IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 8 May 2024, accepted 25 June 2024, date of publication 1 July 2024, date of current version 15 July 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3421338

RESEARCH ARTICLE

A Software Defined Vehicular Network Using Cooperative Intelligent Transport System Messages

DUARTE DIAS[1], MIGUEL LUÍS[1,2], PEDRO RITO[1], (Member, IEEE), AND SUSANA SARGENTO[1,3], (Member, IEEE)

[1]Instituto de Telecomunicações, 3810-193 Aveiro, Portugal
[2]Instituto Superior Técnico, Universidade de Lisboa, 1049-001 Lisboa, Portugal
[3]Departamento de Eletrónica, Telecomunicações e Informática, Universidade de Aveiro, 3810-193 Aveiro, Portugal

Corresponding author: Miguel Luís (nmal@av.it.pt)

This work was supported in part by the European Union/Next Generation EU, through Programa de Recuperação e Resiliência (PRR) [Project Nr. 29: Route 25], in part by the European Regional Development Fund (FEDER), through the Competitiveness and Internationalization Operational Programme (COMPETE 2020) of the Portugal 2020 framework [Project IMMINENCE with Nr. 112314 (POCI-01-0247-FEDER-112314)] and by the Fundação para Ciência e a Tecnologia (FCT)/Ministério da Educação, Ciência e Inovação (MECI) through national funds under the project Multihomed Software Defined Vehicular Networks (MH-SDVanet) (PTDC/EEI-COM/5284/2020).

ABSTRACT The increasing need for smarter means of transportation has brought a greater focus on Vehicular Ad-Hoc Networks (VANETs). VANETs are characterized by frequent topology changes due to the high node mobility, and most of the existing network protocols for mobility management are not well prepared for persistent horizontal handovers. This work studies the integration of Software-Defined Networking (SDN) in the Intelligent Transportation Systems (ITS) universe for the mobility management of VANETs. The idea is to explore the Cooperative Intelligent Transport Systems (C-ITS) messages in the SDN universe allowing the SDN controller to have a deeper view of the topology state. This global view gives the controller proactiveness, anticipating handovers even before they happen, reducing handover delays, and thus, improving the user experience. To evaluate the proposed approach, two types of environments were considered, laboratory and real-world at city scale with connected vehicles and infrastructure. The performance results have shown great improvements compared with a non-proactive approach: exploring the benefits of integrating the ITS and SDN universes, we obtained an almost zero packet loss rate, a reduction in the average delay and an increase in the connectivity time.

INDEX TERMS Vehicular ad-hoc networks, software-defined networks, intelligent transport systems, horizontal handovers, performance evaluation.

I. INTRODUCTION

Smart and connected mobility can be used to improve safety, automation or even pollution consumption in the roads, building new Intelligent Transportation Systems (ITS). Vehicular Ad-Hoc Networks (VANETs), a key part of the ITS framework, are mobile networks that allow vehicles to communicate with each other as well as with roadside base stations, called Road-Side Units (RSUs), located at critical points of the road. VANETs are characterized by frequent topology changes due to the high node mobility, with a node being a vehicle equipped with an On-Board Unit (OBU). In this type of networks, horizontal handovers are much more frequent than in traditional Wireless Local Area Networks (WLANs), mainly because of the velocity associated with the mobile nodes.

To make this type of networks a reality, new IP network protocols for mobility management emerged. Solutions based on extensions of Mobile IP (MIP) [1], Network Mobility (NEMO) [2], Proxy Mobile IPv6 (PMIPv6) [3] and NEMO-enabled Proxy Mobile IPv6 (N-PMIPv6) [4] were introduced to solve the frequent topology changes where each

The associate editor coordinating the review of this manuscript and approving it for publication was Binit Lukose.

mobile node changes its Point-of-Attachment (PoA) very often. Although they were promising solutions, they suffered from a lack of flexibility and abstraction level. For these reasons, these solutions had an excess of signalling overhead and service delay. More recently, with the appearance of SDN, solutions applied to the vehicular environment have emerged, thus creating the Software-Defined Vehicular Network (SDVN) paradigm [5], [6].

The advantages of SDN, such as its flexibility and programmability, made it possible to create solutions with lower complexity and, consequently, lower overhead and less delay in services. However, existing solutions exploring the use of SDN in VANETs were designed to operate only when there is IP traffic upload, *i.e.* they require IP traffic originated from the OBUs side for handovers to be detected by the controller (if there is no IP traffic uplink, the network does not know which OBUs are reachable from each RSU) [7]. These solutions, denoted as reactive approaches for the remaining of this work, do not oversee the state of the network and cannot predict imminent network changes.

Our proposed approach brings the idea of merging vehicular information with SDN control, taking advantage of the existing vehicular messages in the ITS universe, called C-ITS messages. These messages, categorized into several types, allow vehicles, infrastructure, and other road users to exchange information in real-time, enabling applications like collision warnings, traffic management, and navigation assistance. Specifically, Cooperative Awareness Messages (CAMs) are of great interest here. CAMs are exchanged between ITS Stations (ITS-Ss), RSUs, and OBUs to establish and maintain awareness, sharing multiple data like IDs and GPS coordinates [8]. Using these messages together with SDN, it is possible to offer a continuous notion of both the location of the OBUs and their connectivity with neighboring RSUs, thus making it possible to predict the next RSU that a given OBU will connect to. The proposed solution allows to run multiple types of services such as entertainment, traffic management, or road safety, since the network knows how to reach the vehicles regardless of whether the OBUs have initiated prior communication, thus being crucial for the solution to have an accurate and up-to-date notion of the network's topology [9].

The main contributions of this article are the following:

- A network management solution integrating the use of C-ITS messages in an SDN environment;
- A proactive SDVN solution capable of managing flows and initiating downlink L3 communication with a specific OBU even if there is no L3 uplink traffic;
- An exhaustive analysis of horizontal handovers in a proactive SDN enabled vehicular network;
- Performance comparison between the proposed SDN architecture and an existing one (reactive);
- Evaluation tests using real vehicle communication equipment (OBUs and RSUs) in both laboratory and city environments, with connected vehicles and infrastructure.

The remaining of the article is organized as follows. Section II discusses the related work. Section III details the proposed solution design. Section IV details the proactive operating mode of the proposed approach. Section V evaluates the performance of the solution by comparing it with a previous state of the art reactive SDN solution. Finally, Section VI enumerates the conclusions and introduces directions for the future.

II. RELATED WORK

In the last decade, VANETs have received significant attention from the academic and industrial communities. Due to the volatility of the wireless medium, VANETs face several challenges. The concept of Software-Defined Networking (SDN) applied to vehicular networks emerged as a possible solution to mitigate these challenges [10].

In the realm of SDVN, numerous research works have emerged focused on several key areas, such as Security/Privacy and Quality-of-Service/Routing Management. For instance, [11] exemplifies advancements in these areas by introducing a hybrid SDN-VANET architecture. This architecture prioritizes privacy and security while concurrently ensuring good performance in terms of message loss rate, packet delivery rate and delay reduction. Notably, the study integrates the dynamic positioning management of SDN controllers within the network based on road traffic fluctuations. Other research works, such as [12] and [13], delve deeper into the realm of data dissemination and QoS, with their solutions based on network metrics like throughput during solution evaluation.

Recent advances on the 5G network architecture enabled SDN to manage vertical handovers and boost the functionalities of VANETs. This allowed for an increase in the radio access coverage and provided complementary communication paths with the Internet. The work in [14] proposed an optimization strategy to balance the SDN control plane through different communication technologies (5G cellular and ad-hoc). This strategy balances the low latency of 5G cellular with the low cost of ad-hoc networks to obtain an intelligent solution. Experimental results have shown that this strategy can provide smaller latency than other control plane structures. However, these results were obtained using a network simulator, not knowing the feasibility of the solution in a real environment.

Regarding the horizontal handover in VANETs, some recent works exploring SDN technologies have been proposed. The work in [15] proposed a fast handover scheme based on the mobility prediction of vehicles. In this work, a single architecture is considered, and RSUs are not part of the SDN domain. In this solution, two different network domains, with two SDN controllers, are considered. The SDN controllers of the scheme predict movement of vehicles by detecting port status of SDN switches, proceeding to the addition of flows proactively. The performance evaluation is accomplished using a network simulator, and only one metric is discussed which is the handover delay.

The work in [16] follows a different approach, since it combines SDN with the network-based mobility management standard, PMIPv6. In this work, both management entities are present, the LMA of the PMIPv6 architecture, and the SDN controller of the SDVN. This can be seen as a duplication of services and introduces one additional step in the handover management which increases the network overhead. Additionally, the performance analysis is very limited, since only one metric is considered, which is the latency associated with the handover process.

In [17] the authors propose an advanced handover process based on SDN to manage data transmissions in VANETs. The solution consists of a set of SDN enabled vehicles divided into clusters. Each cluster has a cluster head (CH) that relays information from other vehicles via LTE. The handover process is managed by the controller that monitors the vehicles movement and decides when each vehicle needs to handover. Experimental results showed that the proposed approach significantly improves the network performance in scenarios with frequent topology changes.

The work in [7] considers two different SDN architectures that vary in depth in the SDN domain, evaluating the impact of the depth of the softwarization environment. In the first one, designated as reactive, the RSUs are not part of the SDN domain, whereas, in the second one, designated as proactive, they are. Results showed that the lower complexity of the SDN solution allows for a better performance during handovers. The work is quite complete, having realistic traffic profiles and a large set of metrics. Another positive point is that performance evaluation is accomplished using real vehicle equipment and emulated mobility scenarios. However, both solutions are dependent on prior uplink traffic for the solutions to work, which limits the applicability of the solution.

The use of C-ITS messages in a vehicular domain has been standard for a long time. These messages allow applications such as active road safety and traffic efficiency. However, the combination of these messages with an SDN domain is something that has been less explored.

The work in [18] proposes an architecture with a distributed control plane that employs a hierarchy of controllers that can dynamically adjust to environment and network conditions. RSUs are part of the SDN domain as they also contain controllers. The goal of the solution is to enable C-ITS messages dissemination in a dynamically distributed network. This solution combines the use of C-ITS messages with the SDN domain; however, the messages are not aimed to help on the management of the vehicular network.

The work in [19] proposes a hierarchical SDN-based vehicular architecture that aims to improve performance in situations of loss of connection with the central SDN controller. This solution explores the use of periodic beacon messages (C-ITS messages) to let the controller know where the mobile nodes are. This information is used to create routes between different SDN domains when a node needs to transmit information. The solution exploits the use of C-ITS messages in an SDN domain, however, its use does not allow the creation of proactive flows, that is, prior to the occurrence of traffic. Additionally, the solution is only tested on a network simulator.

