Vehicular Communications 31 (2021) 100372

Contents lists available at ScienceDirect

Vehicular Communications

www.elsevier.com/locate/vehcom

Exploring software defined networks for seamless handovers in vehicular networks

Miguel Silva [a], Pedro Teixeira [a,b], Christian Gomes [a], Duarte Dias [a,b], Miguel Luís [a,c,*], Susana Sargento [a,b]

[a] Instituto de Telecomunicações, 3810-193 Aveiro, Portugal
[b] Departamento de Eletrónica, Telecomunicações e Informática, Universidade de Aveiro, 3810-193 Aveiro, Portugal
[c] ISEL - Instituto Superior de Engenharia de Lisboa, Instituto Politécnico de Lisboa, 1959-007 Lisboa, Portugal

ARTICLE INFO

Article history:
Received 10 February 2021
Received in revised form 15 April 2021
Accepted 9 May 2021
Available online 17 May 2021

Keywords:
Software-defined Networks
Vehicular ad-hoc Networks
Mobility management
Horizontal handovers

ABSTRACT

With the growing interest in autonomous driving, constant connectivity for vehicles is becoming essential to enable the complete knowledge of the surrounding area, transmit and receive data that is crucial for the autonomous control. The vehicle mobility results in frequent service interruptions, and therefore, seamless handovers are required to mitigate this problem. Several IP-based solutions have been proposed in the literature, but they require tunneling approaches, which present excessive signaling and data overhead, service delay, and packet loss. One of these approaches, the NEMO-enabled Proxy Mobile IPv6 (N-PMIPv6) architecture, supports transparent handovers and simultaneous multi-homing, but at the cost of a high complexity and network overhead.

This work explores the flexibility of Software Defined Networks (SDNs) in the management of a Vehicular Ad-hoc NETwork (VANET). In particular, the SDN concept is used to provide a seamless horizontal handover for the vehicle and its end-users. Two different SDN architectures are proposed, evaluating the impact of the depth of the softwarization environment. Real vehicular hardware and emulated mobility scenarios are used in the evaluation process where different application services are exploited. Results show that the lower complexity of the SDN solution allows for a better performance during a handover in a VANET, in terms of delays, packet losses and network overhead, making it seamless for the vehicles and its users.

1. Introduction

The recent advances in the automotive industries and telecommunication technologies brought focus on Intelligent Transportation System (ITS), which are part of the services required for autonomous driving, of which Vehicular Ad-hoc NETworks (VANETs) gain much more attention. One of the most researched topic regarding VANETs tackles the end-to-end connectivity of Vehicle-to-Infrastructure communication flows [1]. VANETs are highly-dynamic networks whose connectivity profile is influenced by the speed and direction of its moving elements, On-Board Units (OBUs) equipped in vehicles, and the profile of the network infrastructure, where Road Side Units (RSUs) bridge the gap between the core network and the moving network entities. Therefore, horizontal handovers are much more frequent than in traditional Wireless Local Area Networks (WLANs): an OBU with a connection established with an RSU (also denoted as Point-of-Attachment (PoA) under this scope) will soon be out of its range and establish a new connection with a different PoA.

For many years the Internet Protocol was not prepared to deal with the mobility of its users, as it was implemented assuming that each element would be always in the same location. With the mobility becoming the rule, and not the exception, additional network solutions and mechanisms have been proposed to improve the IP standard, either IPv4 and IPv6, such as Mobile IP (MIP) [2], Network Mobility (NEMO) [3] and NEMO-enabled Proxy Mobile Internet Protocol (N-PMIP) [4,5]. Although these solutions create exciting and practical applications for mobile networks, the handover process is not optimized, especially for VANETs. When a vehicle moves, and consequently changes its PoA, the mobile node needs to send a binding update to the Home Agent (HA) to notify its change of the Care of Address (CoA). Even more optimized solutions that use the concept of the hierarchy of foreign agents

* Corresponding author at: Instituto de Telecomunicações, 3810-193 Aveiro, Portugal.
E-mail address: nmal@av.it.pt (M. Luís).

https://doi.org/10.1016/j.vehcom.2021.100372
2214-2096/

and mobility anchors present an excessive signaling overhead and service delay [6].

More recently, the distinct features of Software-Defined Networking (SDN), such as its flexibility, programmability and network abstraction, have set the stage for a novel networking paradigm termed as Software-Defined Vehicular Networks (SDVNs) [7]. The convergence of SDN with VANETs is seen as an important direction that can address most of the VANET current challenges [8]. In an SDVN data plane entities communicate with the control plane entities for coordinated and efficient communication. The controller provides an up-to-date network view to the application plane that helps it to manage various services (e.g., security, access control, mobility, and QoS) in the network. The RSUs share the collected information about the vehicles and the transportation system to the SDN controller, and execute specific functions that are placed/migrated upon request of vehicles or from other RSUs. Considering the mobility prediction of vehicles, service contents can be migrated to the next visited RSUs to enhance the overall service quality by reducing service latency [9].

In this work we explore the advantages of the SDN concept with respect to the horizontal handover process in a VANET. Following two different SDN architectures, distinguished by the depth of the softwarization environment, a new SDN based solution for seamless[1] handovers in VANET is presented and compared to the most sophisticated and updated existing IP-based solution, the N-PMIPv6 [10]. Performance results obtained with real vehicular equipment and considering two different application services, namely video streaming and file download, show that the proposed solutions, even if not yet capable of multi-homing, present very promising results, specially in terms of the reduction of the average delay and packet loss during the handover process. Moreover, this work shows that SDN solutions are the way forward to decrease the disruption caused by the handover while increasing the flexibility of the handover detection process, minimizing service interruptions during autonomous control and driving. The main contributions of this work can be summarized as follows:

- An assessment of the vehicular network regarding the management of horizontal handovers following the SDN concept;
- Two distinguished SDVN architectures, with different depths regarding the SDN technology, to manage in a seamless way the horizontal handovers caused by the vehicle's movement;
- Deployment, test and evaluation of the proposed SDVN solutions in real vehicular communication equipment (OBUs and RSUs) addressing realistic communication profiles;
- Performance comparison against a non-SDN mobility management solution.

The remaining of this article is organized as follows. Section 2 overviews the related work. Section 3 details the N-PMIPv6 mobility management process. Section 4 explains the proposed architecture for our SDN-based mobility management solution and how handovers are handled. Section 5 overviews the implementation and deployment of the SDN approaches. Section 6 evaluates the SDN solutions and compares them with the N-PMIPv6 approach. Finally, Section 7 presents valuable remarks about this work and introduces future work.

2. Related work

In a network-based approach such as Proxy Mobile IPv6 (PMIPv6), the connectivity suffers from a lengthy handover latency and packet loss during a handover process. The work in [11] proposed a solution based on the Neighbor Discovery used in IPv6 to reduce the latency and packet loss. In [12], the authors proposed a fast handover scheme using IAPP (Inter-Access Point Protocol). Moreover, the work in [13] has shown that in a Mobile IP-based handover, the registration process requires a large number of location updates and excessive signaling overhead, resulting in a significant service delay. The solution proposed in [14] presents inter-technology IP-based handovers in a multihoming capable scenario, *i.e.* when multiple radio access technologies are available. To mitigate service interruptions, the authors in [15] have recently proposed a network-based L2 extension handover scheme for VANETs, reducing the signaling cost and handover latency.

On the other hand, SDVNs have witnessed substantial increments from technical and architectural aspects. With the advances on the 5G network architecture, the SDN has been explored to manage vertical handovers and boost the characteristics of VANETs, namely to increase the radio access coverage and provide complementary communication paths with the Internet.

