A Mobility-based Switch Migration Scheme for Software-Defined Vehicular Networks

Noura Aljeri and Azzedine Boukerche
PARADISE Research Laboratory, EECS, University of Ottawa, Canada
naljeri, boukerch@uottawa.ca

ICC 2021 - IEEE International Conference on Communications | 978-1-7281-7122-7/21/$31.00 ©2021 IEEE | DOI: 10.1109/ICC42927.2021.9500726

***Abstract*—Software-defined vehicular networks (SDVNs) have been a vital addition to the design of intelligent vehicular networks. SDVNs elevate the constraints of static hardware network devices to programmable units, provide a global view of the network status, and standardize the interface between different wireless access technologies. However, the static deployment and assignment of switches to control units do not consider vehicles rapid mobility and diverse densities. In this paper, we propose a mobility-based switch migration scheme for software-defined vehicular networks. The proposed scheme utilizes the vehicles' mobility between switch-enabled roadside units to efficiently migrate selected switches to different controllers. The proposed scheme has been evaluated using the network simulator and reported its performance under realistic mobility traces and environment.**

***Index Terms*—Software-defined vehicular networks, mobility, switch-migration, SDVN**

I. Introduction

Intelligent Vehicular Networks (IVNs) combine various radio access technologies (e.g., WiFi/6, Cellular 5G, WiMAX) to support the massive data load through numerous safety and infotainment applications [1]. However, several challenges arise when adopting IVNs, including vehicles' high mobility, rapid topology changes, and lack of flexibility. Moreover, different applications may require various resource requirements [6]. For example, time-sensitive applications such as map rendering and emergency services require fast and reliable communication, as well as an optimized and up-to-date route to destinations. Hence, a generalized design is needed to mitigate the challenges of vehicular network's diverse heterogeneity.

Software-defined vehicular networks (SDVNs) have been a vital addition to the design of wireless networks [2]. The SDN structure provides an efficient schema to update and manipulate data forwarding rules and control signaling through controller units without service provider's physical hardware interference. However, the design of a centralized control introduces several challenges, including scalability issues and bottleneck problems. Hence, the distributed design idea of deploying multiple control units elevates some difficulties, yet with the cost of other open issues [3]. That includes the optimal number of controllers required, switch assignment issues, as well as switch-migration management.

This paper is partially supported by NSERC, CREATE TRANSIT, and Canada Research Chairs.

This paper proposes a mobility-based switch-migration scheme for software-defined vehicular networks that utilize the mobility information of vehicles' movement between switch-enabled roadside units. The proposed scheme's main objective is to adequately adjust the overall switch-to-controller assignment to fit the vehicles' mobility environment. Moreover, the proposed scheme reduces the total vehicles' transition-delay between switches and controllers.

The rest of this paper is organized as follows. Section II presents an overview of software-defined vehicular networks and recent studies on switch-migration. Section III discusses the network model preliminaries and the relationship between SDVN entities. In Section IV, we describe the proposed mobility-based switch-migration protocol for SDVN, followed by the performance evaluation in Section V. Section VI concludes the paper and point out future research directions.

II. Background

In this section, we discuss the current studies that address switch-migration issues for software-defined vehicular networks. A general overview of SDVN design is described along with its related entities.

The general design of software-defined vehicular networks (SDVN) is based on three main entities, controllers, switch-enabled roadside units, and vehicles. In some related studies, the vehicle is either considered a host entity attached to the nearest switch-enabled road unit or a mobile switch entity [4] [15]. A general overview of SDVN design is presented in Figure 1. The SDVN layers separate the control units from the infrastructure units and vehicles, in which the former is in charge of all switches rules and policies [16]. The roadside units and vehicles are part of the data layer, which only acts as forwarding devices. However, a single control unit's deployment to manage the whole network is not an efficient solution when we consider vehicular networks environment. The rapid mobility of vehicles and continuous communication disruption impacts the overall performance of the network and its scalability limits. The distributed design for SDVN was later proposed [7] [5] to reduce the load on the control unit as well as avoid a single point of failure. Nonetheless, the distributed design came with several challenges and issues, especially with vehicular network's varying densities and rapid mobility.