The work here presented will focus on the use of C-ITS messages in the SDN management process of horizontal handovers. In the proposed architecture, RSUs are inside the SDN domain, and the solution is evaluated using real vehicular hardware in both laboratory and city environment, which is a rare evaluation setup found in the literature. Realistic traffic profiles will be considered for the testing. Furthermore, a large set of evaluation metrics will be considered and the proposed proactive solution will be compared to a reactive SDVN solution.

III. PROACTIVE SDVN SOLUTION DESIGN

The new proposed approach joins the SDN concept with the use of C-ITS messages to improve the proactiveness of vehicular handovers. The base architecture, depicted in Figure 1, comprises a control plane with a single controller, and a data plane that includes a main SDN switch and RSUs that are also SDN switches, being also part of the SDN topology. The main SDN switch is responsible for interconnecting the vehicle infrastructure (RSUs, OBUs and end-users) and the gateway that connects to the Internet. The switches communicate with the controller using the OpenFlow communication protocol [20] to create the flow table, which dictates what should happen to packets arriving at the switches. The connection between RSUs and OBUs is established using ITS-G5 communication technology, a standard used in vehicular environments where handover processes are recurrent.

A. ITS MESSAGES IN THE SDN DOMAIN

To offer proactiveness to the proposed approach, we consider the use of control ITS messages that are disseminated over the network in broadcast between ITS-Stations (ITS-S). In the case of the proposed architecture, these ITS-Ss are RSUs and OBUs. For this solution, the most suitable class of messages are CAMs because they carry signalling information such as location, heading, speed and type of vehicle. This information is relevant to the controller because it allows it to have an idea of the movement of the OBUs, thus allowing the prediction of handovers even before they happen.

In this solution, the creation of flows related with the handover occurs in a proactive manner, that is, the logical handover is prepared even before the physical handover occurs. This is possible because of the C-ITS messages (CAMs) that, together with RSSI (between RSUs and OBUs) observed in the transmission of such messages, make it possible to characterize the OBUs' movement and predict the next positions, and consequently, the time when handovers will occur. The choice of the RSU to which the OBU should connect will be discussed in greater detail in Section V.

Regarding the security aspect of this solution, any malicious modification on CAMs will impact the performance of

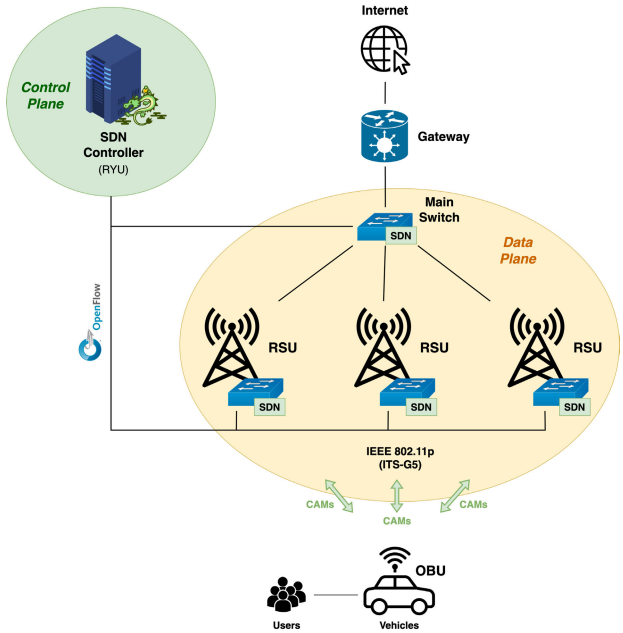


FIGURE 1. Single main switch SDVN architecture.

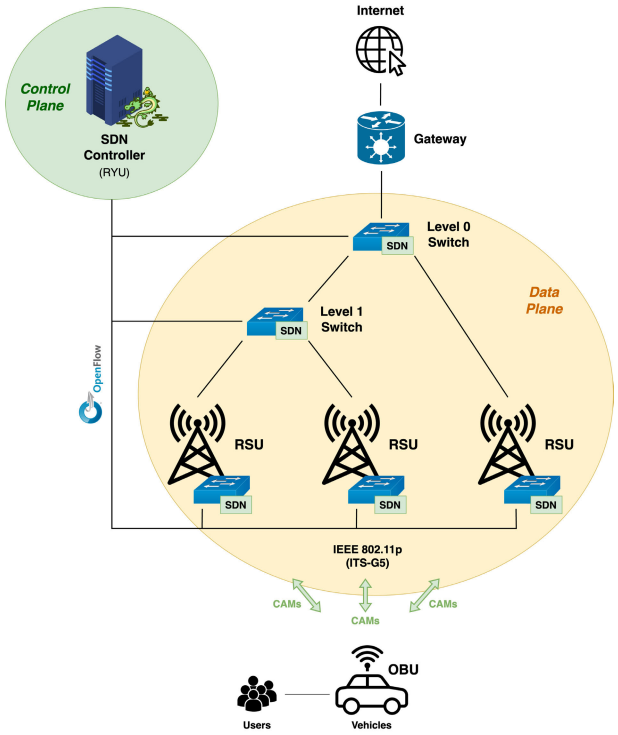


FIGURE 2. *Multiple switch levels* SDVN architecture (two level configuration).

the management solution. For example, if the location and speed fields in a CAM are modified, the SDN controller will consider the vehicle as being in a different place, and the communication flows would be installed in the wrong RSU. This means that the proactiveness of the proposed solution would be affected, and the solution would operate as a reactive SDN solution.

B. SINGLE SWITCH *AND* MULTIPLE SWITCH LEVEL *ARCHITECTURE*

Two approaches representing different complexities of the vehicular SDN infrastructure are considered: the *single switch level* includes a main SDN switch (only one level) and RSUs that are also SDN switches, being also part of the SDN topology (depicted in Figure 1); the *multiple switch levels* includes a hierarchy of SDN switches with multiple levels (depicted in Figure 2).

In the *multiple switch levels* architecture, the top level switch, in this case "level 0" switch, makes the interconnection between the gateway that gives access to the Internet, the sub-level switch and one RSU. In the lowest level, "level 1", the switch is responsible for interconnecting the remaining vehicle infrastructure (RSUs, OBUs and end-users) with the level above it, "level 0". This sub-solution allows both RSUs and switches to be linked to levels that are not necessarily the level just above them. In this solution the switches also communicate with the controller using the OpenFlow communication protocol. This approach aims to distribute the traffic across different switch levels to reduce the probability of a bottleneck, which can happen when there is a higher number of RSUs with only one switch connecting them to the rest of the network.

C. SDN CONTROLLER DECISION

An SDN controller is an application in the SDN architecture that manages the various flows of an SDN switch. Usually, the controller application runs on a server and uses switch management protocols such as OpenFlow [20] or OVSBD [21] to communicate with the switches, giving information on the rules to the packets.

In this architecture, the controller application proactively detects the handover by the CAMs information and RSSI of the OBUs. Based on this information that reaches the controller, this one starts to be aware of several parameters referring to OBUs, such as location, speed, heading, altitude and signal strength (RSSI) towards other RSUs, which will allow the controller to decide which is the optimal RSU for a given OBU at a particular moment. If the current RSU of a given OBU remains optimal, there is no change; otherwise, the controller proceeds to change, taking all necessary actions, including changing the flows on the switches.

It is important to mention that, although the solution does not need IP traffic for handovers to be detected, it is necessary for the controller to know which IP network the RSUs communicate with the OBUs. The controller is able to know the OBU ID through the CAMs information, and this ID corresponds to the 4th byte of the IPv4 address.

D. ROAD SIDE UNIT (RSU) INTERFACES

In this solution, the RSUs are part of the SDN topology, these being SDN switches. Here, one of the RSU's wired

ports are part of the Open vSwitch (OVS),[1] and all traffic that arrives at the RSU via the wireless interface is redirected to a bridge, also in the OVS domain. This bridge makes the interconnection between the wireless interface and the OVS ports. Figure 3 illustrates the RSU interfaces.

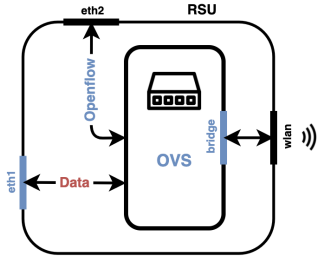


FIGURE 3. RSU Interfaces.

The RSU, being an ITS-S, is constantly transmitting C-ITS CAM messages. The sending frequency depends on the type of station but the RSU, being a fixed station, sends with a frequency of 1 Hz. These messages are broadcast on the network so that other stations and vehicles can receive them if in range.

E. ON BOARD UNIT (OBU) DECISIONS

Unlike RSUs, OBUs do not belong to the SDN domain, and the controller does not affect their behaviour. Therefore, decisions regarding the gateway change are made internally by the OBU's connection manager, which is based on CAM messages coming from the various RSUs spread across the city. Like the RSU, the OBU is also constantly broadcasting CAMs. This information is shared both with other vehicles on the road as well as with the infrastructure so that they can make intelligent decisions.

IV. PROACTIVE SDVN OPERATION AND ALGORITHMS

The solution outlined here consists of two primary components: an SDN controller and the OBU's connection manager. These components implement a shared algorithm responsible for managing horizontal handovers on both the infrastructure and vehicle sides.

The following sections will first explore the algorithm behind the management of horizontal handovers, which is integrated into both modules. Subsequently, a detailed examination of the SDN controller and the OBU connection manager will be provided, covering their respective specifics, algorithms, and challenges.

A. MANAGING HORIZONTAL HANDOVERS

A very important part of the solution is the decision logic, which deals with the decision of which RSU a given OBU should connect to at a given instant of time. This decision logic is shared by both decision makers, the SDN controller and the OBU's connection manager, as a method that executes within its own thread, which is depicted in Algorithm 1.

[1]https://www.openvswitch.org

Algorithm 1 Selecting the RSU to Be the Point-of-Attachment

```
1  Function SelectingRSU(OBUs):
       Input: list of OBUs
       Output: ∅
2      while True do
           /* Applicable to Controller */
3          for each OBU do
4              if connected to RSU then
5                  if RSSI exists then
                       /* Search for RSU that matches OBU's heading */
6                      for r in listOfRSUs do
7                          if OBU heading matches r then
                               /* Comparing r's RSSI with current RSU's RSSI */
8                              if r_RSSI > act_RSSI then
9                                  change current RSU to r
                                   /* Applicable to Controller */
10                                 change flows in switches
                                   /* Applicable to CM */
11                                 change default gateway
12             else
13                 if RSUs in reach then
                       /* Search for RSU that matches OBU's heading */
14                     for r in listOfRSUs do
15                         if OBU heading matches r then
16                             OBU associated with r
                               /* Applicable to Controller */
17                             add flows to switches
                               /* Applicable to CM */
18                             set default gateway
19         wait 500ms
```

This method has slight variations depending on whether it is being executed in the SDN controller or the OBU's

connection manager (simply denoted hereafter as CM). In the case of the controller, the method performs a loop cycle that iterates over the various OBUs that are registered in the controller. For each of the OBUs, the decision is made to find out which RSU is most suitable to be the OBU's default gateway proceeding with the addition of flows in the RSU. In the case of the CM, it consists of an infinite loop that also checks which is the best RSU to be the OBU's default gateway, changing only the default IP gateway.

Regarding the method's common core, this one is divided into two parts, the first one represents the case where the OBU is not currently associated with any RSU, therefore there is no logical link between the two. The second part deals with the case where there is already a logical link.

Concerning the case where there is still no RSU associated with the OBU (line 12), an analysis is made to the list of RSUs that are in reach of the OBU (line 13). For each of the RSUs present in the list, a check is made to see if the location of the RSU matches with the OBU's heading (line 15). There is a match if the RSU is within a certain angle, as shown in Figure 4. In case of a match, this RSU becomes associated with the OBU, thus creating a logical link between the two (line 16). If there is no match between the two, it proceeds to the next RSU on the list. If the list of RSUs is empty or the RSUs present do not satisfy the condition mentioned above, then the process for this OBU is over, thus proceeding to the next OBU.[2]

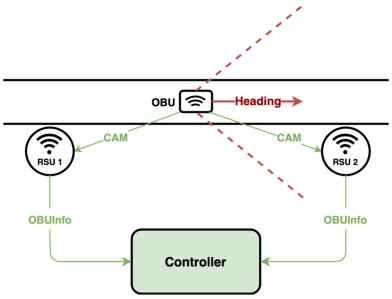


FIGURE 4. OBU heading matching with RSU.

Assuming that there is already a logical link between an OBU and an RSU (line 4), a check is performed to assess if there is an RSSI associated with the linked RSU (line 5). In the positive case, an iteration will be made on the list of RSUs that are in reach of the OBU (line 6). For each RSU, if the location matches with the OBU's heading (line 7), then it proceeds to the next phase, otherwise it proceeds to the next RSU. The next phase consists on the comparison between RSSI of the current RSU with the next RSU's RSSI that satisfied the precondition (line 8). If the RSU's RSSI is greater than the current RSU's RSSI including a certain threshold, which has a value of 2, then a handover will occur (line 9); otherwise, it will continue with the next RSU. When the handover occurs, there is the destruction of the existing logical link between the OBU and the old RSU, and then a new link is created between the OBU and the new RSU.[3]

The two methods that have been described before address the common trunk regarding the selection of the RSU as a Point-of-Attachment. However, there is still another method that is common to both components, the timeout block, which is responsible for defining the maximum validity of the content of a CAM message. This method, like the ones discussed above, runs on a dedicated thread and is independent of the others. This method aims to update the list of RSUs in reach, associated with the OBU, keeping it up to date. This is achieved by eliminating RSUs where the last CAM received is longer than 3 seconds. In the case where the list is empty and the OBU is still connected, if it exceeds 10 seconds, the association between RSU and the OBU, called logical link, is destroyed. This way, there will be no disturbance in the decision logic caused by outdated information. This is a very important method for the proper operation of both components.

The timeout value for the CAM validity (3 secs) was defined based on the element (SDN controller or OBU's CM) with the lowest arrival frequency of CAMs. Since the OBU's CM receives CAMs every 1 second (ideally), it makes no sense to opt for a period shorter than that. Since there are obstacles/interference in the way, it is natural that the period between the arrival of CAMs increases. The chosen value takes into account these factors and was tested in a city environment to prove the feasibility of such value.

Although the decision logic of both elements (controller and OBU's CM) is the same, it does not necessarily mean that both elements behave similarly at all times. The fact that the messages that reach both elements are not the same means that the decisions taken may not be the same. The controller receives messages from the OBUs with a (theoretical) frequency of 10 Hz, while the OBU's connection manager receives messages from the RSUs with a frequency of 1 Hz. Thus, decisions may not be synchronous between both elements; however, the controller, which has the most frequently updated information, will update the flows on the switches even before the connection manager makes the decision. Therefore, the uplink may not be synchronous with the downlink, but there will always be connectivity.

B. CONTROLLER APPLICATION

The controller application is the main component of the proposed approach, since it is responsible for controlling the flows of the several SDN switches in the architecture. It is based on the RYU SDN framework,[4] which is an

[2]It is important to highlight that this heuristic, based on the RSSI and heading, is one possible approach for the selection of the RSU as Point-of-Attachment. Our goal is not to discuss the best heuristic to be used in the handover process. The main contribution of this work lies on the management of the vehicular network through ITS messages but, to showcase the potential of such solution, a heuristic for this process must be used. This was the one selected, which can be further replaced by many other strategies.

[3]The threshold value used in this work (2) was selected based on tests carried out in a city environment, whose setup is shown in Section V.

[4]https://ryu-sdn.org

Open-Source Software (OSS) system that supports the OpenFlow protocol. The proposed solution can be divided into two major parts. These two run in parallel and are central to the correct operation of the architecture. The first part is responsible for the packet handling, and the second is responsible for the logical association of a given OBU to an RSU.

1) PACKET HANDLING

The packet handler is responsible for handling all packets that arrive at the controller from the SDN switches. This method is executed asynchronously whenever an SDN switch does not have a flow that matches the given packet, encapsulating the packet and sending it to the controller. Algorithm 2 describes this process.

Algorithm 2 Packet-In Handler Algorithm

```
1  Function PacketHandler(packet):
       Input: packet to be processed
       Output: ∅
       /* Extract packet protocol      */
2      protocol ← packetProtocol
3      if protocol is OBUInfo then
4          if packet from registered OBU then
5              update OBU info
6              drop packet
7      else if protocol is ARP then
8          if request to OBU then
9              create ARP reply
10             drop packet
11         else
12             send out original packet
13             add flow to switch
14     else if protocol is IP then
           /* Extract packet IP protocol
              */
15         ipProtocol ← packetIpProtocol
16         if ipProtocol in [ICMP, UDP, TCP] then
17             send out original packet
18             add flow to switch
19         else
               /* All other IP protocols
                  are not processed      */
20             drop packet
21     else
           /* All other protocols are not
              processed                  */
22         drop packet
```

When receiving a packet, the relevant information, such as the source and destination addresses, as well as the protocols, are extracted (line 2). From the collection of this information, processing is done, initially, based on the packet protocol. The solution can process the following protocols: ARP, IP and a custom one called OBUInfo discussed in the next paragraph. All other protocols are not processed at this stage and are thus dropped (line 22).

The OBUInfo protocol is a custom protocol whose ethertype number is 0xBBBB (which is not assigned by the IEEE 802 standard). The protocol was developed to be able to differentiate between custom OBUInfo messages (containing the contents of CAM messages plus the RSSI), representing control traffic, from IP packets that represent data traffic. Therefore, the controller can separate both, so there is no need to send control messages within data traffic.

Regarding the supported protocols, each one of them will be explained in greater detail. Whenever the controller receives a OBUInfo packet (line 3), the packet's payload is extracted to obtain the relevant information. Then, a verification occurs to check if the latter came from a registered OBU. If so, the extracted information is used to update the object referring to the OBU present in the controller (line 5). The extracted information includes the OBU ID, GPS coordinates (longitude and latitude), heading, speed, RSSI (between OBU and RSU that received the CAM) and the timestamp. The RSSI is associated with a given RSU, and internally, there is a list of RSUs with its associated RSSI, which represents the RSUs that are within reach of the OBU. In the end, the OBUInfo packet is dropped because it has already fulfilled its purpose and is not reinserted into the network (line 6). If the packet does not come from a registered OBU, it is simply dropped.

When an ARP packet is received (line 7), a verification occurs to see if it is an ARP Request for any of the OBUs that are registered with the controller. If so, the controller creates and sends an ARP Reply to the node that created the message (line 9). The message that will be created will have as source MAC address the RSU to which the OBU is currently connected. This way, the messages coming from the requesting node with the OBU as destination will necessarily be sent via the RSU connected to the OBU. In the end, the ARP Request packet is dropped because it has already fulfilled its purpose and is not reinserted into the network again (line 10). In cases where it is an ARP Reply packet or even an ARP Request with an unregistered OBU as destination, the packet is reinserted into the network and its associated flows are added to the switch (lines 12 and 13), thus ending its processing.

Finally, in the case where the controller receives an IP packet (line 14), the IP protocol of the packet is extracted. If the IP protocol is one of the following ICMP, UDP or TCP, then the packet is reinserted into the network and its associated flows are added to the switch (lines 17 and 18). Otherwise, the packet is dropped (line 20).

2) OBU ASSOCIATION

Regarding the association of an OBU to an RSU, it is done based on the decision logic discussed in the Section IV-A.

Taking such logic into account, a choice is made with which RSU the OBU will create a logical link. The association process will diverge into two parts depending on whether there is an association beforehand.

The first part addresses the case where a first connection is being established for both *single switch* and *multiple switch levels* architectures. As far as the *single switch level* solution is concerned, after creating the logical link between the OBU and the RSU, flows are added proactively to the SDN switches (RSUs and main switches). Figure 5 illustrates the flows on the various switches. When adding flows, firstly, uplink and downlink flows are added to the RSU associated with the OBU, in this case RSU 1. These flows are not reactive, they are very generalist and are only based on pre-known data and only on the IP protocol, making no distinction between the various types of higher layer protocols. The output ports are also pre-known since the controller knows where the OBU is connected, being aware of the topology. In the case of the downlink, this is only based on the RSU's MAC and the OBU's IP, and in the case of the uplink it is even more general based only on the entry port. Secondly, we have the addition of the downlink flow at the main switch level. As for the RSU, the controller holds information about the OBU and the port to get there, which is also only based on the IP protocol and has no higher layer protocol. The uplink flow that is missing at the main switch level is only added when IP traffic occurs (reactive) because there is no pre-known information to be able to create it proactively, at least at the first time. However, during the handover process (changing the RSU to which the OBU is connected), which will be discussed next, this flow will be added proactively. At the end of the addition of the flows, the list of RSUs that are in reach of the OBU is reset so that there is no old RSSI values for RSUs that are no longer within the OBU's reach.

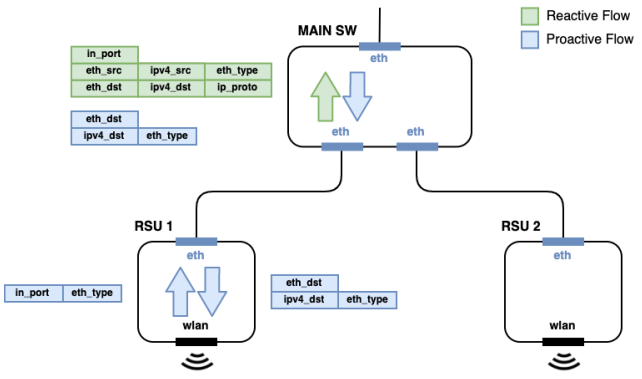


FIGURE 5. Flows related to first connection in the *single switch level* architecture.

Figure 6 illustrates the flows for a *multiple switch levels* configuration with two switch levels and three RSUs, two RSUs in level 1 (RSUs 1 and 2) and one in level 0 (RSU 3). Here, all the switch levels (levels 0 and 1, in this case) behave the same way, so they follow the same reasoning as the main switch in the previous version. Being independent switches, the ports referring to the flows can differ between switches. The controller, with knowledge of the topology, identifies the ports associated with each of the RSUs, enabling proactive addition of flows. Thus, downlink flows are proactively added as there is pre-known information to create them. Uplink flows, on the other hand, are only added when there is IP traffic. In this scenario, as RSU 2 initiates traffic, flows are added to both RSU 2 itself and the level 1 and level 2 switches. However, if the traffic had been initiated by RSU 3, flows would only be added to the RSU 3 and the level 0 switch.

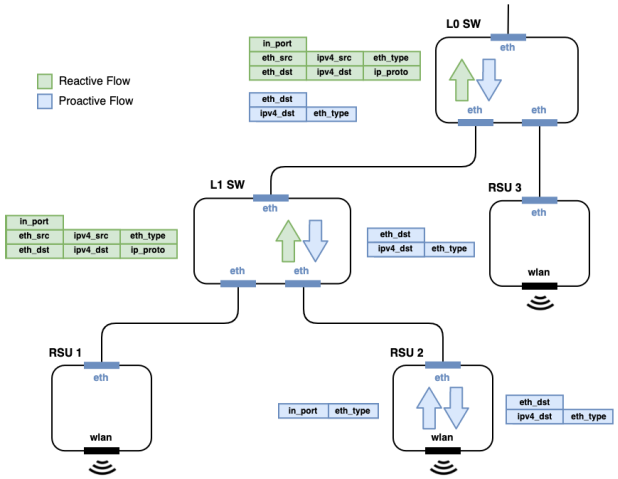


FIGURE 6. Flows related to first connection in the *multiple switch levels* architecture.

The second part addresses the case where there is already a previous association beforehand for both *single switch* and *multiple switch levels* architectures. In the context of the single switch architecture, this scenario depicts an initial connection to RSU 1 followed by a handover to RSU 2, as depicted in Figure 7. In this case, downlink flows are also generated in the SDN switches, mirroring the process of the initial connection. However, in this case, a few extra steps are needed. Initially, as flows are being added, a "gratuitous ARP" carrying the OBU IP is sent to inform the sender that the MAC has been changed, so that it starts sending new packets through the new RSU (RSU 2). Then, the previously mentioned reactive flow (Main SW) is replicated and established on the main switch, with only the input port and the MAC source corresponding to the new RSU 2. Lastly, the outdated downlink flows are removed.

In the context of multiple switch levels architecture, the scenario depicts an initial connection to RSU 2 (level 1 switch) followed by a handover to RSU 3 (level 0 switch), as depicted in Figure 8. In this setup, the top-level switch (level 0) takes charge of sending the "gratuitous ARP" containing the OBU IP, signaling the sender about the MAC change. Subsequently, on the same switch, the existing reactive flow is duplicated, with modifications limited to the MAC and input ports, thereby creating a new flow. Concerning the lower-level switches (level 1), the previously

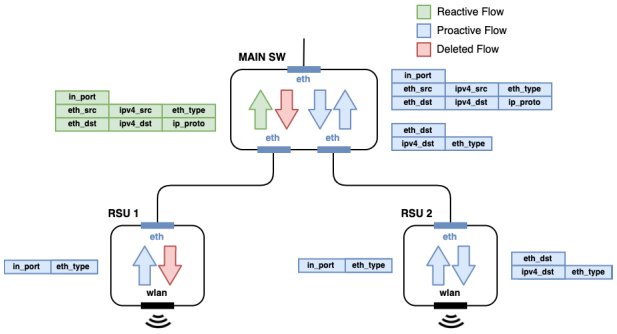


FIGURE 7. Flow modifications during handover in the *single switch level* architecture.

established reactive flow underwent a similar duplication process on the switch associated with the new RSU 3. If the RSU is directly linked to a higher-level switch (level 0, as in this case), flow duplication is unnecessary since a suitable flow already exists.

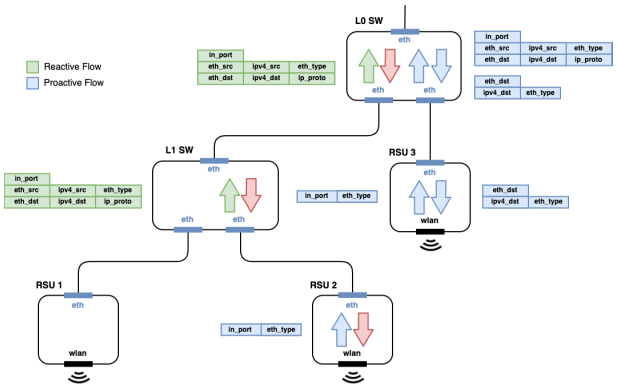


FIGURE 8. Flows modifications during handover in the *multiple switch levels* architecture.

C. OBU'S CONNECTION MANAGER

The OBU's connection manager (CM) is an essential element for the correct testing of the controller. It shares with the latter the decision logic for the horizontal handover: it is responsible for choosing the best RSU to which the OBU should connect at a given moment. The CM can be, as for the controller, divided into two main parts, working in parallel. The first part is responsible for the CAM messages handling, and the second is responsible for the association of the OBU to an RSU.

Concerning the first case, when a CAM message is received, its information is extracted, which contains the station ID, GPS coordinates (longitude and latitude), station type, RSSI (between OBU and RSU) and the timestamp. If it corresponds to an RSU that has not yet been registered, which means that no CAM message has been received from this RSU, an object is created for that RSU, associating all the above information with it. If the RSU already exists, the RSSI and the timestamp are updated.

In parallel, and regarding the association, when associating to an RSU, the CM only changes the IP of the default gateway to the IP corresponding to the RSU to which it wants to connect. This IP, although does not come in CAM messages, it is easily obtainable since the norm that is followed in this architecture, and already mentioned before, is that the last octet of the IPv4 address is based on the RSU ID.

V. PERFORMANCE EVALUATION

This section presents the lab and city setup scenarios, and the obtained results of the proposed approach. The following sections explain in greater detail the two types of environment used for the evaluation of the proposed solution. Section V-A focuses on a laboratory, controlled, environment while Section V-B focuses on a real-world city scenario. Both in the laboratory and city environments, a comparison will be made between the proposed solution and a pre-existing SDVN solution called reactive, which is based on L3 traffic triggered handovers [7].

A. LABORATORY ENVIRONMENT

This subsection addresses the setup used in the laboratory environment, the emulation scripts needed to test the solution, the scenarios used to illustrate specific use cases, as well as both the functional and application tests carried out in this environment.

1) SETUP

In this work, the system elements used to evaluate the proactive approach are the ones represented in Figure 1. The components used in this setup are the following:

- **SDN controller:** it is implemented on a machine running Ubuntu operating system. The controller application is based on the SDN framework RYU,[5] and it was chosen because it is an Open-Source framework that supports the OpenFlow protocol [20].
- **Main Switch / RSU / OBU:** it is implemented on PC Engines APU3 platforms (SBCs). Regarding the RSUs/OBUs, their boards contain an ITS-G5 (IEEE 802.11p) interface.
- **End user:** it is implemented using a Raspberry Pi running Ubuntu. It is directly connected to the OBU via an Ethernet interface.

Table 1 presents the specifications of the equipment used in laboratory environment.

2) EMULATION SCRIPTS

Evaluating solutions in laboratory environments has advantages and disadvantages. On one hand, it is possible to have a highly controlled environment in which it is possible to minimize external factors. On the other hand, this environment brings limitations regarding the movement of devices used in the solution. Since the proposed approach is based on the movement of an OBU along a given

[5]https://ryu-sdn.org

TABLE 1. Laboratory environment: equipment specifications.

Equipment	CPU [MHz]	Mem [MB]	Linux Kernel	OS
SDN Controller	3600 (2 cores)	4096	4.15.0	Ubuntu 18
APU	1000 (4 cores)	4096	5.7.10	Debian 8
RPi 3 Model B	1200 (4 cores)	1024	4.4.38-v7+	Ubuntu 16

path, which will imply changes in the GPS data (latitude, longitude, heading, etc.), in the RSSI and consequently in the throughput, scripts were created to emulate the displacement of the OBU along the path.

To support the controller, scripts are placed at the RSU level, which send emulated OBUInfo messages (CAMs plus RSSI) to the controller via OVS as if they were CAMs/RSSI coming from the OBU. Emulated messages vary their location (latitude, longitude, heading, etc.) and RSSI values over time to match a given scenario. Regarding the CM present in the OBU, it also needs emulated data to work, so there is also a script in the OBU that supports the CM.

To efficiently run the tests, it is necessary that the multiple scripts present on multiple machines can be executed synchronously. The synchronization of the various scripts is highly important for the information to reach both the controller and the CM in a synchronized manner - just like in a real-world scenario -, and guaranteeing that the intended route is correctly emulated. For this, an extra script is developed that remotely executes the several scripts on each machine. Figure 9 shows the emulation setup with the emulation scripts and their role on the setup.

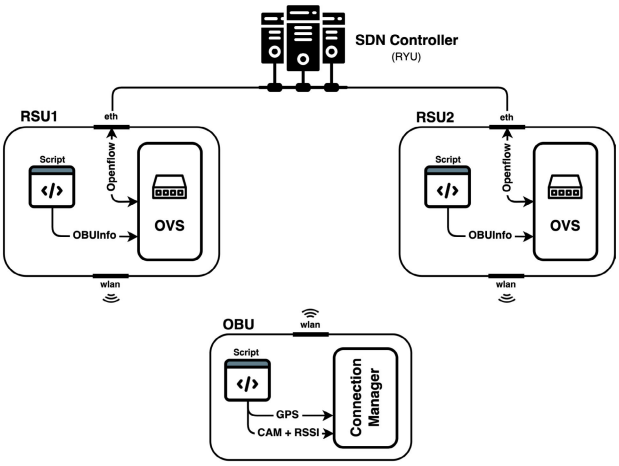


FIGURE 9. Emulation scripts architecture.