SDN-enabled 5G-VANET was proposed to resolve the rising traffic conditions by promoting the Heterogeneous Network (HetNet) concept [16]. Neighboring vehicles are put under the same cluster, depending on the real-time road situations using SDNs' global information collection and network control capabilities. Because of the segregation of information plane and control plane, 5G-VANET handles and encourages centralized control over HetNets, by giving a comprehensive network view. The work in [17] evaluated different strategies to balance the SDN control plane through the different communication technologies assuming a trade-off between cost and network control latency. An SDN-enabled social-aware clustering algorithm for 5G-VANET was proposed to empower the adaptive and efficient clustering for communication and information sharing between Base Stations [18]. The core logic is to build a cluster of nodes that follow similar social matches, which can possibly have the same future routes. An improved genetic algorithm has been also proposed to optimize the dynamic network changes in VANETs [19]. It includes the solution for adjusting the dynamic changes often occurring in the network, and also gives the guarantee about the solution diversity. The main idea is to balance the use of V2V and V2I traffic to minimize the latency. Fuzzy analytic hierarchy process and multi-path transmission control protocol are also used to solve the problem of vertical handover in heterogeneous wireless networks based on SDN [20].

Other authors explore the concept of SDN in vehicular environments by making use of Fog Computing to enable more efficient vehicular communications [21], to coordinate the load balancing for inter-domain communication in 5G vehicular networks [22], or even to leverage the concept of Internet of Vehicles with multiple SDN controllers [23].

Regarding the horizontal handover in VANETs, some recent works exploring the SDN technologies have been proposed. The work in [24] proposed a fast handover scheme based on the mobility prediction of vehicles. The authors do not provide any information about the mobility prediction scheme and a single architecture is considered where the RSUs are not included in the SDN plane. Still, two different network domains, with two SDN controllers, are assumed. The performance evaluation, using a network simulator, is too simplistic since the handover delay is the only metric under discussion: there are no statistics about the control overhead, or even the information about the packet loss, delay and jitter for a network service.

Following the same rationale, the work in [25] discusses the advantages of anticipating the handover process aiming to reduce the control signaling overhead and network congestion. The work proposes the use of a Mobility Anchor, as used by the IP-based solutions, and duplicates the packets to increase the probability

[1] The term seamless is used because the handover process is transparent to the end-user, and not because the service has no interruptions, which will depend on the network connectivity.

of packet delivery, which may overload the network unnecessarily, even more in scenarios of high mobility and frequent handovers. Regarding the performance evaluation, and just like the previous work, a network simulator is used and no specific services, with different traffic profiles, are considered.

The work in [26] follows a different approach as it merges both philosophies: the SDN concept with the PMIPv6 architecture. The authors keep both management entities, the Local Mobility Anchor (LMA) of the PIMIPv6 architecture, and the SDN controller of the SDVN, which can be seen as a duplication of services, and introduces one additional step in the handover management increasing the network overhead: the RSUs communicate with the SDN controller, which then communicates with the LMA. Additionally, the LMA in architectures such as PMIPv6 is the point of concentration of all communications between the vehicular domain and the Internet, due to the tunnels created to handle the mobility, which may represent a point of overload and a critical point of failure. Additionally, the performance analysis is limited since the only metric of analysis is the latency associated with the handover process.

Contrary to the above-discussed studies, in this work we focus on the advantages of the SDN management of the horizontal handover when two different SDN architectures are considered - when the RSUs are inside and outside of the SDN environment. The behavior of the SDN controller is highly detailed and both solutions are evaluated using real vehicular hardware - a rare evaluation setup found in the literature - and different and realistic traffic profiles. Furthermore, a large set of evaluation metrics are considered and compared with N-PMIPv6 [10] solution.

3. Base work

This section describes an enhanced version of the N-PMIPv6 mobility approach, to use it as a baseline for comparison purposes with the SDN-based solution. Based on the NEMO-enabled Proxy Mobile IPv6 (N-PMIPv6) [4], it has been added with multihoming support [14,10,27]. The main entities used in this protocol are shortly explained as follows:

- LMA: the Local Mobility Anchor is the Home Agent (HA) of this architecture. It manages the binding states of all mobile nodes, it is responsible for the routing process, and it also stores information about the overall network status to optimize the traffic balancing distribution;
- MAG: the Mobile Access Gateway is responsible for tracking and notifying the LMA about all the mobility-related aspects. The MAGs are RSUs where the end-devices can connect through wireless interfaces, and serve as a gateway to the LMA. They behave like a PoA to OBUs;
- mMAG: a mobile MAG acts as a mobile router in a vehicle, being responsible for providing IP Internet access to the vehicle and its users. The mMAG is an OBU that connects itself and the end-user with the mobility network;
- CN: the Correspondent Node is a peer node with which a Mobile Node (the vehicle and its users) wants to communicate. It represents an entity connected to a global network (such as the Internet). The LMA bridges the CN specific traffic to the mobile network connected to the LMA.

Fig. 1 illustrates the current N-PMIPv6 approach. It considers an inter-technology scenario, where the mMAGs (in the OBUs) can be connected to the infrastructure through the MAGs (RSUs) using IEEE 802.11g/n, IEEE 802.11p/WAVE, LTE, or others. To handle mobility, the traffic is sent in IPv4-IPv6 and IPv6-IPv6 tunnels, introducing significant overhead.

Additionally, the architecture contains a connection manager running in the OBUs. This entity is responsible for the selection of

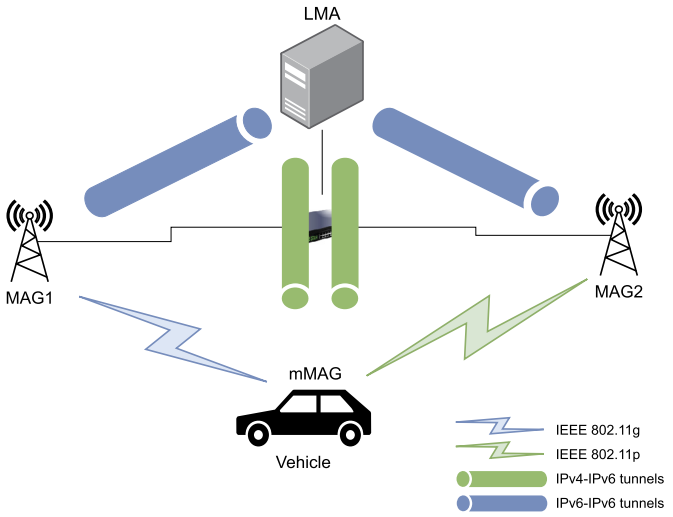

Fig. 1. Current N-PMIPv6 architecture. (For interpretation of the colors in the figure(s), the reader is referred to the web version of this article.)

the best technologies and connections available. It can also make single and multiple connections simultaneously, through multihoming, allowing the VANET to have the best load balancing possible for the traffic that flows through the OBU.

The control messages exchanged in this approach are Router Solicitation (RS) and Router Advertisement (RA) messages between the OBUs and the RSUs, and Proxy Binding Update (PBU) and Proxy Binding Acknowledgment (PBA) messages between the RSUs and the LMA. These messages are customized to fulfill our needs in diverse functionalities and help in the handover process, still complying with the legacy protocol specifications. When the OBU changes its PoA, it sends an RS message with all the needed information to inform the LMA about this handover process. At the RSU, this information from the RS is integrated into the PBU sent to the LMA. After that, the LMA updates its User Cache Entry, starting the handover process at the LMA side. A PBA and a RA message are also sent in the downlink. This process typically happens twice to assure the correct handover in case of loss of some control messages. As the control messages piggyback the needed information to perform the handover, the handover process does not add considerable overhead. On the other hand this process takes significant time as the handover is just completed after both the LMA and the OBU change their routes to use the new PoA after processing the information that is sent through these messages, and the updated tunnels supporting the communication between the OBU and the LMA through the new RSU are established. Moreover, the amount of overhead in the data plane is significant, due to the successive tunnels (required for the mobility process and for the IPv4 services in the vehicles or from its users).

A handover in a VANET happens when an OBU, to which one or more users can be connected, through a connection manager, understands that the RSU (or RSUs in the case of multi-homing) that it is connected to does not have a strong enough signal. Then, the communication path used to communicate with the Internet should be updated through a better RSU, if available. This process should be transparent for the vehicles (OBUs) and the users connected to the OBU, and should ensure that it is a seamless process by providing the minimum latency and packets' loss.

4. SDN-based handovers

Software Defined Networking (SDN) is a network paradigm that aims at splitting the network architecture into control and data planes [28,29]. In the control plane, one or more controllers are

responsible for creating and managing rules that are to be applied to the data/forwarding plane equipment, namely SDN switches.

Communication between both planes is supported by communication protocols such as the OpenFlow protocol [30], maintained by the Open Networking Foundation (ONF).[2] The OpenFlow protocol defines the set of messages between controllers and switches. This set of messages includes the ones responsible for creating and updating a rule in a switch. The rules are programmable and define the actions of the switch, like dropping or forwarding a packet, considering the controller decision through a user defined criteria, information about the network topology, performance metrics, and others.

This means that, unlike the IP based solution, the control plane of an SDN based network can be programmed to detect the occurrence of a handover and then act accordingly. For example, it can be performed by changing routes to a mobile node and updating the information of the location of this node in a data structure for management or monitoring of the network, without the required overhead of a solution like N-PMIPv6. Also, unlike N-PMIPv6, the control plane is decoupled from the traffic forwarding, increasing not only flexibility, but also providing the possibility for future extensions of SDN based solutions.

The proposed SDN based mobility solution enables fast mobility for VANETs through a unified network abstraction. This is performed by creating heterogeneous wireless devices at the RSUs, giving more flexibility to its use, such as generate abstraction, and using them as SDN switches with a unified interface. Instead of having a LMA as in the N-PMIPv6 approach, responsible for the management of all communications, with SDN, one or more controllers are responsible for the handover detection and control.

In this work we propose two SDN architectures whose differences lie on the depth of the softwarized environment. In the first one, RSUs do not present SDN capabilities, while in the second one, the SDN environment is extended to the edge.

4.1. SDN architecture 1 - pure reactive solution

Fig. 2 illustrates the first approach for a mobility-based SDN solution. In this first stage, we consider a centralized control plane with a single SDN switch. This switch is responsible for establishing the connection between the vehicular infrastructure - RSUs, OBUs and end users - and an external gateway that has access to the Internet. The switch allows the processing of packets from and to the OBUs (and its users) by the control plane.

In this architecture, the controller application detects the handover after it has already started: it does that by detecting a change in the source MAC address of the packets coming from a specific OBU. This change of MAC address shows that the packet is coming from a different RSU than the one to which the OBU was previously connected to.

Since the handover detection is based on the analysis of packets, this solution requires data traffic between an OBU and another endpoint, so that the handover can be detected. If no traffic is generated, the handover will not be detected.

For this solution to work, the RSUs need to be connected to at least one SDN switch, which in turn, needs to be connected to one or more SDN controllers. For experimental purposes, one SDN switch is considered, but in a real situation, several switches would be needed to ensure proper load balancing from the traffic of many RSUs in a VANET.

Beyond supporting the handover, there are other benefits of using an SDN switch instead of a regular switch, even without the controller. For example, one advantage is the possibility to allocate and assign the network resources, such as bandwidth, in a dynamic approach.

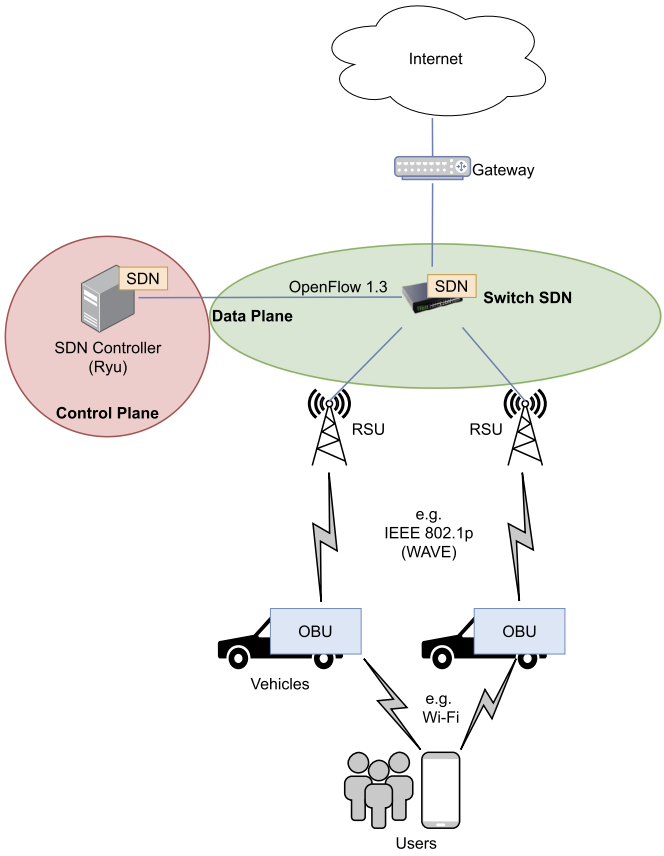

Fig. 2. SDN mobility network - architecture 1.

4.2. SDN architecture 2 - a step towards a proactive solution

A possible improvement to the SDN based architecture is a more proactive detection, where the handover is detected immediately when it is happening, even if there is no traffic. For this purpose, a second architecture is proposed, where the RSUs themselves are part of the SDN topology by also being SDN switches. This way RSUs can communicate to the controller(s) the changes in the topology, including the handover situation.

Such architecture, where the RSUs are part of the SDN topology, has not been fully explored previously. Thus, this work will give preliminary insights on the advantages of having another layer of SDN switching, and if it presents a huge impact in the overall performance of the mobility management system.

Fig. 3 illustrates this approach. Here, the RSUs are part of the SDN data plane and abstracted in SDN switches with an unified interface. Enabling the control plane to manage the network and network resources at the RSUs' level allows for more flexibility for specific V2V applications, not requiring the traffic to flow through the physical SDN switch.

4.3. The controller role

After detecting a handover, the controller needs to modify the flows in the switches as needed to ensure a seamless handover. In the proposed solution, the controller, after detecting a handover, changes the rules in the affected switches, *i.e.* the switch where the OBU was connected and the switch where it is now connected. These changes ensure that any traffic to the OBU is forwarded through the new switch, and not the old one. Section 5.1 details

[2] https://opennetworking.org/.

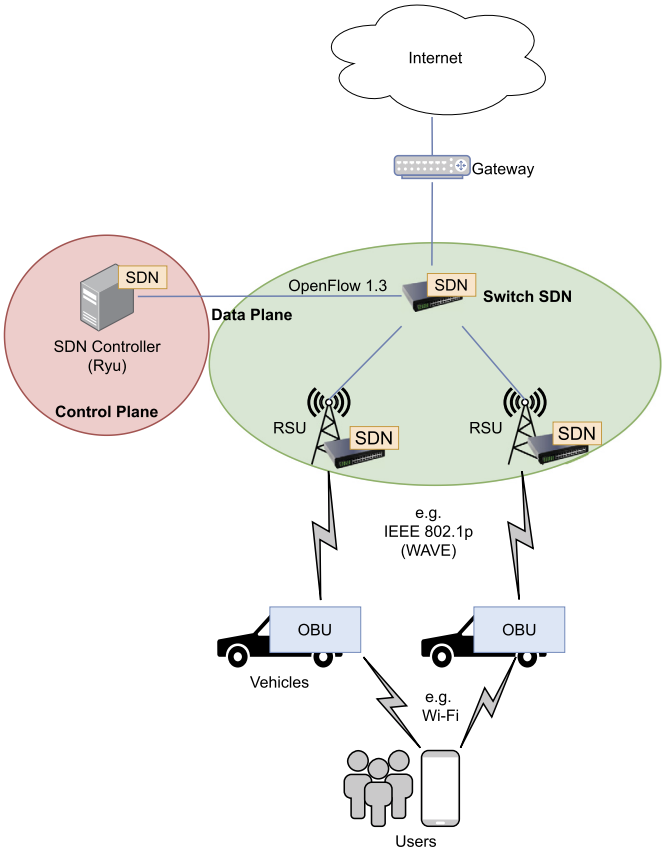

Fig. 3. SDN mobility network - architecture 2.

the mechanisms implemented in the controller application so it can properly work in such mobility approaches.

5. Implementation and deployment

The first step towards the implementation and deployment of both architectures started with building the base topology. This base topology follows the layout presented in Figs. 2 and 3, and includes several main components, namely: OBUs, RSUs, end-users, a gateway, SDN switch(es) and, of course, an SDN controller.

The deployment of the topology considers several steps: (1) the configuration of the nodes that make up the vehicle network, that is, the configuration of network addresses, default routes, network forwarding as well as Network Address Translation (NAT) (more specifically, the Source NAT) on each one of the OBUs, so that a variety of end-users can be represented as the single OBU in the network; (2) the configuration of the SDN switch (or switches depending on the topology), which in this case means the configuration of an OpenFlow switch software [31]; and finally, (3) the configuration of the SDN controller to enable the control communication with the SDN switch(es).

Once the topology is deployed and all components are correctly interconnected and configured, it is possible to start the development of the SDN controller application.

5.1. SDN controller application

The SDN controller application is the main component of the solution and is responsible for managing the flow control on each SDN switch. In our approach, the SDN controller application can be divided into several main processing steps, as illustrated in Fig. 4, which will be discussed next.

5.1.1. Handling unknown packets

The main method in the controller application is the packet handler, the entry point of the packets sent by the SDN switch to the SDN controller. The packet handler is invoked asynchronously, every time a packet arrives at the SDN switch and it does not have a matching flow. When a packet is received, it is checked the pertinent information of this packet (e.g. addresses, protocols). After the collection of this information, the packet gets processed based on its protocol (this will be explained in further detail in the sections below). In this specific controller application, only the ARP and IP packets (including specific IP protocols) are processed; any other packets are simply not processed and therefore dropped.

Once the original packet has been processed, it reaches its final stage in the application, where two options exist: the original packet is dropped, *i.e.* it is not needed anymore (e.g. ARP Request which the controller replied to), or the original packet is simply sent out to its original destination.

This last option is also divided in two, depending on whether the application knows how to reach the destination of the packet. If the application does not know where to send the packet, it floods the packet in the network; otherwise, it sends it to its destination and, in that case, it also adds the required flows on the SDN switch so that the next packet of that type can simply go through the switch to its destination. This whole process repeats itself anytime a new packet arrives at the SDN controller.

5.1.2. Handling ARP packets

When processing ARP packets, our application only checks each ARP packet to see if it is an ARP Request for any registered OBU. If that is not the case, then this processing step is over and the packet is simply sent out as explained in section 5.1.1. When the ARP packet is indeed an ARP Request for a registered OBU, the application processes it further.

Anytime there is an ARP Request for a registered OBU, since there is never a flow that will match that packet on the SDN switch, the controller will always get and process that packet. When it receives such a packet, it is responsible for checking if it is indeed an ARP Request for a registered OBU; if that is the case, then it creates and sends an ARP Reply to the requesting node. In this reply, the sender's MAC address will be the RSU's MAC address to which the OBU is connected at the moment, instead of the OBU's MAC address, so that any traffic sent from the requesting node to the OBU will now be sent via the RSU to which it is currently connected.

5.1.3. Handling IP packets

When the controller application receives any IP packet, the first step it does is to check if this packet came from any OBU; this is done by checking if the source IP address of the packet is an address that belongs to the sub network that enables the communication between the RSUs and OBUs (this is possible due to the fact that this information as well as information about all the RSUs is pre-registered in the controller application). If this packet is originated from an OBU, the controller registers the OBU using the required information, then it is also possible to detect which RSU the OBU is connected to. For that, the controller application checks if the MAC source address of the packet belongs to any of the known RSUs; this way, it is possible to create a logical link in the application between the OBU and the RSU which it is currently connected to. Once the application knows about this logical link, this process is no longer carried out, and only the existence of the link itself is verified. This information will be important when detecting if a handover has happened.

The last two main processing steps of the IP packets happen on two specific situations, which are: checking for a handover and checking for packets sent via a wrong RSU. These situations are ex-

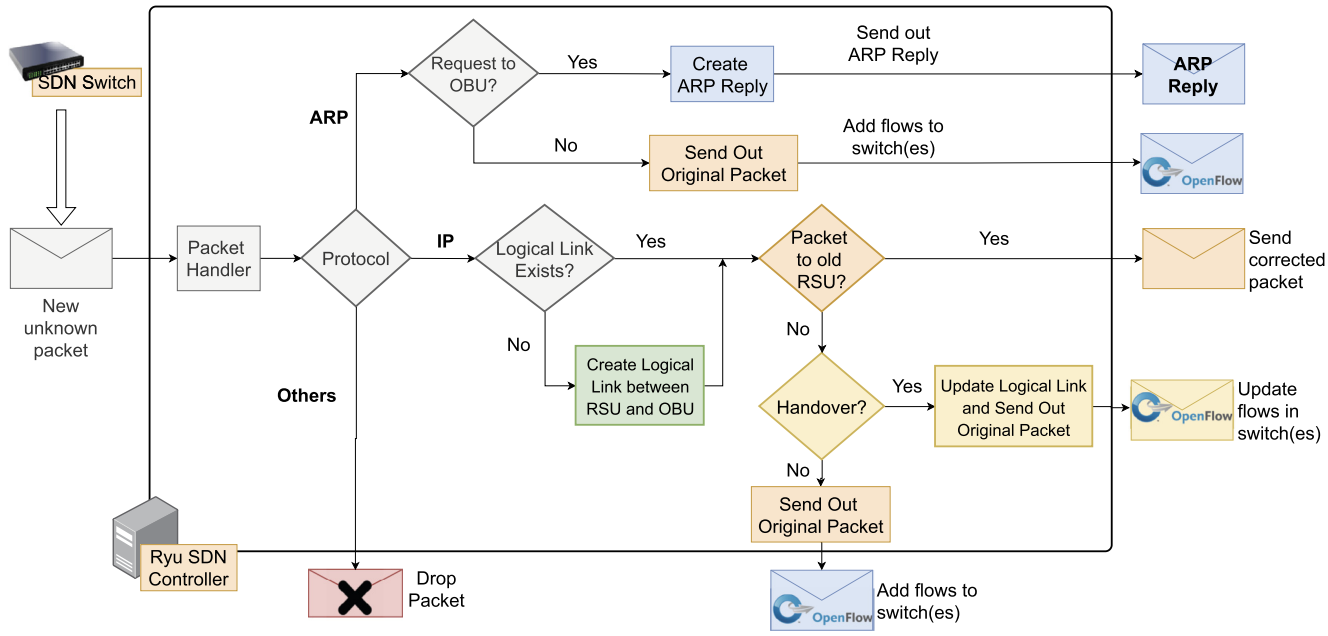

Fig. 4. Simplified SDN controller application architecture.

plained next, starting by the handover detection process, and then by the handover process itself when it is detected.

5.1.4. Handover detection

The handover detection in the SDN architecture 1 is straightforward, as mentioned in 4.1. This process begins by first checking if the packet came from a registered OBU (via its source address): if that is the case, then the source MAC address of the packet is extracted and compared against the MAC address of the RSU to which the OBU is currently connected to. If the MAC addresses are not the same, a handover has occurred, since the packet arrived from a different RSU; on the other hand, if they are the same, no handover has occurred and the handover checking process is over. Once a handover is detected, the logical link is updated, meaning that the application updates the RSU to which the OBU is now connected to, and removes any flows on the SDN switch matching the OBU and the old RSU. Once this step is complete, the handover is now finished.

In the architecture 2, the handover detection can take advantage of the fact that the RSUs are part of the SDN plane, which allows the controller to know about changes in the topology of the VANET at the RSU level, such as the disconnection or connection of an OBU. In a simulated software environment, such as Mininet, the handover can be detected when it happens, since the port change in a RSU means that a OBU was connected and disconnected. However, in a real situation, this is not viable in a hardware scenario, where wireless communications are used.

This means that architecture 2 is not proactive, but a step towards a proactive solution. Using a set of custom messages - for example extending the OpenFlow symmetric experiment messages that allow the creation of custom messages, or the usage of CAMs (Cooperative Awareness Messages defined by ETSI for ITS awareness services) - would allow RSUs - part of the SDN environment of the network only in architecture 2 - to inform the controller about changes in the OBUs, therefore achieving a proactive solution.

5.1.5. Rerouting packets

The last main processing step is to check for packets that were sent to an OBU via a wrong RSU (*i.e.* RSU to which an OBU was previously connected to) and re-route them. This problem can happen anytime a handover occurs while a packet from an OBU is outside the vehicular network; when that packet returns, the gateway might still not know that a handover has occurred, meaning that it can send the packet via the wrong RSU. Thus, this process was introduced to mitigate this problem, and consequently, reduce packet loss.

This process begins by checking if the packet is for a registered OBU (via its destination address); if this is the case, then the packet's destination MAC address is extracted and compared against the MAC address of the RSU to which the destination OBU is currently connected to. If the MAC addresses are the same, this means that the packet was sent via the correct RSU and the processing is over, and therefore, the packet can move on to the next processing step. On the other hand, if the MAC addresses are different, we are in the presence of a packet sent via a wrong RSU. When that happens, a copy of the original packet is created, the destination MAC address is changed to the MAC address of the correct RSU, and finally the packet is sent out to the destination. Before finishing this processing step, a gratuitous ARP packet is also created on behalf of the OBU and sent to the network, in order to update the ARP mappings (mainly of the gateway) and avoid this problem in the future.

Once the packet has been through all the IP processing stages, it moves on to the final processing stage mentioned previously in 5.1.1.

6. Evaluation setup

This section presents the setup and tested scenarios carried out to evaluate the performance of both SDN architectures against the N-PMIPv6 solution in a vehicular network environment.

6.1. Evaluation scenario

In order to test the performance of both solutions, IP and SDN based, a custom topology, as shown in Fig. 5, is designed and deployed. This topology, implemented in a laboratory, using real vehicular equipment (OBUs and RSUs), allowed to emulate two real life scenarios.

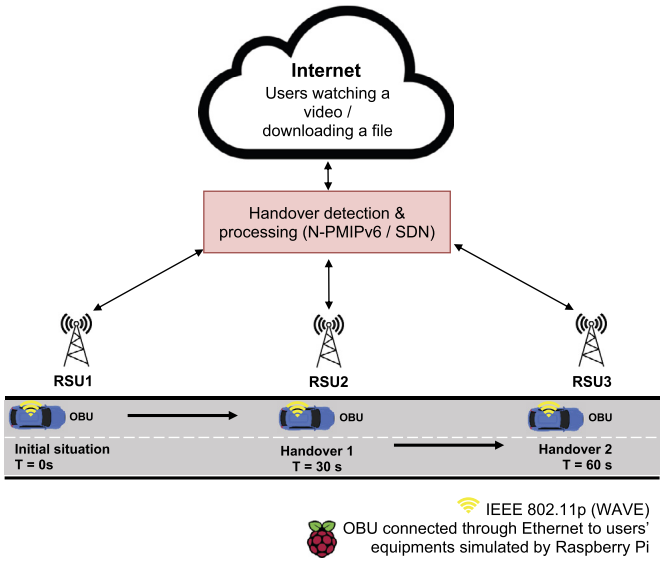

Fig. 5. Illustration of the topology used in the test scenarios. In this topology, a vehicle - represented by an OBU - passes through several RSUs, and the handover detection and processing systems, based on N-PMIPv6 or SDN, ensure that mobile users can keep using the Internet, for example to watch a video or downloading a file.

The emulated real life scenario consists of a vehicle equipped with the OBU (in this instance only the OBU is present given the laboratory topology) passing by three distinct RSUs. Each time the vehicle moves closer to a new RSU, it changes the RSU with which it is communicating, thus performing a handover. In this work we assume disjoint RSUs radio coverage, which means that it is not possible for one OBU to be in the range of communication of two RSUs simultaneously. As the vehicle moves, the end users which are connected to the vehicle's OBU might be streaming a video or downloading a file.

Since the topology emulates a real life scenario in a laboratory setup, the handover itself was forced by developing a script to control the behavior of the connection manager, as described in Section 6.2. The equipment used in the experiments is later detailed in Section 6.3.

6.1.1. Scenario 1: video streaming

In this scenario, to emulate a video stream reaching the vehicle through its OBU, the end user equipment will make a connection to a server running outside the network and start receiving data according to a set of specific parameters. To simulate Variable Bit Rate (VBR) video traffic, the following set of parameters were considered [32]:

- Constant inter-departure time of 24 packets/s;
- Packet size which follows normal distribution of mean 27791 bytes and standard deviation of 6254 bytes.

6.1.2. Scenario 2: file transfer

To emulate a scenario where a file is downloaded to the vehicle, the end user equipment will also make a connection to a server outside of the network and start the download according to a set of specific parameters:

- Constant inter-departure time of 700 packets/s;
- Packet size of a constant size of 1000 bytes.

Table 1
Evaluated metrics.

Metric	Evaluation process
Average delay	Measured using D-ITG[a]
Jitter	Measured using D-ITG
Time per handover	By comparing the packet timestamps traveled in the old and new paths
Overhead	N-PMIPv6: Number and size of RS and RA messages[b] SDN: Number and size of OpenFlow messages
Packet loss	Measured using D-ITG

[a] http://traffic.comics.unina.it/software/ITG/.
[b] Router solicitation and router advertisement messages.

6.2. Evaluation methodology

To understand the performance of both solutions, a set of performance metrics are considered. Having in mind the great differences between the previous IP based mobility and the SDN mobility, these indicators are as general as possible, while also allowing the direct comparison of both approaches. Table 1 lists the considered metrics and how they were evaluated in N-PMIPv6 and SDN approaches.

A set of tests were executed for both scenarios and network architectures. Real network hardware is used during the evaluation process, but the mobility profile is emulated in the laboratory environment. To be more precise, a script was developed to control the behavior of the connection manager, *i.e.* we decided the exact moment for the handover to occur. Although such behavior does not represent the uncertainty of a real vehicular environment, it is a fair mechanism in order to guarantee the same conditions among all network architectures and respective tests.

For each scenario, several tests are conducted where different numbers of handovers are performed (0, 1 or 2), for a total test duration of 90 seconds. Each test is executed 10 times, using both UDP and TCP traffic with the help of the D-ITG software [33], and between a server located outside the network and the end user equipment, connected to the OBU. The final results represent the average values and their 95% confidence interval.

It is important to note that the tests for the video streaming scenario using TCP are considered as important as the ones in UDP, to understand the overall performance of the developed approaches, even if most video streaming solutions rely heavily on UDP or other protocols. The same applies for the file transfer scenario when using UDP.

6.3. Equipment

To run all the required tests, two topologies, like the one illustrated in Fig. 5, are deployed: one to test the SDN architectures, and another one to test the N-PMIPv6 solution.

In terms of equipment, for the N-PMIPv6 deployment, the LMA is deployed on a dedicated machine running Ubuntu as its Operating System (OS), whereas the RSUs and OBUs are implemented in Single Board Computers (SBCs) denoted as NetRiders running a custom OS,[3] and with both IEEE 802.11/WiFi and IEEE 802.11p/WAVE (V2V technology).

As for both SDN architectures, the SDN controller is deployed on a dedicated machine running Ubuntu as its OS, and for the SDN framework, the controller application is running on Ryu.[4] This framework was chosen given the fact that it is an open-source SDN controller that supports the OpenFlow protocol and, since it is based on Python, it allows for quick and effective development,

[3] http://veniam.com.
[4] http://ryu-sdn.org/.

Table 2
Equipment specifications.

Equipment	CPU [MHz]	Memory [MB]	Linux Kernel	OS
LMA	1400 (2 cores)	2048	4.14.3	Ubuntu 16
NetRiders	680	64	3.7.4	VeniamOS 19.2
SDN controller	3600 (2 cores)	4096	4.15.0	Ubuntu 18.04
APU	1000 (4 cores)	4096	5.7.10	Debian 8
RPi 3 Model B	1200 (4 cores)	1024	4.4.38-v7+	Ubuntu 16.04.6

Table 3
Control packet overhead [number of packets].

	Number of handovers		
	0	1	2
Video streaming emulation - UDP			
N-PMIP	72 ± 1	79 ± 1	89 ± 1
SDN Arch 1	18 ± 1	97 ± 4	155 ± 15
SDN Arch 2	69 ± 4	225 ± 13	268 ± 12
Video streaming emulation - TCP			
N-PMIP	71 ± 1	80 ± 1	91 ± 1
SDN Arch 1	17 ± 1	65 ± 10	96 ± 18
SDN Arch 2	62 ± 3	150 ± 14	194 ± 19
File transfer emulation - UDP			
N-PMIP	72 ± 1	78 ± 2	90 ± 1
SDN Arch 1	13 ± 1	36 ± 3	60 ± 8
SDN Arch 2	67 ± 3	126 ± 5	162 ± 4
File transfer emulation - TCP			
N-PMIP	73 ± 1	81 ± 2	90 ± 2
SDN Arch 1	16 ± 1	35 ± 2	71 ± 8
SDN Arch 2	58 ± 1	109 ± 7	167 ± 11

while also providing good performance for small network topologies [34]. As for the RSUs and OBUs, these are implemented in APU platforms which are SBCs produced by PC Engines,[5] containing both IEEE 802.11/WiFi and IEEE 802.11p/WAVE interfaces.

Finally, when it comes to the end user equipment, these are simulated using RPis connected directly to the OBUs through their Ethernet interfaces. Table 2 presents the specifications of the equipment used.

7. Results

This section presents and discusses the most important results for both scenarios and for all the network architectures. First we will analyze the results related to the network characteristics, such as packet overhead, packet loss and time per handover. Then, we will extend the discussion by analyzing the impact of the handovers on each of the two scenarios, the video streaming and file transfer, when it comes to the average delay time and jitter.

7.1. Packet overhead

Table 3 presents the overhead in terms of the number of control packets observed during the tests on all the mobility solutions when performing, zero, one or two handovers. Similarly, Table 4 presents results about the network overhead, but in this case it is represented in terms of size (in kbytes). The control packet overhead should be seen as an indicator to understand the complexity and cost of each mobility solution.

When comparing the results between the SDN approaches and the baseline N-PMIPv6, it is not possible to get a clear conclusion on which is the best when it comes to the overhead. The SDN architecture 1 has the least amount of overhead in almost every scenario with the exception of UDP video streaming. On the other hand, the SDN architecture 2 has the highest overhead values out of all solutions (mainly in the cases of one or more handovers). Finally, the N-PMIPv6 solution is better in terms of overhead than the SDN architecture 2 in the cases where handovers occurred, but loses out to SDN architecture 1 on almost all scenarios.

Table 4
Control packet overhead [kbytes].

	Number of handovers		
	0	1	2
Video streaming emulation - UDP			
N-PMIP	13.0 ± 0.93	13.3 ± 0.19	15.0 ± 0.20
SDN Arch 1	1.9 ± 0.09	20.3 ± 0.97	30.5 ± 3.08
SDN Arch 2	12.3 ± 0.97	41.1 ± 2.58	48.6 ± 2.79
Video streaming emulation - TCP			
N-PMIP	12.8 ± 0.88	13.5 ± 0.16	15.3 ± 0.31
SDN Arch 1	1.97 ± 0.14	12.04 ± 2.02	17.9 ± 2.8
SDN Arch 2	6.4 ± 0.34	26.4 ± 2.84	33.6 ± 3.81
File transfer emulation - UDP			
N-PMIP	12.9 ± 0.85	13.1 ± 0.27	15.2 ± 0.37
SDN Arch 1	1.3 ± 0.11	10.4 ± 1.44	21.7 ± 3.22
SDN Arch 2	9.5 ± 1.13	33.2 ± 1.79	44.5 ± 1.56
File transfer emulation - TCP			
N-PMIP	13.1 ± 0.88	13.2 ± 0.33	15.1 ± 0.44
SDN Arch 1	2 ± 0.28	6.2 ± 0.36	13.3 ± 1.63
SDN Arch 2	6.3 ± 0.29	17.3 ± 1.49	27.6 ± 2.41

Results show a clear difference between the SDN architectures, with the second architecture, where the RSUs are part of the SDN network, being the solution with larger overhead, but also with a larger variability. This is explained by the fact that, since in the second SDN architecture, the RSUs are also SDN switches, the overhead - counted as the number of OpenFlow packets - increases substantially. This happens because the OpenFlow messages are now sent to and from three SDN switches (one on each RSU), thus increasing the control overhead to around three times when compared with architecture 1.

In addition, some of the OpenFlow messages that translate into the additional overhead are messages sent by the switches and are not processed fast enough by the controller, especially in specific test scenarios like UDP video streaming: when a handover occurs, there a lot of packets sent by the SDN switch to the controller in a short period of time.

The fact that the SDN controller is centralized and without redundancy, means that it might be a source of such processing delays. This is not a big problem when the SDN controller is only responsible for one SDN switch, but this proved to increase the control overhead when more SDN switches are present. While both SDN architectures are feasible in terms of the costs that they mean to the network, when it comes to packet overhead, special caution is required when scaling the number of SDN switches.

Possible solutions to mitigate this problem would be adding more SDN controllers in a distributed way, or migrating the controller application to a more professional one. Considering the SDN switches, these could also be upgraded to real SDN switches as opposed to the software SDN switches used in both SDN architectures.

Another disadvantage of the N-PMIPv6 that should be taken into consideration, and not visible in the presented results of control overhead, is that this solution, because of the tunnels it requires (as described in Section 3), also introduces a significant data overhead, non-existent in the SDN solutions. This data overhead results from the existence of the IPv4-to-IPv6 tunnel between the LMA and the OBUs, and the IPv6-to-IPv6 tunnel between the LMA and the RSUs. Such mechanisms result in the addition of two IPv6

[5] http://pcengines.ch.

Table 5
Time per handover [ms].

Video streaming emulation	
N-PMIPv6	8307.6 ± 57.15
SDN Arch 1	12.9 ± 0.32
SDN Arch 2	13.4 ± 0.30
File transfer emulation	
N-PMIPv6	6804.9 ± 94.08
SDN Arch 1	9.4 ± 0.48
SDN Arch 2	8.2 ± 0.58

Table 6
Packet loss [%].

	No Handover	Per Handover
Video streaming emulation [UDP]		
N-PMIPv6	0.002 ± 0.00	6.673 ± 0.66
SDN Arch 1	0	0
SDN Arch 2	0	0.0 ± 0.04
File transfer emulation [UDP]		
N-PMIPv6	0	7.298 ± 0.51
SDN Arch 1	0	0.0 ± 0.01
SDN Arch 2	0	0
Both scenarios [TCP]		
All	0	0

headers over the original IPv4 data packet, increasing its size and data overhead by 80 bytes in total per data packet transmitted.

7.2. Handover time

Table 5 presents the handover time, *i.e.* the time for the handover process to be completed, which is a great indicator of the complexity of the handover process and its impact in the user experience - the increase in the handover time also means an increase in the average delay of the packets, and potentially, a worse user experience.

Results show consistently that the SDN architectures have handover times within tens of milliseconds, while the N-PMIPv6 shows consistent results within 6-8 seconds. This can be explained by the large complexity of the handover process in the N-PMIPv6 solution, specially the deletion and creation of tunnels as described in Section 3 whenever the OBU connects itself to a new RSU. On the other hand, the simpler handover process in the SDN architectures - which is the result of the fast packet processing and flow changes by the SDN controller application, as explained in Section 5.1.4 - enables fast and efficient handovers, which results in significantly lower times when compared with the N-PMIPv6 solution.

7.3. Packet loss

Table 6 presents the packet loss percentage of the different solutions. This metric is evaluated considering two distinct situations: measuring the packets lost in a period without handovers, and when handovers are observed. An important conclusion can immediately be taken from the results: the TCP based tests showed no packet loss. This was the theoretically expected result since TCP recovers from losses of packets, at the cost of greater complexity when compared with UDP. In addition, it confirms that both SDN architectures work without packet losses in TCP scenarios.

In the UDP tests, some packet loss was to be expected in situations where handovers occurred, given the fact that communication is suspended for a brief period of time while the handover is processed. In this case the results show the clear advantage of the SDN architectures in terms of robustness. The SDN approaches

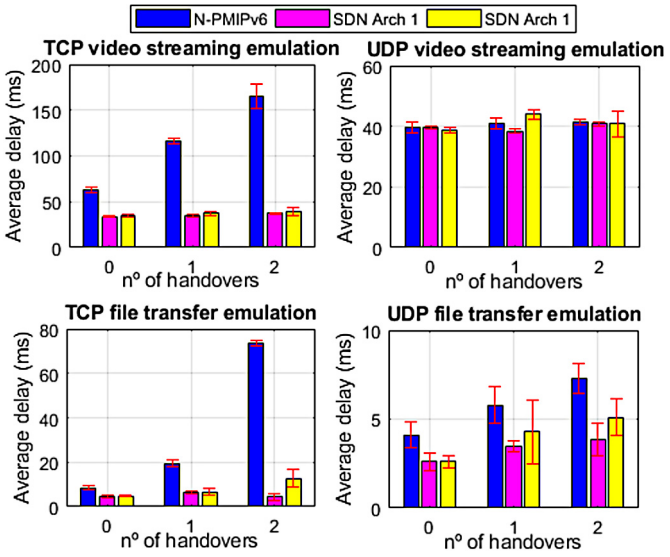

Fig. 6. Average delay time.

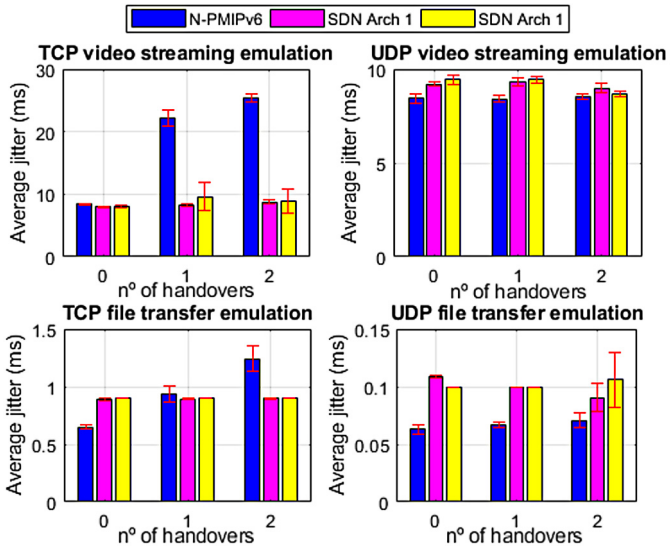

Fig. 7. Average jitter.

have only marginal packet losses (0.01% or less), while the N-PMIPv6 has losses of around 6-8% per handover. This is a clear consequence of the time it takes to perform a handover, where the longer the handover takes to complete, the more packets are dropped. As seen in Table 5, the N-PMIPv6 solution has a significantly higher time per handover, which justifies its higher packet loss values.

7.4. Average delay time and jitter

Figs. 6 and 7 present the average delay and the jitter of the data packets, respectively, for both services and network solutions.

The average delay results show that the N-PMIPv6 solution presents significantly worse values in almost all tested scenarios when compared with the SDN solutions, with the exception of the video streaming scenario in UDP, where the results are similar between all solutions. This means that, in most scenarios, the N-PMIPv6 solution has a greater impact on the time it takes for packets to get to the end user, which in turn means a worse overall experience for the user. As for the SDN solutions, they present

very similar values in all of the tested scenarios, with a slight difference between the two in the file transfer scenario in UDP.

In the TCP scenarios, it is possible to observe the impact that the handover time has on the average delay. It is clear that the N-PMIPv6 solution presents significantly higher average delay values when compared with both SDN solutions. This can be explained by the way TCP itself works. Since TCP recovers from losses, anytime a packet is dropped (in this case due to the handovers), it has to be re-transmitted, which increases the packet delay. This means that, the longer the handover process takes, the longer it takes for packets to be re-transmitted, which in turn means an increase to the average packet delay. This explains why the values are higher for the N-PMIPv6 solution, as it takes significantly longer time to perform a handover when compared with the SDN solutions. Another important aspect to remark is that the number of handovers seems to affect the average delay in the N-PMIPv6 solution: the more handovers are performed, the higher is the delay. This fact is not observed in the SDN solutions.

In both UDP scenarios, it is possible to observe that the handover time has no impact on the average packet delay for the video streaming scenario. In this scenario, the N-PMIPv6 solution does not suffer from the higher handover time when compared with the SDN solutions, regarding the average delays.

Another important conclusion that can be drawn from the results is that both SDN architectures present, for all the scenarios, around the same average delay. This shows that the fact that RSUs are or not a part of the SDN topology has little to no influence in this metric.

The average jitter, *i.e.* the variation in the time between data packets received, the first evident conclusion is that the N-PMIPv6 presents significantly worse values in TCP video streaming emulation. However, and unlike in the average delay, the N-PMIPv6 presents slightly better results in the video streaming emulation in UDP, and for download scenarios both in TCP and UDP. This can be a consequence of the single centralized controller architecture. As described in Section 7.1, this means that the controller might be a source of processing delays, therefore introducing some variability to the time it takes for packets to be processed by the SDN controller.

8. Conclusions

This article explored the use of the SDN concept to optimize the handover times in vehicular communications. In this article, two SDN based approaches were proposed, with different levels of SDN support, in the core or in the edge. The support of SDN in the edge enables the communication of the topology changes from the RSUs to the controller, providing the support towards a more proactive handover.

These approaches were compared to an existent N-PMIPv6 solution. The results of SDN based approaches show that they are able to provide much lower handover times (around 10 ms), average delays, more robustness thanks to the lower packet loss, and higher flexibility thanks to the programmability of the SDN controller.

Future improvements concern the evolution towards a fully proactive solution, inter network communication and eventual scalability issues as a result of the addition of more SDN-capable entities or fast changing VANETs. The proactive operation mode can be provided through the usage of information from the connection manager of an OBU, or through the usage of CAMs ITS messages to inform the controller about changes in the OBUs, and providing the information for predictive handovers. Other topics of future research include the integration of other technologies, such as cellular, including C-V2X, and the extension to provide and enable multihoming. Such integration leads to new challenges such as the inter domain handovers, which may involve multi-SDN controller environments where the communication between the SDN controllers will be critical for the seamless handover operation. Hypothetical scalability issues could be solved by instructing the SDN controller to proactively notify all the SDN switches involved in a new flow after the first notification arrives at the controller. Finally, understanding how SDVN environments behave when there is no connectivity between the OBU and the infrastructure is also a topic of future research.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

This work is funded by FCT/MCTES through national funds under the project MH-SDVanet (PTDC/EEI-COM/5284/2020).

References

[1] S. Al-Sultan, M.M. Al-Doori, A.H. Al-Bayatti, H. Zedan, A comprehensive survey on vehicular Ad Hoc network, J. Netw. Comput. Appl. 37 (2014) 380–392.
[2] C. Perkins, IP Mobility Support for IPv4, Revised, RFC 5944, RFC Editor, https://tools.ietf.org/html/rfc5944, November 2010.
[3] V. Devarapalli, R. Wakikawa, A. Petrescu, P. Thubert, Network Mobility (NEMO) Basic Support Protocol, RFC 3963, RFC Editor, https://tools.ietf.org/html/rfc3963, January 2005.
[4] I. Soto, C.J. Bernardos, M. Calderon, A. Banchs, A. Azcorra, Nemo-enabled localized mobility support for internet access in automotive scenarios, IEEE Commun. Mag. 47 (5) (2009) 152–159, https://doi.org/10.1109/MCOM.2009.4939291.
[5] N. Capela, S. Sargento, An intelligent and optimized multihoming approach in real and heterogeneous environments, Wirel. Netw. 21 (6) (2015) 1935–1955, https://doi.org/10.1007/s11276-015-0896-1.
[6] A. Gladisch, R. Daher, D. Tavangarian, Survey on mobility and multihoming in Future Internet, Wirel. Pers. Commun. 74 (2014) 45–81, https://doi.org/10.1007/s11277-012-0898-6.
[7] Z. He, J. Cao, X. Liu, SDVN: enabling rapid network innovation for heterogeneous vehicular communication, IEEE Netw. 30 (4) (2016) 10–15, https://doi.org/10.1109/MNET.2016.7513858.
[8] I. Yaqoob, I. Ahmad, E. Ahmed, A. Gani, M. Imran, N. Guizani, Overcoming the key challenges to establishing vehicular communication: is SDN the answer?, IEEE Commun. Mag. 55 (7) (2017) 128–134.
[9] J. Santa, J. Ortiz, P.J. Fernandez, M. Luis, C. Gomes, J. Oliveira, D. Gomes, R. Sanchez-Iborra, S. Sargento, A.F. Skarmeta, MIGRATE: mobile device virtualisation through state transfer, IEEE Access 8 (2020) 25848–25862.
[10] F. Castro, A. Martins, N. Capela, S. Sargento, Multihoming for uplink communications in vehicular networks, in: 2017 Wireless Days, 2017, pp. 230–237.
[11] J. Kang, D. Kum, Y. Li, Y. Cho, Seamless handover scheme for proxy mobile IPv6, in: 2008 IEEE International Conference on Wireless and Mobile Computing, Networking and Communications, 2008, pp. 410–414.
[12] J. Lee, J. Park, Fast handover for proxy mobile IPv6 based on 802.11 networks, in: 2008 10th International Conference on Advanced Communication Technology, vol. 2, 2008, pp. 1051–1054.
[13] S. Saha, S. Bhattacharjee, R.N. Bhowmick, A.K. Mukhupadhyay, D. Nagamalai, Analysis of hierarchical mobile IP based fast mobility management schemes, in: 2009 First International Conference on Networks Communications, 2009, pp. 338–343.
[14] D. Lopes, S. Sargento, Network mobility for vehicular networks, in: 2014 IEEE Symposium on Computers and Communications, ISCC, 2014, pp. 1–7.
[15] J. Choi, Y. Han, S. Min, A network-based seamless handover scheme for VANETs, IEEE Access 6 (2018) 56311–56322.
[16] X. Duan, Y. Liu, X. Wang, SDN enabled 5G-VANET: adaptive vehicle clustering and beamformed transmission for aggregated traffic, IEEE Commun. Mag. 55 (7) (2017) 120–127.
[17] H. Li, M. Dong, K. Ota, Control plane optimization in software-defined vehicular Ad Hoc networks, IEEE Trans. Veh. Technol. 65 (10) (2016) 7895–7904.
[18] W. Qi, Q. Song, X. Wang, L. Guo, Z. Ning, SDN-enabled social-aware clustering in 5G-VANET systems, IEEE Access 6 (2018) 28213–28224.
[19] C.-C. Lin, H.-H. Chin, W.-B. Chen, Balancing latency and cost in software-defined vehicular networks using genetic algorithm, J. Netw. Comput. Appl. 116 (2018) 35–41.

[20] H. Tong, X. Liu, C. Yin, A FAHP and MPTCP based seamless handover method in heterogeneous SDN wireless networks, in: 2019 11th International Conference on Wireless Communications and Signal Processing, WCSP, 2019, pp. 1–6.
[21] J.C. Nobre, A.M. de Souza, D. Rosário, C. Both, L.A. Villas, E. Cerqueira, T. Braun, M. Gerla, Vehicular software-defined networking and fog computing: integration and design principles, Ad Hoc Netw. 82 (2019) 172–181.
[22] Z. Latif, K. Sharif, F. Li, M.M. Karim, S. Biswas, M. Shahzad, S.P. Mohanty, DOLPHIN: dynamically optimized and load balanced PatH for INter-domain SDN communication, IEEE Trans. Netw. Serv. Manag. (2020) 1, https://doi.org/10.1109/TNSM.2020.3045725.
[23] T. Yuan, W.d.R. Neto, C.E. Rothenberg, K. Obraczka, C. Barakat, T. Turletti, Dynamic controller assignment in software defined internet of vehicles through multi-agent deep reinforcement learning, IEEE Trans. Netw. Serv. Manag. (2020) 1, https://doi.org/10.1109/TNSM.2020.3047765.
[24] X. Yin, L. Wang, A Fast Handover Scheme for SDN Based Vehicular Network, Mobile Ad-hoc and Sensor Networks, vol. 747, Springer, Singapore, 2018, pp. 293–302.
[25] N. Mouawad, R. Naja, S. Tohme, Fast and seamless handover in software defined vehicular networks, in: 2019 Eleventh International Conference on Ubiquitous and Future Networks, ICUFN, 2019, pp. 484–489.
[26] S.M. Raza, D.S. Kim, H. Choo, Leveraging PMIPv6 with SDN, in: Proceedings of the 8th International Conference on Ubiquitous Information Management and Communication, ICUIMC '14, Association for Computing Machinery, New York, NY, USA, 2014.
[27] R. Lopes, M. Luís, S. Sargento, Real-time video transmission in multihomed vehicular networks, in: 2019 IEEE Vehicular Networking Conference, VNC, 2019, pp. 1–4.
[28] W. Xia, Y. Wen, C.H. Foh, D. Niyato, H. Xie, A survey on software-defined networking, IEEE Commun. Surv. Tutor. 17 (1) (2014) 27–51.
[29] D. Kreutz, F.M.V. Ramos, P.E. Veríssimo, C.E. Rothenberg, S. Azodolmolky, S. Uhlig, Software-defined networking: a comprehensive survey, Proc. IEEE 103 (1) (2015) 14–76, https://doi.org/10.1109/JPROC.2014.2371999.
[30] W. Braun, M. Menth, Software-defined networking using OpenFlow: protocols, applications and architectural design choices, Future Internet 6 (2) (2014) 302–336.
[31] E.L. Fernandes, E. Rojas, J. Alvarez-Horcajo, Z.L. Kis, D. Sanvito, N. Bonelli, C. Cascone, C.E. Rothenberg, The road to BOFUSS: the basic OpenFlow userspace software switch, J. Netw. Comput. Appl. (2020) 102685.
[32] M.W. Garrett, W. Willinger, Analysis, modeling and generation of self-similar VBR video traffic, in: Proceedings of the Conference on Communications Architectures, Protocols and Applications, SIGCOMM '94, Association for Computing Machinery, New York, NY, USA, 1994, pp. 269–280.
[33] A. Botta, A. Dainotti, A. Pescapè, A tool for the generation of realistic network workload for emerging networking scenarios, Comput. Netw. 56 (15) (2012) 3531–3547.
[34] O. Salman, I.H. Elhajj, A. Kayssi, A. Chehab, SDN controllers: a comparative study, in: 2016 18th Mediterranean Electrotechnical Conference, MELECON, 2016, pp. 1–6.