When we consider the distributed design of SDVN, several questions arise, such as the number of required controllers, switch-to-controllers assignment strategy, and more. In case of overloaded control or failure in one control unit, one crucial question is the switches' capability of shifting their link to new controling units. That is usually referred to as the switch-migration process. Now, the static assignment concept between controllers and switches is more stable in closed environment networks such as institutions or wired networks. However, when dealing with rapid mobility environments such as vehicular networks, a more dynamic approach is preferable to adapt according to the current network status.

Several studies investigated switch-migration solutions over different network constraints [10] [17] [12]. The reason for migration can be described in three cases as follows. i) congested traffic on controllers. ii) controller failure or shutdown; in this case, switches must migrate to another control unit. iii) mobile-switches moving further away from their current control unit and require a new connection. Basically, most decision-making solutions involve traffic load balancing issues. The elastic distribution of SDN control unit is a promising approach to handle control units failure and unbalanced traffic load [11] [13]. Dixit et al. [14] proposed a load balancing-based switch-migration solution, namely Elasticon. The proposed scheme triggers the migration process when controllers are overloaded and change their master control unit. Other studies improve the migration efficiency using simulated annealing techniques [9].

However, existing migration solutions do not consider other factors, especially in vehicular networks environment, such as vehicles' mobility characteristics and movement patterns. To the best of our knowledge, limited work has been done toward switch-migration solutions that better fit the vehicular network environment.

III. Preliminaries

Software-defined vehicular networks can be modelled using an undirected connected graph $G = (S, L)$ with S switches $S = s_1, s_2, .., s_n$ and L links that describe the connection between two switches $s_i, s_j \in S$. The link L is typically defined by the distance between two switches, whereas in vehicular networks, that link L can be described as the number of hops it takes to reach s_j from s_i. Other metrics define the link relationship between two switches, including the traffic load, link stability, and more. For this paper's purpose, we consider the link between two switches to be the number of hops, in which one-hop is a direct link between two nodes. We define a set of control units C, which includes $c_1, c_2, ..c_k$ with K controllers and are assigned a set of switches X, a subset of S. Based on the general description of software-defined network functionalities, each controller can communicate with switches through the southbound API and the application layer through the northbound API. The controllers claim their set of switches through packet_out messages, and switches respond with packet_in messages. This process is referred to as the neighbor discovery process. Switches shares their available port numbers, which allow controllers to send discovery packets to each port with a flow rule to generate an overview graph of the network. After the initial setup is finished, each controller is aware of its switches list and can now add/update flow tables on switch-enabled roadside units.

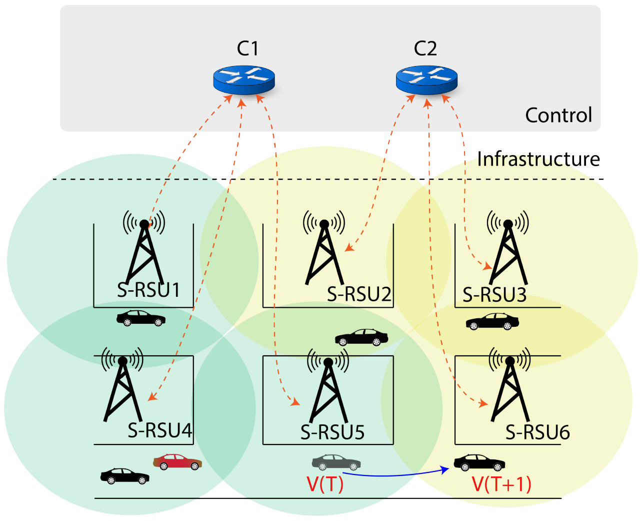

Fig. 1: General overview of DSVN.

In order to allow switches to migrate between different controllers, two new flow rules are added to every switch and control unit to facilitate the migration process. First, the session update message is generated by each vehicle transferring from one switch to another and sent to the old switch via the new switch. Second, a migration trigger message is initiated (based on gathered mobility information) and generated by each switch every period of time and sent to their corresponding control unit. Furthermore, we assume that each switch maintains a cache list of its neighboring switch-enabled RSUs along with a time-stamped mobility rate and the number of vehicles moving between them. The control units also maintain a list of assigned switches as well as a list of migration requests. The latter contains the requester, target switch, and reason of migration with a time-stamp value.

To measure the movement of vehicles in terms of mobility rate, we assume vehicles are aware of their real-time location through embedded GPS sensors and can trigger their migration process to the target switch unit based on the received signal strength. Upon vehicles' reception of advertisement packet from nearby switch-enabled RSUs, it will initiate a registration request to the selected S-RSU. When the vehicle receives a reply from the target S-RSU, it will know the switch identification and its corresponding control unit. Vehicles will compute their session-time with the current S-RSU when they are first connected and stop it after disconnection. This information will be later used to measure the mobility rate between two neighboring S-RSUs.

IV. The Proposed Dynamic Topology Control Scheme

This section describes the proposed mobility-based switch migration protocol for software-defined vehicular networks

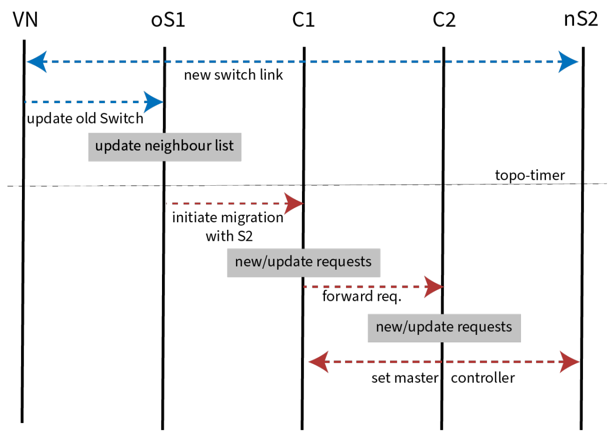

Fig. 2: switch-migration process

Algorithm 1 Neighbor Discovery and Session Update

```
Require: cTime, infoBS, update_packet
Require: p =(S(oS,oC),Dest(nS,nC),SR(in,out),pktNum)
 1: while recv update_packet (p) do
 2:     if addr == p.dest(nS) then
 3:         found=0, index =-1
 4:         for x ∈ infoBS do
 5:             if x.id == p.id then
 6:                 found=1, index=x.indx
 7:                 break
 8:             end if
 9:         end for
10:         if !found then
11:             infoBS.add(p.oS,p.oC, countV=1, flag=1, sR=(out-in), updateT=cTime)
12:         else
13:             if infoBS[index].flag==-1 and p.nC != infoBS[index].C then
14:                 new controller is not yet valid
15:             else
16:                 infoBS[index].update(flag=1,countV+1,nC, sR+(out-in),updateT=cTime)
17:             end if
18:         end if
19:     else
20:         forward update packet to destination
21:     end if
22: end while
```

(SDVN). The proposed protocol works in three main steps: periodic neighbor discovery, session update process, and the migration process.

In what follows, we define each process step and requirements. Assuming that vehicles can migrate between controllers based on their location or region, vehicles are attached to the nearest switch-enabled roadside unit (S-RSU) and given access to its corresponding control unit. We assume switch-enabled roadside units are periodically advertising their availability through a wireless channel. S-RSU changes its controller when the mobility rate between itself and neighboring S-RSU is above a threshold value, in which a migration trigger is initiated and processed. The mobility rate is defined by the average total time it takes vehicles to migrate between two neighboring S-RSUs.

A. Neighbor discovery and session update

When vehicles move between one switch-enabled roadside unit (S-RSU) to another, they need to also transition their current flow with the previous S-RSU to the new one. This process is generally referred to as the handover process. In software-defined vehicular networks, the transition process of ongoing communication between any given vehicle and a service is referred to as migration of service or flow. Hence, S-RSU and the control unit must transfer and update the vehicle's new flow information to be forwarded to the new S-RSU. In our work, we utilize this process to help S-RSUs and controllers to discover more neighbors.

Vehicles can discover new S-RSUs through periodic broadcast or solicitation messages. Therefore, when vehicles first discover that they have a new link with nS-RSU, it sends an update message to the previous oS-RSU that contains the following: source address, a destination address, vehicles' new connection (nS-RSU), and its corresponding control unit (nC), in addition to the vehicles' session-time within the previous oS-RSU before the transition. A detailed algorithm of the neighbor discovery process is presented in Algorithm 1. Meanwhile, controllers periodically update other controllers with updated locations of vehicles. This step helps each controller obtain a global connectivity graph of the network entities and the real-time vehicle location.

Upon receiving the vehicles' update message, the old controller will either create a new entry for the nS-RSU or update a previously collected update message with a new time-stamp. Moreover, S-RSU also maintains session information entry for each neighboring S-RSU, which holds the number of vehicles that have transferred to them and the total session time. As presented in Algorithm 1, the vehicle sends an update message to its current link nS-RSU, which will be forwarded to the old oS-RSU for an update. The previous oS-RSU will update or add a new entry to its list of neighbors along with the number of vehicles transferred and their total session time. This information will assist in evaluating the average mobility rate between two neighboring S-RSUs. Thus, a decision can be made whether they should belong to the same controlling domain or not. If two S-RSUs are from different control domains and vehicles are transferring between them more rapidly, then the cost will increase, and the delay will be higher as new switch-enabled RSUs will need to learn how to handle the upcoming vehicle through the control unit. The time that takes the control unit to transfer vehicles' flow table from one switch to another in the same domain takes less time than moving to another domain. Not to mention, the vehicles' rapid

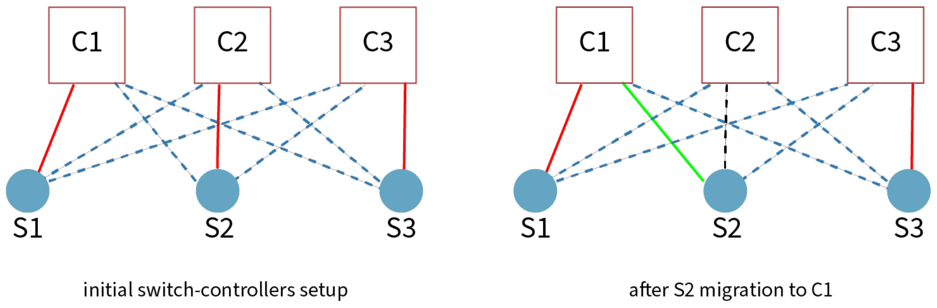

Fig. 3: Switch-to-Controllers assignment before and after migration process.

Algorithm 2 Periodic Monitoring Process

```
Require: addr, infoBS, $\alpha$, cTime
 1: min= MAXINT
 2: index=-1
 3: if $len(infoBS) > 2$ then
 4:     for $x \in infoBS$ do
 5:         if x.c != addr.c and x.sR $<$ min then
 6:             min = x.sR
 7:             index= x.indx
 8:         end if
 9:     end for
10:     if index != -1 and min$<$ $alpha$ and infoBS[index].flag !=-1 then
11:         migrate switch x to current addr.controller
12:         infoBS[index].flag=-1
13:         initiate_migration(infoBS[index].id)
14:     end if
15: end if
16: update Topo_timer(Tx)
```

movement will have a significant impact on service disruption and transfer failure if not handled efficiently.

B. Switch migration process

The critical part of the proposed scheme is how to handle the migration of switch-enabled RSUs from one controller to another with minimum overhead and delay. The switch migration process is detailed in Algorithm 2, which is triggered periodically by S-RSUs every time-interval λ. The selection of how frequently the switch should monitor the current mobility rate measurement and acts upon it is curial. In our scheme, we believe the time-interval should reflect the average time vehicles reside within S-RSU. We further discuss the time-interval selection strategy in Section V.

The switch-migration process is based on the gathered information by each switch-enabled roadside unit. Each S-RSU calculates the mobility rate by the average session time

Algorithm 3 Migration

```
Require: migration_request packet ($pkt$), $reqList$, $cList$
 1: while $pkt$ do
 2:     if $pkt.targetS \in reqList$ then
 3:         i= reqList.index(pkt.targetS)
 4:         validate the migration request
 5:         if $pkt.sR < reqList[i].sR$ then
 6:             update request list $reqList$
 7:         else
 8:             return
 9:         end if
10:     else
11:         add new request list $reqList$
12:     end if
13:     if $c$ != $pkt.targetC$ then
14:         forward packet to $pkt.targetC \in cList$
15:     else
16:         Initiate role change process
17:         change $pkt.targetS$ master controller to $pkt.newC$
18:     end if
19: end while
20: procedure AT TARGET S-RSU
21:     $targetS$ receives migration message
22:     $targetS$ sets $newC$ as new master controller
23:     $targetS$ sets UpdatedControl=1
24:     all future advertisement will contain $newC$ address
25: end procedure
```

between itself and a neighboring S-RSU outside its domain. For example, in Figure 3, S1 and S2 are neighbors with two different control units. Let us assume that S1 has the neighbor list as presented in Figure 3. It will calculate the average session time of S2 and S3 based on the total session time and the number of vehicles passed between them. Then, find the minimum session time among S2 and S3 for evaluation. Currently, we only perform one switch-migration per S-RSU;

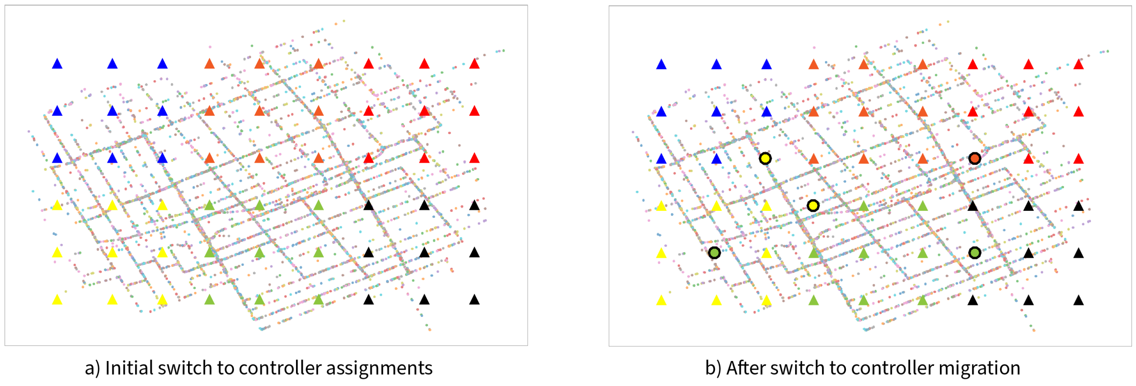
a) Initial switch to controller assignments
b) After switch to controller migration

Fig. 4: Simulation scenario of the City of Ottawa urban environment. a) S-RSU initial deployment scenario.b) After migration assignments (for interval $100s$, $alpha = 5s$).

however, in some cases, more than one neighbor might be eligible for migration at the same time. That will be considered in future versions of the proposed scheme. Now, the minimum session value of S1 neighbor is compared either against a threshold value $alpha$ or the average session time within S1's domain neighbors.

When the average session value is below a threshold, a migration trigger is sent by S-RSU to its controller containing the neighbor switch identification and corresponding control unit. The controller will then update or create a new entry to the list of requests sent by S-RSU holding the requester, target S-RSU, reason, and time-stamp. The idea here is to avoid duplicate migration requests to the same S-RSU and avoid the ping-pong effect. The switch is migrated between two controllers within relatively small gaps and close session values. The control unit will then forward the migration request to the other control unit in which the target S-RSU currently resides. The final decision of migration is upon the target S-RSU control unit based on previous requests. Finally, the target S-RSU's control unit sends a message to the new controller and notify the target S-RSU of the recent migration request. Following that, the two controllers process flow migrations between them, and the target S-RSU has a new master control unit. At this moment, the target S-RSU is now advertising itself with a new master controller and send/receive flow table information from the new control unit.

V. Performance Evaluation

This section evaluates the performance of the proposed switch-migration scheme for software-defined vehicular networks based on vehicles' mobility measurements using the network simulator NS-2. Each vehicle and S-RSU are given wireless communication capabilities through the IEEE802.11p standard protocol. The implementation of the proposed scheme is introduced in the network layer. In this paper, we use direct communication between vehicles and S-RSUs.

We simulate vehicles' mobility using SUMO and the city of Ottawa's urban environment from OpenStreetMaps (OSM). The scenario consists of 700 vehicles with a speed of 0-30 m/s. The selected area of simulation covers 3000x3000 m and assigned 54 switch-enabled RSUs (S-RSU), as seen in Figure 4(a), with the communication range set to $250m$. In this set of experiments, the deployment of S-RSUs is based on a greedy approach that guarantees maximum coverage over the selected area. However, since currently, this issue is outside the scope of the proposed work, we plan to efficiently deploy infrastructure units based on the context-awareness of the selected areas and vehicles' densities in future studies. The number of controllers is set to 6 and are initially assigned to switches in a block-based manner. There are other solutions to how we can optimally select each cluster based on k-Means clustering method [8].

Three time-interval values are used for testing, 30 s, 50 s, and 100 s. Surely, introducing more frequent time-interval may produce finer granularity, yet with the cost of triggering very early migration that might not be required later on. Also, as the time-interval increase, the migration trigger may miss needed opportunities to initiate a switch-migration process during a specific period. To better understand the relationship between the time-interval value and the migration process, we evaluate the performance of each time-interval under different threshold values.

In Figure 5, we display the percentage of migrations triggered (orange line) with different time-intervals, as well as the average delay in vehicles' migration between switch-enabled RSUs. As noticed, as the time-interval increases, more migration triggers are presented, specifically when we look at threshold value $alpha = 5$. Moreover, a different threshold value is also considered in Figure 5, that triggers a migration request based on the domain neighbors session to mobility rate, referred to as Avg. The idea was to explore the possibility of not relying on fixed threshold values to trigger migration requests and rely on the overall average mobility rates among S-RSUs in the same controller domain. Although the results did not present better values than thresholds 5 and 3; however, the change in its percentage values did not profoundly impact the cost in vehicles' migrations. Furthermore, because of the introduction of the requests cache table on the controller

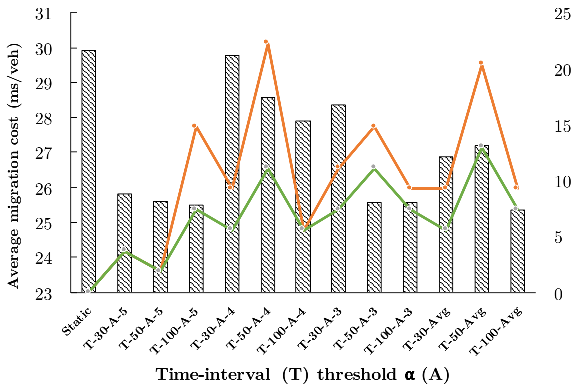

Fig. 5: Performance evaluation of the vehicles' migration delay and percentage of switch-migrations.

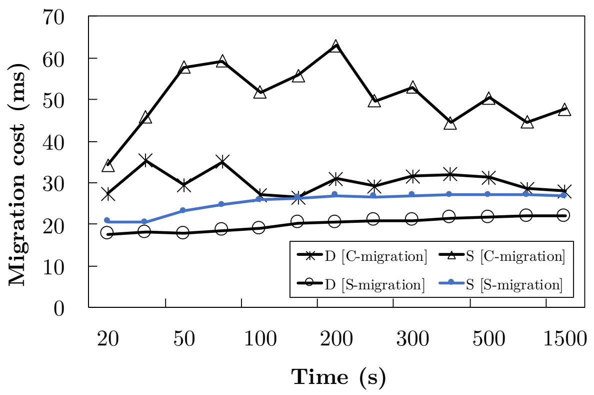

Fig. 6: Comparison of Static (S) and Dynamic (D) switch-migration: Switch(S)-migration and Controller(C)-migration, Dynamic using T-interval (100s) and Threshold(5s).

units, we have avoided the trigger of redundant migration requests and ping-pong events between S-RSUs, as seen in Figure 5 (green line).

Figure 6 portrays vehicles' migration cost in terms of delay of the proposed mobility-based method and the static deployment of S-RSUs. As seen, the change in switch-to-controller assignment during simulations based on vehicles' mobility between them has reduced the cost of inter-domain movement by 40%. This is due to the fact that the proposed method eliminates controller migration when high mobility is presented between two neighboring switches. By including these switches under the same umbrella of control, vehicles will not lose attachment to the domain controller and only change their local attachment to sub-domain S-RSU. Therefore, no migration failure can occur, and a faster migration process is exhibited. However, because more switches are moving between controller units, a slight increase in sub-domain migration is presented.

VI. Conclusion

In this paper, we proposed a mobility-based switch-migration scheme for software-defined vehicular networks. The proposed scheme utilizes vehicles' mobility between switch-enabled roadside units to adapt switch-to-controller associations to reduce vehicles' transition cost and latency. The performance evaluation was conduct under the city of Ottawa's urban environment and realistic vehicles' mobility. The results indicated that the proposed switch-migration process reduces the overall vehicles' migration delay. The selection of the most appropriate time-interval and threshold values minimizes the overhead of unnecessary migration triggers. In future works, we plan to investigate the impact of switch-enabled RSUs location deployment on the migration process and include multiple factors in the decision-making process.

References

[1] Younes, M.B. and Boukerche, A., 2019. Safety and efficiency control protocol for highways using intelligent vehicular networks. *Computer Networks*, 152, pp.1-11.
[2] Aljeri, N. and Boukerche, A., 2020. Mobility Management in 5G-enabled Vehicular Networks: Models, Protocols, and Classification. *ACM Computing Surveys* (CSUR), 53(5), pp.1-35.
[3] Oktian, Y.E., Lee, S., Lee, H. and Lam, J., 2017. Distributed SDN controller system: A survey on design choice. *computer networks*, 121, pp.100-111.
[4] He, Z., Cao, J. and Liu, X., 2016. SDVN: Enabling rapid network innovation for heterogeneous vehicular communication. *IEEE network*, 30(4), pp.10-15.
[5] Liyanage, K.S.K., Ma, M. and Chong, P.H.J., 2018. Controller placement optimization in hierarchical distributed software defined vehicular networks. *Computer Networks*, 135, pp.226-239.
[6] J. Chen, and et al., Service-oriented dynamic connection management for software-defined internet of vehicles, 2017. *IEEE Trans. Intell. Transp. Syst.* 18 (10), pp.2826–2837.
[7] Correia, S., Boukerche, A. and Meneguette, R.I., 2017. An architecture for hierarchical software-defined vehicular networks. *IEEE Communications Magazine*, 55(7), pp.80-86.
[8] G. Wang, Y. Zhao, J. Huang, Q. Duan, and J. Li, A K-means-based network partition algorithm for controller placement in software defined network, in *IEEE International Conference on Communications*, 2016, pp. 1-6.
[9] T. Hu, J. Lan, J. Zhang, and W. Zhao, EASM: Efficiency-aware switch migration for balancing controller loads in software-defined networking, 2019 *Peer-to-Peer Netw. Appl.*, vol. 12(2), pp. 452–464.
[10] Wang, C.A., Hu, B., Chen, S., Li, D. and Liu, B., 2017. A switch migration-based decision-making scheme for balancing load in SDN. *IEEE Access*, 5, pp.4537-4544.
[11] Dixit, A.,and et al., 2014, October. ElastiCon; an elastic distributed SDN controller. In 2014 ACM/*IEEE Symposium on Architectures for Networking and Communications Systems*, (pp. 17-27).
[12] Cheng, G., Chen, H., Wang, Z. and Chen, S., 2015, May. DHA: Distributed decisions on the switch migration toward a scalable SDN control plane. In 2015 *IFIP Networking Conference (IFIP Networking)* (pp. 1-9).
[13] Garg, S., and et al., 2019. MobQoS: Mobility-aware and QoS-driven SDN framework for autonomous vehicles. *IEEE Wireless Communications*, 26(4), pp.12-20.
[14] Dixit, A., Hao, F., Mukherjee, S., Lakshman, T.V. and Kompella, R., 2013. Towards an elastic distributed SDN controller. In Proceedings of *the second ACM SIGCOMM workshop on Hot topics in software defined networking* (pp. 7-12).
[15] Wenjie Li, Zheng Qin, Hui Yin, Rui Li, Lu Ou, and Heng Li. 2016. An Approach to Rule Placement in Software-Defined Networks (*MSWiM '16*). 115–118.
[16] Liangxiao Xin, Johannes K. Becker, Stefan Gvozdenovic, and David Starobinski. 2019. Benchmarking the Physical Layer of Wireless Cards Using Software- Defined Radios (*MSWIM '19*), 271–278.
[17] Liang, C., Kawashima, R. and Matsuo, H., 2014, December. Scalable and crash-tolerant load balancing based on switch migration for multiple open flow controllers. In 2014 *Second International Symposium on Computing and Networking* (pp. 171-177)