3) SCENARIOS

In this work, a VANET architecture is developed based on the concepts of SDN and ITS messages. For a proper evaluation of the proposed solution, several scenarios are created, emulating distinct and specific situations that may occur in a vehicular environment.

a: SCENARIO 1 – SMOOTH TRANSITION OF CAM MESSAGES BETWEEN RSUS

The first scenario focuses on testing how the controller/CM would react if a vehicle passes three RSUs with coverage intersection between them without signal obstruction. Therefore, the transition of CAMs between RSUs is accomplished with RSSI values varying smoothly as the vehicle approaches/departs from the RSUs. Figure 10 illustrates this scenario.

FIGURE 10. Scenario 1 – Smooth transition of CAM messages between RSUs. P1, P2 and P3 represent RSUs and colored circles represent their wireless coverage.

This scenario is aimed to show the perfect case where there is continuous coverage by RSUs. However, it is unrepresentative of the real-world, as it does not consider the obstacles present in the urban environment, such as buildings. The next two scenarios will consider some of the characteristics of the urban environment.

b: SCENARIO 2 – UNEVEN TRANSITION OF CAM MESSAGES BETWEEN RSUS

This scenario considers the signal degradation caused by buildings, other vehicles and others. Thus, this second scenario focuses on observing the behaviour of the solution in the case where an OBU passes through two RSUs with coverage intersection. However, in this scenario, there is signal obstruction along the way. Signal obstruction will cause fewer CAMs to reach the controller, which will cause the variation of RSSI values along the path to be non-continuous. Figure 11 shows this scenario.

c: SCENARIO 3 – LOSS OF CAMS RECEPTION

This scenario considers that not all zones are covered by RSUs, which translates into zones without connectivity. This third scenario consists on the analysis of the behaviour of the controller when it stops receiving CAMs from OBUs. Only two RSUs will be considered, each one at its end, and there is no wireless coverage intersection between the two (Figure 12).

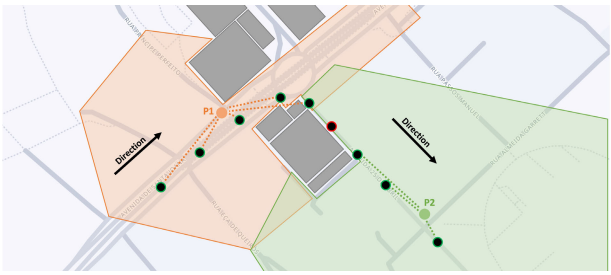


FIGURE 11. Scenario 2 – Uneven transition of CAM messages between RSUs. P1 and P2 represent RSUs, green points represent wireless coverage from either P1 or P2, and the red point is a location without wireless coverage due to obstacles.

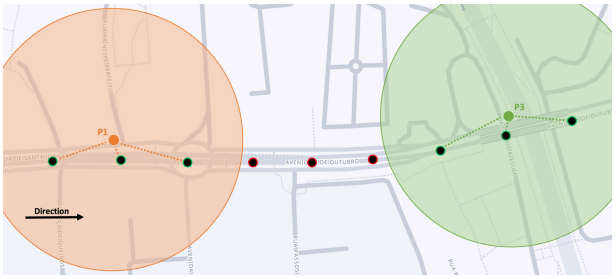


FIGURE 12. Scenario 3 – Loss of CAMs reception. P1 and P3 represent RSUs.

4) FUNCTIONAL TESTS

To analyze and evaluate the behaviour of the proposed solution, it is necessary to resort to a set of functional tests. This section will discuss the results of such tests when considering the three scenarios presented before.

a: SCENARIO 1

Several RSSI (dBm) values were defined that represent the signal strength between RSUs (P1, P2, P3) and the OBU over time, as shown in Table 2. The test lasts 1 minute and the period of connectivity with P1 and P3 is 24 seconds, and 30 seconds with P2. For every 12 seconds, it is possible to observe which RSU the OBU is connected to (logical link exists), as well as if there is connectivity with the infrastructure.

As can be seen in Table 2, the OBU starts connected to P1. After $t = 12$ *s*, the OBU begins to be within reach of P2 (confirmed by the reception of CAM messages transmitted by P2). A handover occurs at $t = 18$ *s* because the P2's RSSI is higher than that of P1's ($P2_{RSSI} \geq P1_{RSSI} +$ threshold), with a threshold value of 2, and the direction of the OBU matches with P2's location. Likewise, there is also a handover at $t = 36$ *s* from P2 to P3. In this scenario, the OBU has always connectivity to the infrastructure, so there is no interruption in services.

Table 3 shows the performance results (average values and standard deviation) in the *single switch* architecture. These tests were executed without services running in parallel, so they only revealed metrics regarding the control part of the solution.

TABLE 2. Scenario 1 - Connection status for different values of RSSI (* handover took place).

Time (s)	0	12	18	24	30	36	48
P1's RSSI (dBm)	-60	-60	-70	-	-	-	-
P2's RSSI (dBm)	-	-70	-60	-50	-60	-70	-
P3's RSSI (dBm)	-	-	-	-	-70	-60	-60
Direction	45°	45°	45°	45°	45°	45°	45°
Connected to	P1	P1	P2*	P2	P2	P3*	P3
Connectivity	yes	yes	yes	yes	yes	yes	yes

TABLE 3. Scenario 1 in the *single switch* architecture - controller performance without application traffic.

Delay handover (ms/hand.)	5.89 ± 0.97
Overhead (pkts/s)	15 ± 0.06
Overhead (kbytes/s)	2.22 ± 0.01

As far as the handover delay is concerned, this corresponds to the duration needed to update the switches with the correct flows. This operation is ideally done before there is a gateway change in the OBU, that is before there is the handover itself. Thus, although the value is high, it does not reflect communication interruption time.

Regarding overhead, this represents the control packets (OpenFlow) that circulate in the network. The overhead value can be obtained analytically based on the pre-known information. Knowing the period of connectivity with each RSU and the frequency of CAMs for this scenario, it is possible to calculate the average number of CAMs arriving at the controller, given by

$$\frac{(P1_{\text{conn\_time}} + P2_{\text{conn\_time}} + P3_{\text{conn\_time}}) \times CAM_{\text{freq}}}{test\_dur}.$$

Replacing the several variables by their respective values we obtain

$$\frac{(24 + 30 + 24) \times 10}{60} = 13\, pkts/s,$$

which represents the number of CAMs arriving at the controller per second. Now, adding to this value the remaining control packets used in the SDN domain, such as the *keep alive*, it results on 15 pkts/s.

Regarding the *multiple switch levels* architecture, the results for the Scenario 1 are presented in Table 4.

TABLE 4. Scenario 1 in the *multiple switch levels* architecture - controller performance without application traffic.

Delay handover (ms/hand.)	9.83 ± 1.19
Overhead (pkts/s)	16 ± 0.13
Overhead (kbytes/s)	2.31 ± 0.04

This version gets a higher handover delay than the previous one as the addition of a new level of switches creates the need to add flows to different levels to guarantee connectivity to the

outside of the vehicular domain. As the addition of flows is done sequentially, it increases the total duration of handover preparation. The overhead suffers only a slight increase due to the *keep alive* messages that circulate in the network, since this is essentially composed of CAM messages.

To better evaluate the behaviour of the proposed proactive solution, functional comparative tests were carried out between this version and the reactive SDVN solution [7]. These tests aim to analyze the OBU's RSSI of the logical link over time. Knowing that the reactive version relies on uplink traffic to work, tests were performed with uplink traffic with a very small packet size with the following periodicity: 1 every 5, 2, 1 and 0.1 (named as highly frequent) seconds. The results obtained are shown in Figure 13.

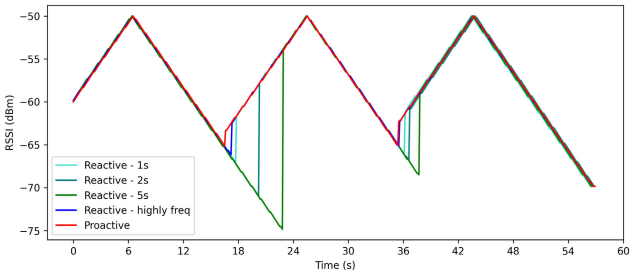


FIGURE 13. Scenario 1 – RSSI of the OBU's logical link.

In the proactive version, the RSSI values range between -65 and $-50\ dBm$. As can be seen, at $t = 16\ s$ and $t = 35\ s$, the RSSI increases abruptly, which indicates the occurrence of handovers. The transition is not entirely smooth because the controller only proceeds with the handover if this value exceeds a given threshold. In the reactive version, as the time between uplink traffic increases, the period of connection between the OBU and the previous RSU also increases. For this reason, the signal degrades, thus obtaining lower RSSI values. The abrupt changes in RSSI indicate the occurrence of handovers. It leads to the conclusion that, in the reactive version, handovers occur faster if the uplink traffic is more frequent. In case the uplink traffic is highly frequent, the behaviour becomes similar to the proactive case.

These tests have shown that the reactive version is uplink traffic-dependent for handovers to happen, and the longer the period between traffic, the greater the handover delay. It is also possible to conclude that the proactive version always obtains the best RSSI in the logical link, independent of the existence of uplink traffic.

b: SCENARIO 2

The analysis of this scenario is done in the same way as in the previous scenario, with RSSI and direction values being defined for various time points. The emulation lasts for 1 minute, and the period of connectivity with P1 and P2 is 30 seconds. As can be seen in Table 5, the OBU starts at P1 and, at $t = 24\ s$, P2 starts to receive CAM messages from the OBU, as it was the last time the OBU had coverage from P1. Still, at $t = 36s$ there is the handover to P2, not because the RSSI of P2 is greater than the last recorded by P1, but because the period since the last CAM received by P1 is higher than the CAM timeout (in these tests this value was set to 3 seconds). This timeout guarantees that the list of CAMs received is not out-of-date, so information with a duration longer than this timeout is not considered. Thus, the information related to P1 is no longer considered from that moment on, triggering a handover to P2 at $t = 33\ s$. After the handover, it remains connected to P2. The proactive solution, although discarding outdated information after a certain timeout, does not guarantee that connectivity exists at all times. At $t = 30\ s$, the OBU will still have a logical link associated with P1 but, as it is already out of reach, it turns out that there will be no connectivity to the infrastructure although it is already within P2's reach.

TABLE 5. Scenario 2 - Connection status for different values of RSSI (* handover took place).

Time (s)	0	12	18	24	30	36	48
P1's RSSI (dBm)	-70	-50	-55	-60	-	-	-
P2's RSSI (dBm)	-	-	-	-80	-70	-60	-50
Direction	45°	45°	45°	90°	135°	135°	135°
Connected to	P1	P1	P1	P1	P1	P2*	P2
Connectivity	yes	yes	yes	yes	no	yes	yes

Table 6 shows the controller related results obtained for this scenario considering a *single switch* architecture. These tests are also executed without application traffic.

TABLE 6. Scenario 2 in the *single switch* architecture - controller performance without application traffic.

Delay handover (ms/hand.)	2.77 ± 0.06
Overhead (pkts/s)	11 ± 0.01
Overhead (kbytes/s)	1.60 ± 0.01

Again, the overhead measures control packets (OpenFlow) circulating in the network, and its value is given by

$$\frac{(P1_{\text{conn\_time}} + P2_{\text{conn\_time}}) \times CAM_{\text{freq}}}{test\_dur},$$

which results in

$$\frac{(30+30)\times 10}{60} = 10\,pkts/s.$$

Taking into account that this second scenario contains fewer intersection coverage zones, it is normal that this number is lower than that obtained in the first scenario. To match with the value presented in Table 6, one must consider the remaining control packets inherent in the SDN domain.

Concerning the *multiple switch levels* architecture, the results are shown in Table 7. This version has also a higher handover delay than the previous one due to the addition of a new level of switches. The overhead suffers only a slight increase due to the *keep alive* messages circulating in the network.

TABLE 7. Scenario 2 in the *multiple switch levels* architecture - controller performance without application traffic.

Delay handover (ms/hand.)	6.84 ± 2.14
Overhead (pkts/s)	12 ± 0.11
Overhead (kbytes/s)	1.66 ± 0.01

c: SCENARIO 3

In this scenario the emulation lasts for 1 minute, and the period of connectivity with the RSU P1 and P3 is 24 seconds. As can be seen in Table 8, the OBU starts connected with P1, with connectivity until $t = 24$ s. At $t = 30$ s, P3 starts to receive CAM messages from the OBU. Between $t = 24$ s and $t = 30$ s there are no CAMs to be received so there is no connectivity during this period. However, the logical link with P1 was maintained.

TABLE 8. Scenario 3 - Connection status for different values of RSSI (* handover took place).

Time (s)	0	12	18	24	30	36	48
P1's RSSI (dBm)	-60	-60	-70	-	-	-	-
P3's RSSI (dBm)	-	-	-	-	-70	-60	-60
Direction	45°	45°	45°	45°	45°	45°	45°
Connected to	P1	P1	P1	P1	P3*	P3	P3
Connectivity	yes	yes	yes	no	yes	yes	yes

Table 9 shows the controller related results obtained for this scenario considering a *single switch* architecture. These tests, just like the ones performed in the previous scenarios, were executed without application traffic.

TABLE 9. Scenario 3 in the *single switch* architecture - controller performance without application traffic.

Delay handover (ms/hand.)	3.09 ± 0.89
Overhead (pkts/s)	10 ± 0.01
Overhead (kbytes/s)	1.41 ± 0.02

The overhead value obtained is given by

$$\frac{(P1_{\text{conn\_time}} + P3_{\text{conn\_time}}) \times CAM_{\text{freq}}}{test\_dur},$$

which results in

$$\frac{(24 + 24) \times 10}{60} = 8\, pkts/s.$$

Concerning the *multiple switch levels* architecture, the results are illustrated in Table 10. This version also has a higher handover delay than the previous one due to the addition of a new level of switches. Once again the overhead suffers a slight increase due to the *keep alive* messages that circulate in the network.

5) APPLICATION TESTS

For the application tests, two use cases are considered. The first one replicates a video streaming service, through UDP, and the second one simulates a file transfer through TCP. In both use cases, the OBU is the source of traffic, which means that the uplink direction encompasses the majority of the traffic exchanged between the OBU and the infrastructure. The services are created by the D-ITG tool,[6] a platform capable of generating traffic at packet level accurately replicating appropriate stochastic processes. Video streaming follows the set of parameters defined in [22] of 24 packets/s and packet size with a normal distribution of mean 27791 bytes and standard deviation of 6254 bytes. The file transfer is set to 700 packets/s and a constant packet size of 1000 bytes.

TABLE 10. Scenario 3 in the *multiple switch levels* architecture - controller performance without application traffic.

Delay handover (ms/hand.)	3.36 ± 0.18
Overhead (pkts/s)	11 ± 0.08
Overhead (kbytes/s)	1.47 ± 0.01

These two uses cases are performed for each of the three scenarios discussed in V-A3, evaluating the proactive SDVN solution presented in this work, considering the *single switch* architecture, against the reactive SDVN version for comparison purposes.

a: HANDOVER DELAY

Table 11 presents the handover delay observed during the tests. The SDVN reactive version consistently has significantly lower handover delay values than the proactive version, for all scenarios, regardless of the application. This is because the handover process in the proactive version is more complex, and therefore more time-consuming. However, the increase in handover delay does not translate into an increase in packet delay, as will be observed in sub-section c). The change of flows is done in advance, so that, when the handover happens, there is no delay between the instant of handover and the instant when the new flow is defined. Thus, the difference between the two versions is not significant, nor it represents a worse user experience.

TABLE 11. Handover delay [ms/handover].

	Scenario		
	1	2	3
Video Streaming (UDP)			
Reactive SDVN	0.04 ± 0.01	0.02 ± 0.01	0.02 ± 0.01
Proactive SDVN	8.44 ± 2.03	6.59 ± 1.81	7.03 ± 2.72
File Transfer (TCP)			
Reactive SDVN	0.03 ± 0.01	0.02 ± 0.01	0.02 ± 0.01
Proactive SDVN	8.22 ± 1.88	9.88 ± 1.96	6.80 ± 2.14

b: CONTROL PACKET OVERHEAD

Table 12 presents the overhead in terms of the number of control packets observed during the tests. Similarly, Table 13

[6] http://traffic.comics.unina.it/software/ITG/

presents the same results but in terms of number of bytes of control packets. The results obtained show a clear difference both in the number of packets as well as in the number of bytes between the proactive and reactive versions. As can be seen, the proactive version has higher values compared to the reactive version regardless of the use case. While the variation of values in the reactive version is relatively low, in the proactive version, the variation is highly influenced by the scenario. This increased overhead is explained by the fact that the proposed solution depends on the content of ITS messages to work. When CAM messages reach the RSUs, they are transformed into OpenFlow messages so that their information reaches the controller. Since each scenario presents different characteristics, both in the number of handovers/RSUs and in the number of signal intersections between RSUs, this value will vary. The first scenario, with two handovers and two signal intersection periods, will have a higher number of CAM messages reaching the controller, thus increasing the overhead. From the results, we can also conclude that the size of these OpenFlow messages is smaller than the rest of the OpenFlow control overhead because the relation between size and number of packets is smaller for the proactive version than for the reactive version.

TABLE 12. Control packet overhead [pkts/s].

	Scenario		
	1	2	3
Video Streaming (UDP)			
Reactive SDVN	2 ± 0.18	2 ± 0.28	2 ± 0.15
Proactive SDVN	17 ± 0.54	12 ± 0.10	10 ± 0.013
File Transfer (TCP)			
Reactive SDVN	2 ± 0.13	1 ± 0.10	1 ± 0.12
Proactive SDVN	16 ± 0.29	11 ± 0.08	10 ± 0.02

TABLE 13. Control packet overhead [kbytes/s].

	Scenario		
	1	2	3
Video Streaming (UDP)			
Reactive SDVN	0.40 ± 0.04	0.31 ± 0.05	0.18 ± 0.02
Proactive SDVN	2.49 ± 0.07	1.68 ± 0.01	1.48 ± 0.03
File Transfer (TCP)			
Reactive SDVN	0.27 ± 0.02	0.19 ± 0.02	0.31 ± 0.03
Proactive SDVN	2.37 ± 0.05	1.65 ± 0.01	1.47 ± 0.00

The advantages of the proactive version come at a cost: the increased control packet overhead. However, there may be ways to reduce this difference. The default value of CAMs transmission rate in the OBU is 10 *Hz*, a value that greatly influences the overhead value. By decreasing the amount of CAM messages that goes to the controller, it is possible to decrease the control overhead without compromising the solution's performance.

c: DELAY

Figure 14 presents the average delay observed during the tests. As can be seen from the results, in use case 2 (file transfer through TCP), the proactive SDVN version had lower average delay values than the reactive SDVN version, about 15% lower. Such reduction is common to the three scenarios and is in the same proportion for each of them. In use case 1 (video streaming through UDP), the proactive version obtained lower values, although the difference is no longer so noticeable.

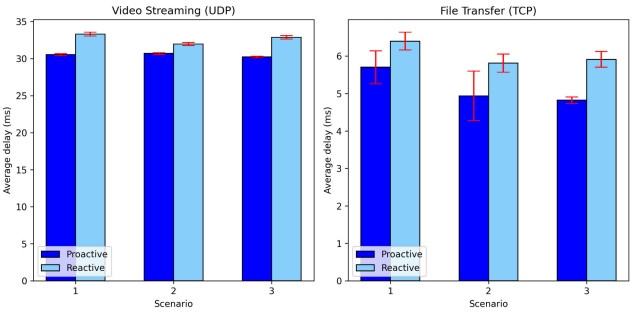


FIGURE 14. Average delay [ms].

This improvement is because, in the proactive version, flows are added to upcoming switches before there is an actual handover, *i.e.*, as there is already a flow, no additional time is wasted in that process. These results further reinforce the idea that the increase in handover delay in the proactive version does not translate into an increase in average delay. The average delay decreases, showing that there is no connection between the two metrics.

d: JITTER

Figure 15 presents the jitter observed during the tests. As can be seen from the results, regardless of the scenarios, the values are very similar between the two controller versions. The values vary between use cases, but not between scenarios and versions. The similarity between versions is because both solutions share a common trunk: both solutions process a packet equally. Thus, neither version of the controller is at an advantage over the other regarding this metric.

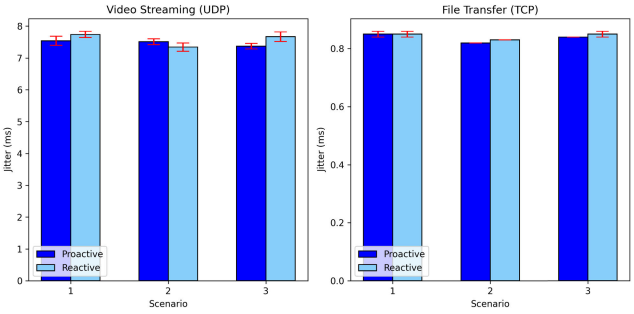


FIGURE 15. Average jitter [ms].

B. CITY ENVIRONMENT

This section describes the scenario, the vehicular setup, the tests and the performance results in a real-world city environment.

1) AVEIRO TECH CITY LIVING LAB

The Aveiro Tech City Living Lab (ATCLL) [23], deployed in Aveiro, Portugal, is an initiative created by the city and Instituto de Telecomunicações, that combines an advanced communications infrastructure with an urban platform for data management and innovative analytics. This infrastructure, illustrated in Figure 16, comprises 44 nodes strategically spread across the city, and are connected through fiber link technology to a data processing center.

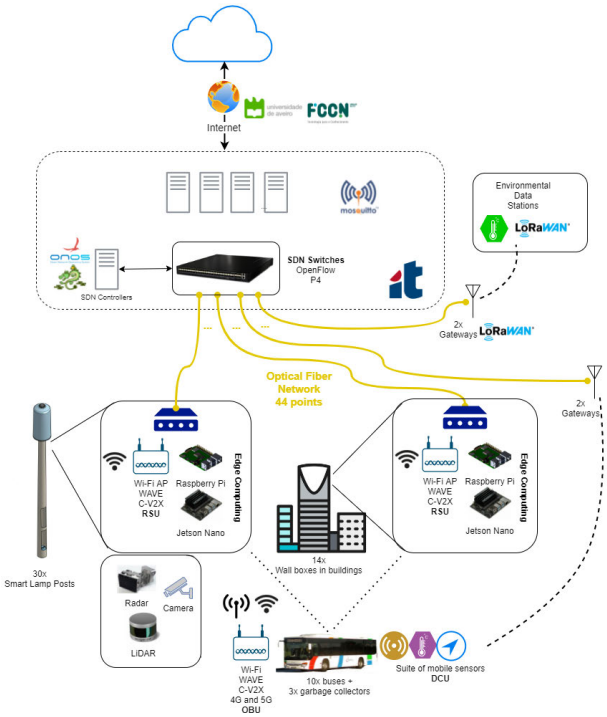


FIGURE 16. Aveiro Tech City Living Lab infrastructure.

Each one of the ATCLL node is a multi-technology communication access point (with 5G, 4G, ITS-G5, WiFi, and LoRaWAN/LoRa) also equipped with a Multi-access Edge Computing device (NVIDIA Jetson Nano, NVIDIA Jetson Xavier and Raspberry Pi 4). This platform will be used to run the city environment tests.

2) SETUP

The system elements used in the city environment setup are the same used in the laboratory environment, represented in Figure 1. For the real world environment, these are the components used:

- **SDN controller:** identical to the one used in the laboratory environment.
- **Main Switch / RSU / OBU:** implemented on PC Engines APU3 platforms (SBCs). For both RSUs and OBUs, their boards contain an ITS-G5 (IEEE 802.11p) interface. Regarding the RSUs, the SBCs were placed in 5 nodes of the ATCLL. As for the OBU, it contains a GPS antenna to obtain the current location, and it contains a ITS-G5 antenna more powerful than those used in the laboratory. The SBC is located inside a vehicle that travels a given path (Figure 17).
- **End user:** identical to the one used in the laboratory environment.

FIGURE 17. OBU Setup.

Table 14 presents the specifications of the equipment used in the city environment.

TABLE 14. City environment: equipment specifications.

Equipment	CPU [MHz]	Mem [MB]	Linux Kernel	OS
SDN Controller	2500 (4 cores)	8192	5.4.0-84-g	Ubuntu 20
APU	1000 (4 cores)	4096	5.7.10+	Debian 10
RPi 3 Model B	1200 (4 cores)	1024	4.4.38-v7+	Ubuntu 16

3) SCENARIO

The scenario consists of five RSUs (placed on P3, P5, P6, P19 and P26 of ATCLL) with coverage intersection between P3, P5 and P6. The wireless coverage of the RSUs placed on P19 and P26 do not intersect with any other RSU's wireless coverage. In this scenario, the vehicle travels along the path defined in Figure 18, thus passing through the referred RSUs, triggering horizontal handovers as it travels. The vehicle's average speed is about 25 km/h.

4) PERFORMANCE RESULTS

The use case is a file transfer (UDP) from the Cloud to the OBU, which means that it represents a downlink service. The file transfer is set to 10 Mb/s of bandwidth and a block size of 1448 bytes. A set of performance metrics are obtained with a frequency of 10 Hz. The route trip, as described in the previous section, covered a distance of around 2000 m, and the duration corresponds to the time required to cover such

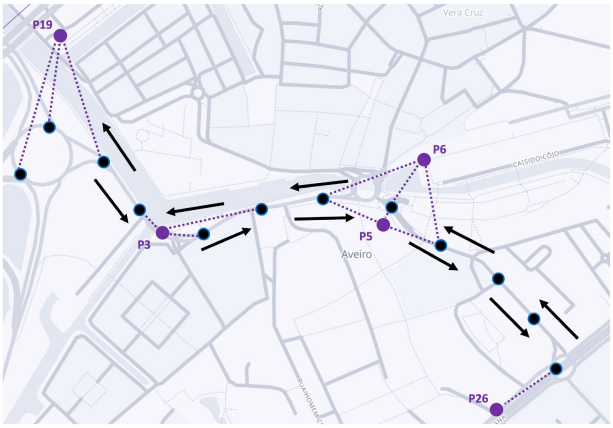


FIGURE 18. City environment: scenario.

distance. This duration varies between tests due to the fact that vehicular traffic conditions slightly changed between tests.

This use case is performed using two versions of the SDVN solution, the proactive version (in the *single switch* architecture) and the reactive version. The SDVN reactive version is tested with two uplink traffic rates, 1 packet every second (simply denoted as 1 *s*) and 1 packet every 5 seconds (denoted as 5 *s*). The reactive version is dependent on the uplink traffic to work, so an uplink traffic with very small packet size and duration is generated.

a: RECEIVED SIGNAL STRENGTH INDICATION

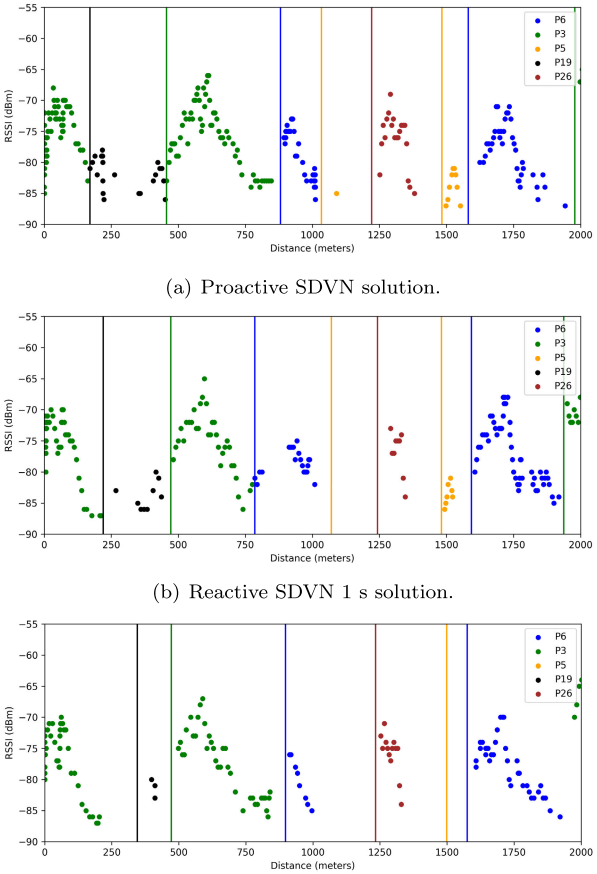
Figure 19 presents the RSSI over distance observed during the route for all versions. These values are obtained by the infrastructure which captures the CAMs that circulate in it. As a result, the frequency of values obtained in the figure changes along the way. The vertical lines represent the occurrence of handovers, as these are associated by colour with a given RSU. The existence of a line implies that a handover has occurred.

As can be observed, the proactive version makes handovers through all RSUs traversed by the OBU (P3, P19, P6, P5 and P26). Regarding the reactive SDVN 1 *s* version, it is already possible to observe that it does not perform horizontal handovers on all RSUs (e.g. on P5). In the reactive SDVN 5 *s* version, the same behavior happens more often which indicates that the OBU is missing better logical links with the infrastructure.

These results can be explained by the reaction time of each version. The more responsive the solution, the more intelligent the choice of the ideal RSU will be. Likewise, if there is an RSU with reduced coverage, the controller may not detect it, so the higher the responsiveness, the greater the number of RSUs detected. With a reduced range of the RSU P5, the responsiveness of the solution to the RSU detection becomes more crucial, especially if the vehicle speed is high.

b: THROUGHPUT

Figure 20 presents the throughput over distance observed during the route for all SDVN solutions. As can be seen from



(a) Proactive SDVN solution.

(b) Reactive SDVN 1 s solution.

(c) Reactive SDVN 5 s solution.

FIGURE 19. RSSI of the OBU's logical link over distance.

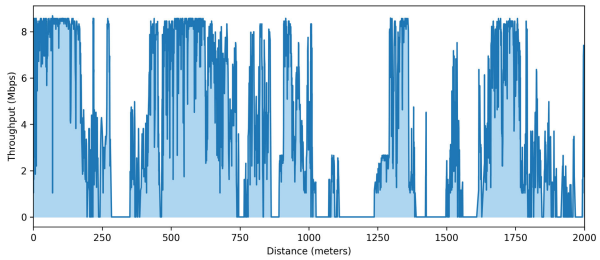
the results, the proactive version is the one that has the longest periods of logical connectivity with the infrastructure, since it is the one with the shortest periods of null throughput (no connectivity). Regarding the reactive SDVN 1 *s* version, the periods without connectivity are higher at 800 *m*, 1000 *m* and 1300 *m*. In the reactive SDVN 5 *s* version, these periods last even longer, with new periods of lack of connectivity around 25 *m*.

Table 15 summarizes the results of Figure 20. The proactive version obtains logical connectivity with the infrastructure (non-null throughput) during about 69% of the journey, while the reactive SDVN 1 *s* version obtains logical connectivity during 62% and the reactive SDVN 5 *s* is only able to achieve logical connectivity during 56% of the journey.

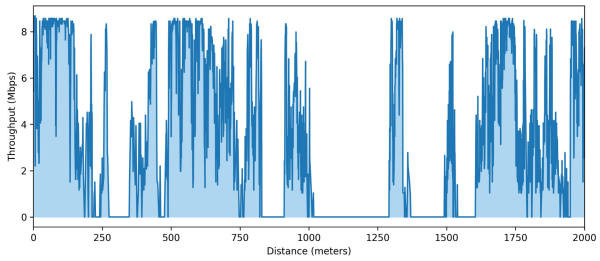
TABLE 15. Logical connectivity [%].

Proactive SDVN	Reactive SDVN 1 s	Reactive SDVN 5 s
68.69%	62.37%	56.01%

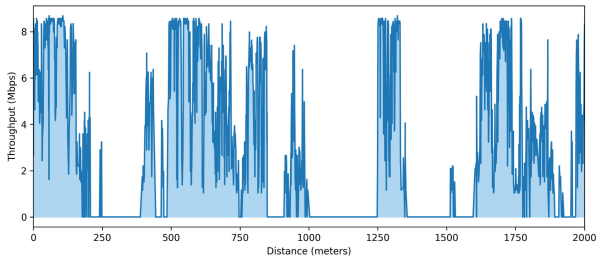
These results show the strong dependence of all SDVN versions on uplink traffic, which is always present in the



(a) Proactive SDVN solution.



(b) Reactive SDVN 1 s solution.



(c) Reactive SDVN 5 s solution.

FIGURE 20. Throughput over distance.

proactive version, since it corresponds to the CAMs that circulate through the network; in the reactive version, the traffic corresponds to the generated IP traffic. Regardless of the type of traffic, whether L2 or L3, its rate is the key to a quick network adaptation to the vehicle's movement.

Figure 21 presents a comparison of all SDVN versions concerning throughput over distance observed during the route. Each curve represents a moving average, with a sliding window of 50, of all the points associated with each version (null values included). This figure allows the simplification of Figure 20 by joining the three previous sub-figures into one. It is possible to observe the periods of non-connectivity of some versions previously reported, in which the solutions, due to the lack of responsiveness, delay the handover to a new RSU.

Figure 22 shows the non-conditional average throughput for each of the versions, which take into account all values (with and without connectivity). It is possible to observe different throughput values for each SDVN version. The proactive version achieves the highest average with 3.5 *Mbps* of throughput, against 3.2 *Mbps* and 2.7 *Mbps* achieved by the reactive SDVN 1 *s* and reactive SDVN 5 *s*, respectively.

These results are explained by the fact that, as mentioned before, the proactive SDVN version obtains more extended

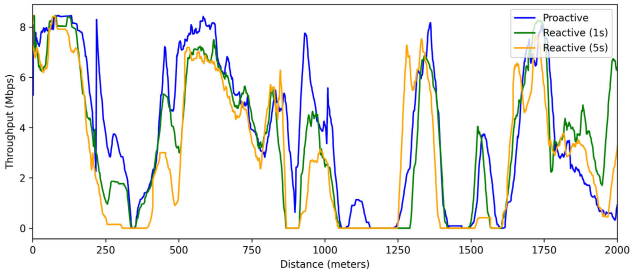


FIGURE 21. Throughput over distance comparison.

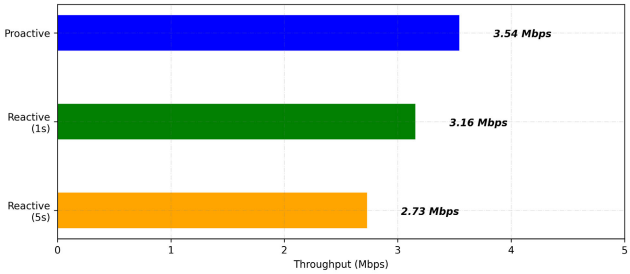


FIGURE 22. Average throughput comparison (considering null and non-null samples).

periods of connectivity with the infrastructure, so it gets fewer values of zero (without connectivity). Another factor that also influences the throughput is the long delay of handovers in the reactive versions, although its impact is lower.

Figure 23 shows the conditional average throughput for each SDVN version. These values only take into account the connectivity periods. By removing the null throughput values, the three versions get very similar values around 5 *Mbps*, with the proactive version getting 5.2 *Mbps*, the reactive 1 *s* 5.1 *Mbps* and the reactive 5 *s* getting the worst result with 4.9 *Mbps*.[7]

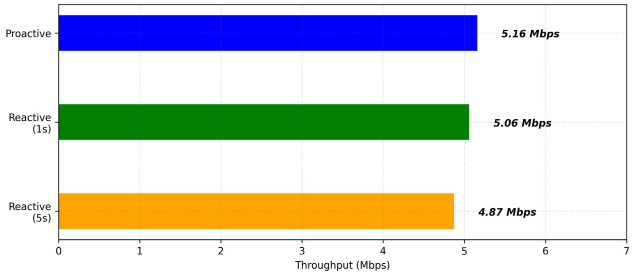


FIGURE 23. Average throughput comparison (considering only non-null samples).

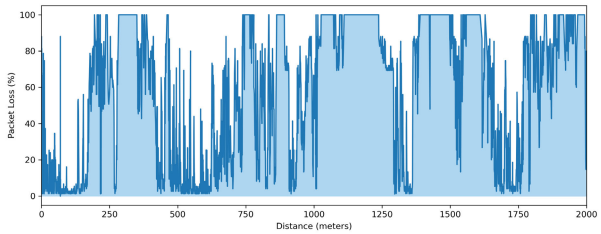
In a city environment, there is not such a smooth variation of RSSI, and consequently of throughput, as seen in a laboratory environment. It is possible to move from zones

[7]It should be noted that such throughput values are justified by the use of IEEE 802.11p/ITS G5, antenna gains, and many other elements, for the connection between the infrastructure (RSUs) and the moving vehicles (OBUs). Still, it is not the scope of this work to assess the performance of such wireless links.

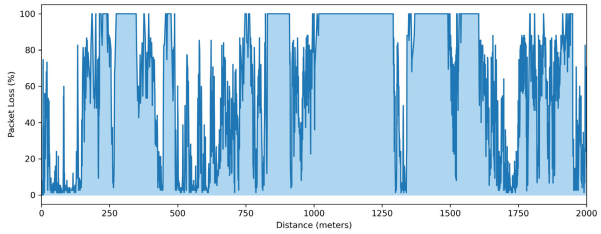
with connectivity to zones without it very quickly. As a result, the reactive SDVN versions get longer periods without connectivity compared to the proactive version, resulting in a lower value of average throughput, still very close to the one achieved by the proactive SDVN version.

c: PACKET LOSS

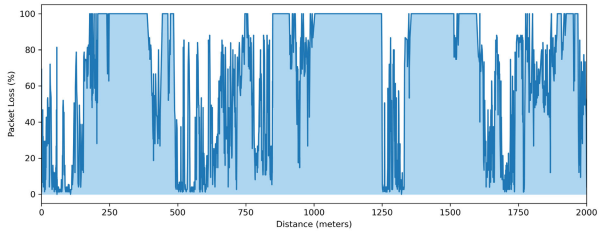
Figure 24 presents the packet loss over distance observed during the route. As we can observe, between 100 *m* and 1600 *m*, the proactive version has shorter periods with total packet loss when compared to the same time period for the reactive versions. This time period matches with the period already discussed during the throughput analysis, which is null for a longer distance. In this way, it is possible to observe the relationship between throughput and packet loss since the two are inversely interconnected. Once again, these results show the great importance of the responsiveness of solutions to RSU changes.



(a) Proactive SDVN solution.



(b) Reactive SDVN 1 s solution.



(c) Reactive SDVN 5 s solution.

FIGURE 24. Packet loss over distance.

VI. CONCLUSION

This article researched an approach to combine the use of C-ITS messages (CAMs) with an SDN architecture in a vehicular environment, to improve awareness of the vehicular nodes and the network disruptions. To achieve this, it was necessary to bring the SDN domain to the RSUs level, thus passing them to the controller's control. The addition of C-ITS messages in this solution allowed the controller to offer a deeper view of the status of each OBU, which includes GPS, RSSI, and other parameters, thus being able to make necessary changes in advance.

Through lab and real city testing, we have show that the proposed solution reduces the average delay, which contributed to a better user experience. The increase in connectivity time is a big advantage, which allows for higher stability in the use of the network. The results obtained allowed not only to evaluate the correct operation of the proposed approach, but also to compare it over a state of the art approach and observe its advantages.

Future improvements of this work include the increase in the number of SDN-capable entities. Bringing the SDN domain down to the OBU level would allow the OBU to be controlled by the controller. Such an approach could eventually eliminate the need for ITS messages. Other topics of future research include the integration of new communication technologies. Currently, the solution only exploits ITS-G5 (802.11p) technology, which has its advantages but also its disadvantages. The use of other technologies such as cellular, including C-V2X, would make it possible to offer multihoming to the solution, making it more versatile and robust. Finally, the addition of a more complex mobility path prediction based on machine learning algorithms is also a topic of future research.

REFERENCES

[1] C. Perkins, *IP Mobility Support for IPv4, Revised*, document RFC 5944, Internet Requests for Comments, RFC Editor, Nov. 2010. [Online]. Available: https://datatracker.ietf.org/doc/html/rfc5944

[2] V. Devarapalli, R. Wakikawa, A. Petrescu, and P. Thubert, *Network Mobility (NEMO) Basic Support Protocol*, document RFC 3963, RFC Editor, Jan. 2005. [Online]. Available: https://datatracker.ietf.org/doc/html/rfc3963

[3] S. Gundavelli, *Proxy Mobile IPv6*, document RFC 5213, RFC Editor, Aug. 2008. [Online]. Available: https://datatracker.ietf.org/doc/html/rfc5213

[4] I. Soto, C. J. Bernardos, M. Calderon, A. Banchs, and A. Azcorra, "Nemo-enabled localized mobility support for internet access in automotive scenarios," *IEEE Commun. Mag.*, vol. 47, no. 5, pp. 152–159, May 2009. [Online]. Available: https://ieeexplore.ieee.org/document/4939291

[5] Z. He, J. Cao, and X. Liu, "SDVN: Enabling rapid network innovation for heterogeneous vehicular communication," *IEEE Netw.*, vol. 30, no. 4, pp. 10–15, Jul. 2016. [Online]. Available: https://ieeexplore.ieee.org/document/7513858

[6] N. B. Truong, G. M. Lee, and Y. Ghamri-Doudane, "Software defined networking-based vehicular Adhoc network with fog computing," in *Proc. IFIP/IEEE Int. