivating by this strong assumption, we tended to propose an architecture that can operate not only in infrastructure-covered zones but also in uncovered zones. Indeed, the proposed architecture, presented in the next section, is a semi-centralized SDN-based HetVNet architecture that supports heterogeneous communications: vehicle to vehicle (V2V), vehicle to infrastructure (V2I), infrastructure to infrastructure (I2I), and infrastructure to Cloud (I2Cloud).

III. PROPOSED ARCHITECTURE

In this section, we present a novel flexible semi-centralized architecture for an SDN-based architecture for HetVNets, called *cluster-based software defined heterogeneous vehicular networks (CSDHVN)*. It is based on a combination of different promising concepts that aim to: *(i)* extend traditional vehicular network resources capabilities by integrating already existing and largely deployed cellular networks (4G/5G) and high resources Cloud computing, *(ii)* partition and reorganize the network through clustering techniques to reduce interferences and overhead, and better support scalability, density and mobility, and *(iii)* use the emerging concept of SDN with hierarchically distributed multi-controllers to mitigate network heterogeneity and introduce flexibility, programmability, and facilitate the network management. The control in this architecture is semi-centralized: centralized in covered areas (*i.e.,* infrastructure-based zones) where the coverage of the fixed infrastructure is available, and distributed multi-hops in uncovered areas (*i.e.,* infrastructure-less zones) where the coverage of the fixed infrastructure is not available. The description of this architecture is detailed in below.

A. System Architecture

The basic idea of our architecture, as illustrated in Fig. 1, is summarized as follows: *(i)* based on the availability or not of fixed infrastructure coverage, we divide the network into two zones: covered infrastructure-based areas and uncovered infrastructure-less areas, *(ii)* partition and organize the network topology according to certain criteria (*e.g.,* mobility, position, etc.) into two hierarchical clusters: *static level-1 clusters*, which are formed during the installation of cellular base stations. They have a size equals to the radio range of a base station (LTE\eNodeB) and *dynamic level-0 clusters*, which are periodically formed and updated using clustering algorithms such as in [13] for covered areas and in [14] for uncovered areas. Level-0 clusters have a maximal size equals to the IEEE 802.11p radio range and regroup vehicles which are close to each other and have similar characteristics (*e.g.,* direction, velocity, position, etc.), *(iii)* deploy a local mobile distributed controller on each level-0 cluster-head, *(iv)* deploy a central distributed controller on each level-1 cluster-head (*i.e.,* cellular base stations such as eNodeB), and *(v)* deploy a global controller on the Cloudlet server. The architecture installs the multi-controllers in the network edge, physically close to vehicles to ensure a short end-to-end response delay and support delay-sensitive (safety) services requirements

The main novelty of the proposed architecture is that it provides a complete and flexible architecture with hierarchical multi-controllers that perform in both covered areas, in which the control is centralized through three hierarchical distributed multi-controllers and uncovered areas, in which the topology is structured in clusters and the control is multi-hops distributed through the local mobile controllers at cluster-heads. In this case, local controllers collaborate and communicate with each other to construct a backhaul for system communications. The uncovered areas can also be used as a fallback recovery mechanism in covered scenarios when the connection with central controllers is lost. The proposed architecture is agile, when a vehicle leaves a covered zone to an uncovered zone and vice versa the system switches between an infrastructure-based centralized control and an infrastructure-less distributed control to ensure the continuity of service in a transparency way.

To allow SDN to operate on traditional HetVNet equipment, the architecture proposes to add an SDN module that offers necessary hardware (*i.e.,* computing and storage unit) and software (*i.e.,* SDN operating system, hypervisors and network services, etc.) resources to enable these equipment to support SDN, as illustrated in Fig. 1. To avoid adding new materials and limit the hardware modifications, the architecture prefers to reuse, if possible, the available computing and storage resources of the network equipment. The SDN architecture of our architecture is described in below.

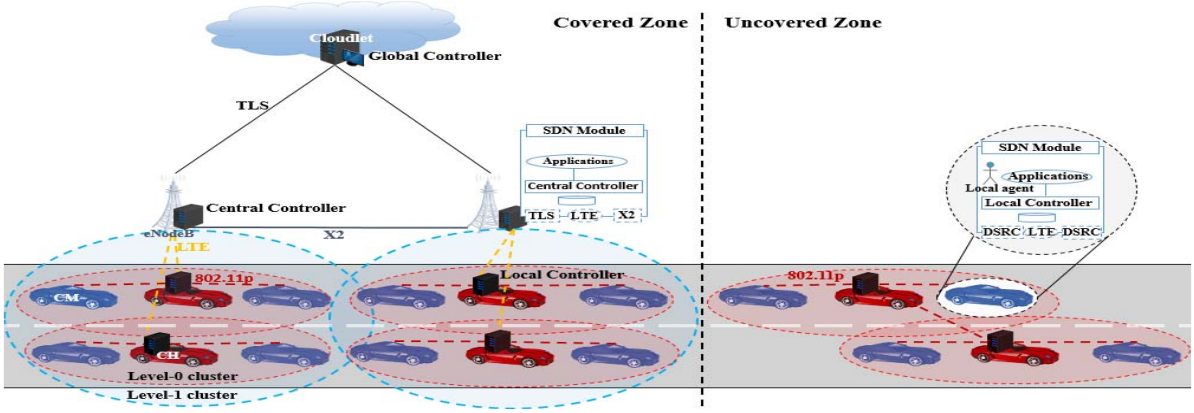


Fig. 1. Cluster-based software-defined for a heterogeneous vehicular network.

B. SDN Architecture

Our architecture is based on the three SDN layers:

1. **Data plane layer:** It consists of all network equipment that only performs the collection and the forwarding of data information, *i.e.,* mobile vehicles cluster members, roadside units (RSUs) and all network components between the base station and the cloudlet server. The data plane is divided according to the compounds' mobility into two types:

- **Fixed data plane:** It is known as *fixed* because it consists of static RSUs, which provides a support for V2V communications when two neighboring vehicles cannot communicate with each other, and all the fixed network forwarding equipment between the cellular base station and the cloudlet server.

- **Mobile data plane:** It is known as *mobile* because it consists of mobile vehicles level-0 cluster members. A vehicle compatible SDN in our architecture (or simplify a vehicle) is a traditional mobile vehicle with the SDN module to support SDN functionalities, as illustrated in Fig. 1. SDN vehicles ensure data information forwarding and the monitoring and the collect of vehicle parameters through a *software local collect and monitoring agent*. This monitored information is periodically communicated to the local controller. A vehicle can sometimes play the role of a local controller when is selected as a cluster-head. The architecture of vehicle compatible SDN is deployed at the *facilities* layer of the standard OSI CALM (communication architecture for land mobile), which assumes the existence of multiple wireless interfaces in a vehicle. Thus, each vehicle has a *flow table* and has several communication interfaces: *(i)* two broadband wireless interfaces (*i.e.,* IEEE 802.11p), one for the V2V control communications between cluster members and the local controller and the other for the data communications between vehicle cluster members and, *(ii)* a wideband wireless cellular interface (LTE/4G) used for V2I communications between local controllers and central controllers.

2. **Control plane layer:** It consists of all network equipment that centralizes the network intelligence and control, *i.e.,* multiple controllers at the cluster-head, the eNodeB and the cloudlet. In our architecture, software controllers are deployed as virtual machines on a hypervisor at the SDN module. Each controller maintains a global view of the network state of the zone that it controls. The control plane is divided according to the mobility of its compounds into two types:

- **Fixed control plane:** It is known as *fixed* because it consists of all controllers hosted on the fixed infrastructure, *i.e.,* the global controller installed on the Cloudlet server and the central controllers installed on cellular base stations (eNodeB). An eNodeB compatible SDN (or simplify eNodeB) in our architecture is a traditional eNodeB with an additional SDN module to support SDN functionalities. Each eNodeB has several communications interfaces: *(i)* two wired interfaces, an *X2* interface that connects to other eNodeB and allows I2I communications between central controllers, and an Internet interface TLS (Transport Layer Security) used for I2Cloud communications between central controllers and the global controller, and a wideband wireless cellular interface, used for infrastructure to vehicle (I2V) communications between the central controllers and the local controllers, as illustrated in Fig. 1.

- **Mobile control plane:** It is known as *mobile* because it consists of the local controllers deployed on the mobile vehicles cluster heads.

The proposed architecture uses a hierarchy of multiple controllers to alleviate high-level controllers by delegating a part of general responsibilities to lower controllers, reduce the high-density overhead and conserve bandwidth by sending to higher controllers only out-capabilities requests in an aggregated and compressed message. The proposed architecture defines three hierarchical controllers:

a. **Global controller**: It is the level-2 controller deployed on the cloudlet server. The global controller has a very powerful storage, calculation capabilities and a global view of the whole network topology. It only intervenes to handle very specific requests that require a global view of the whole network, very large resources in terms of computing/storage and operations which are bandwidth-sensitive or that cannot be served by lower-level central controllers. The global controller is considered as level-2 controller and represents the master of whole network controllers.

b. **Central controllers:** They are level-1 distributed controllers deployed on cellular base stations (eNodeB) with powerful storage and calculation capabilities. Central controllers handle specific operations that require a central view, large computing/storage resources, operations that are not delay-sensitive or operations that cannot be deserved by lower-level local controllers. They are considered as level-1 controllers and act as a slave of the global controller, a master for correspondent local controllers and equivalent to each other.

c. **Local mobile controllers**: They represent level-0 distributed controllers deployed on the SDN module on each vehicle, *i.e.,* the local controller is initially in a standby mode and activated only when its hosting vehicle becomes a cluster head. Each local controller has modest storage and calculation capabilities. Local controllers deal with local general requests, delay-sensitive requests and operations that do not need large storage/computation resources. The local controller is considered as a master in its domain, a slave of its central controller and equivalent to its local controller's neighbors. The local controller is the

unique controller that intervenes in the covered and uncovered scenarios.

3. **Service and Applications layer:** It contains all the business applications installed on the controllers.
4. **Communication interfaces:** Currently there are neither standardized SDN interfaces for directly integrating SDN into HetVNets, nor an ad-hoc (no IP-oriented) version of OpenFlow protocol compatible with security applications features of HetVNet. Thus, we propose to use a customized version of OpenFlow protocol adapted to HetVNet scenarios as southbound API to communicate between the control plane and the data plane and to use a customized interface as Northbound API to communicate between the control plane and applications [4].

C. *Fall Back Recovery Mechanism*

Among fall back recovery mechanisms for SDN in HetVNet available in the literature, one initiative [2] proposes to switch to the traditional vehicular networks when the unique controller fails. This assumption may increase the complexity of software and hardware design [12]. Another initiative [4] uses a trajectory prediction to pre-install entries in the flow table, which can be used if the controller fails. But, this initiative does not envisage any solution if the failure lasts after the end of the entry lifetime. In [12], the authors use two types of hierarchical controllers, and if the high-level controller fails, the low-level controllers collaborate together to ensure the service. This solution only considers the high-level controller failure and does not support low-level controller failures.

Our architecture is based on the use of a hierarchy of distributed multi-controllers. As a fallback recovery mechanism, we propose to permanently anticipate the possible failure of each controller at all levels and prepare the recovery scenario in advance while exploiting intelligently and efficiency the hierarchy of controllers. Therefore, a list of candidate recovery controllers is permanently managed by each local controller. The identifier (*id*) of the best candidate is periodically communicated to all level-0 cluster members and stored in a newly added recovery field in the flow table. Therefore, if an upper controller fails (level-2, *resp.* level-1), the architecture will switch to a multi-hops distributed control scheme in which the controllers of the strictly lower level (level-1, *resp.* level-0) takes over the service and collaborate with each other to ensure service continuity. If a low-level controller (level-0) fails, the pre-prepared recovery controller will take over to ensure the service continuity. Vehicles can directly start their requests to this recovery controller who is the corresponding identifier is pre-installed in their flow tables.

IV. Simulation Results

In this section, we present our simulation configuration and results. The simulation scenario is implemented using the network simulator NS3 [15] and the traffic mobility simulator SUMO [16]. The main purpose of these simulations is to demonstrate the reliability and the efficiency of the proposed architecture and study the impact of the controller deployment distance on the average flow rule installation time. For simulation, we consider a simple HetVNet scenario similar to that illustrated in Fig. 1, where the network topology is deployed in 10000 x 10000 m area. The eNodeB is situated 1000 m far from the road. The node density is 100 vehicles. Each vehicle has a velocity between 15 and 25 m/s and it is equipped with a 4G cellular interface and an IEEE 802.11p interface.

In the evaluation illustrated in Fig. 2, we simulate a scenario where a vehicle moves passing by an uncovered road segment where the connecting link with the central controller is not reachable even because of obstacles (*e.g.,* high building, tunnel, etc.) or because of the absence of the fixed infrastructure. Afterward, we evaluate how our architecture (CSDHVN) with the multi-level hierarchical controllers reacts to this situation. We compare the performance of the proposed architecture with others SDVN infrastructure-based works in [4-9], [11],[12] (we called, *SDVN Infrastructure-based works*), that deploy the controller on the fixed vehicular infrastructure according to the packet delivery ratio, which represents the ratio of packets successfully received to the total sent.

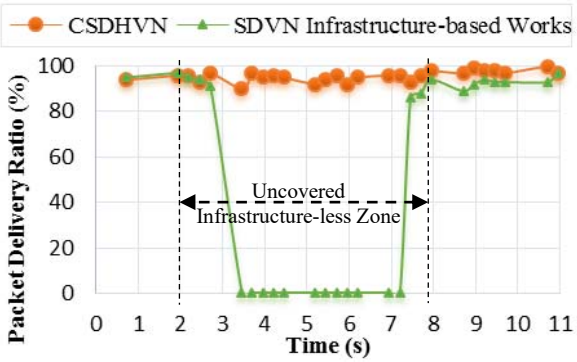
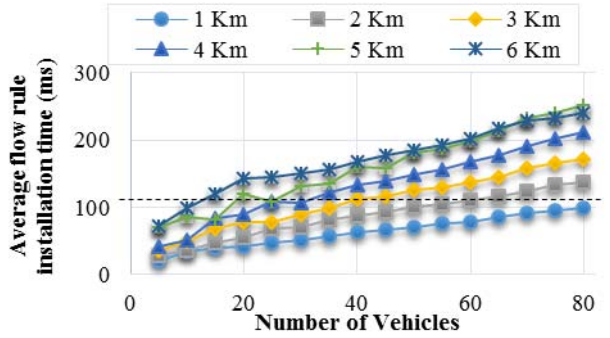


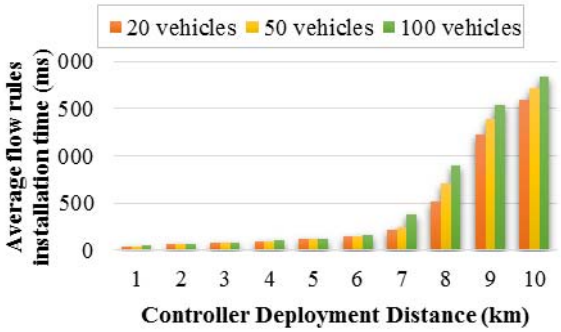
Fig. 2. Effect of uncovered zones on packet delivery ratio.

Fig. 2 shows that just when the vehicle enters into the uncovered infrastructure-less zone, the packet delivery ratio in SDVN infrastructure-based works starts to decrease dramatically and it still drops until the vehicle leaves the uncovered zone and the connection with the controller is established once again, after that, the system resumes a good delivery ratio. Contrarily, our CSDHVN maintains a good packet delivery ratio. This behavior can be justified by the fact that in the related infrastructure-based SDVN solutions in [4-9], [11], [12], the SDN controller is installed somewhere on the fixed vehicular infrastructure and if the connection with this controller is lost or it is unavailable, the service will be interrupted due to the centralized control logic of SDN. More efficient, our CSDHVN architecture ensures a good packet delivery ratio even in the covered infrastructure-based zone or in the uncovered infrastructure-less zone, thanks to the multi-level hierarchical controllers and specially the local controller on the cluster head vehicle that takes over to ensure the service continuity when the connection with the controller on the fixed infrastructure is lost. This evaluation demonstrates the reliability and the efficiency of our CSDHVN architecture and confirms that the consideration of infrastructure-less zone in the design of SDVN architecture is primordial given the negative effect that represents on the packet delivery ratio.

We study in the evaluation illustrated in Fig.3, the impact of the physical distance of controller deployment from the vehicles, on the flow rule installation time, which represents the elapsed time since a vehicle requests the controller for a flow rule and the time when it is installed in the flow table. Thus, we focus on a scenario of a set of vehicles connected to a controller and we varied the distance between the vehicles and the controller for different vehicles number.



(a)



(b)

Fig. 3. Effect of Controller deployment distance on flow rule installation time.

Fig. 3 shows that the average flow rule installation time increases with the increase of vehicles number and the controller deployment distance. Also from Fig. 3. we can conclude that to ensure a flow rules installation time sufficient to satisfy the lower latency requirement of the most of vehicular road safety applications (≤ 100 ms [1]) in such dense networks as HetVNet, the SDN controller must be installed on the edge of network, the nearest as possible from the vehicles (no far than 5 km). Our CSDHVN architecture is based on multi-level hierarchical controllers. The low-level controller (local controller) is installed on the cluster head vehicle, no more than 300 m far of vehicles and the second, central controller, on the eNodeB (far about 1 km). This can ensure a low flow rules installation time and better handles the delay-sensitive and low latency road safety applications.

V. CONCLUSION

Based on the taxonomy discussion of SDN-based HetVNet architectures, we conclude that the current general tendency should be the use of multiple distributed controllers installed on the edge of the network with an efficient fall back recovery mechanism. Moreover, the uncovered infrastructure-less zones should be considered in the design of SDN-based HetVNet architectures. Consequently, we present in this paper a semi-centralized flexible SDN-based HetVNet architecture with hierarchical multi-controllers installed on the edge of the network. Our SDN-based architecture is robust thanks to an effective fallback recovery mechanism, and works in both covered and uncovered areas. We also demonstrate the feasibility and the efficiency of our proposed architecture compared to the related SDVN infrastructure-based works based on simulation results.

The implementation complexity and cost, the high mobility of vehicles, and the dense network topology, rest open challenges in front of SDN integration in VANETs. As future work, we plan to perform more intensive experimentations to proof the efficiency of the proposed architecture.

REFERENCES

[1] K. Zheng, Q. Zheng, P. Chatzimisios, W. Xiang, and Y. Zhou, "Heterogeneous Vehicular Networking: A Survey on Architecture, Challenges, and Solutions," in *IEEE Communications Surveys & Tutorials*, vol. 17, no. 4, 2015, pp. 2377-2396.

[2] I. Ku, Y. Lu, M. Gerla, R. L. Gomes, F. Ongaro, and E. Cerqueira, "Towards software-defined VANET: Architecture and services," *13th Annual Mediterranean Ad Hoc Networking Workshop*, 2014.

[3] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "Openflow: enabling innovation in campus networks," SIGCOMM Comput. Commun. Rev., vol. 38, no. 2, 2008, pp. 69–74.

[4] Z. He, J. Cao, and X. Liu, "SDVN: enabling rapid network innovation for heterogeneous vehicular communication," in *IEEE Network*, vol. 30, no. 4, July-August 2016, pp. 10-15.

[5] S. Nirjon, A. Nicoara, C.-H. Hsu, J. P. Singh, and J. A. Stankovic, "Multinets: A system for real-time switching between multiple network interfaces on mobile devices, " ACM Trans. Embed. Comput. Syst., vol.13, no. 4s, 2014, pp. 121:1–121:25.

[6] Y. C. Liu, C. Chen, and S. Chakraborty, "A Software Defined Network architecture for GeoBroadcast in VANETs," *IEEE International Conference on Communications (ICC)*, London, 2015, pp. 6559-6564.

[7] X. Huang, J. Kang, R. Yu, M. Wu, Y. Zhang, and S. Gjessing, "A Hierarchical Pseudonyms Management Approach for Software-Defined Vehicular Networks," *IEEE 83rd Vehicular Technology Conference (VTC Spring)*, Nanjing, 2016, pp. 1-5.

[8] X. Wang; C. Wang, J. Zhang, M. Zhou, and C. Jiang, "Improved Rule Installation for Real-Time Query Service in Software-Defined Internet of Vehicles," in IEEE Transactions on Intelligent Transportation Systems , vol.PP, no.99, 2016, pp.1-11.

[9] Q. Zheng, K. Zheng, H. Zhang, and V. C. M. Leung, "Delay-Optimal Virtualized Radio Resource Scheduling in Software-Defined Vehicular Networks via Stochastic Learning," in *IEEE Transactions on Vehicular Technology*, vol. 65, no. 10, Oct. 2016, pp. 7857-7867.

[10] N. B. Truong, G. M. Lee, and Y. Ghamri-Doudane, "Software defined networking-based vehicular Adhoc Network with Fog Computing," *IFIP/IEEE International Symposium on Integrated Network Management (IM)*, Ottawa, 2015, pp. 1202-1207.

[11] M. A. Salahuddin, A. Al-Fuqaha, and M. Guizani, "Software-Defined Networking for RSU Clouds in Support of the Internet of Vehicles," in *IEEE Internet of Things Journal*, vol. 2, no. 2, 2015, pp. 133-144.

[12] A. Kazmi, M. A. Khan, and M. U. Akram, "DeVANET: Decentralized Software-Defined VANET Architecture," *IEEE International Conference on Cloud Engineering Workshop (IC2EW)*, Berlin, 2016.

[13] G. Remy, M. Cherif, SM. Senouci, and F. Jan, Y. Gourhant, "LTE4V2X: LTE for a Centralized VANET Organization," IEEE GLOBECOM'2011, Houston, Texas, 2011, pp. 5-9.

[14] G. Remy, M. Cherif, SM. Senouci, F. Jan, and Y. Gourhant, "LTE4V2X - Collection, dissemination and multi-hop forwarding," IEEE ICC'2012, Ottawa, Canada, 2012, pp. 10-15.

[15] Network simulator 3 (ns-3), http://www.nsnam.org

[16] Simulation of urban mobility (sumo), http://sumo-sim.org/.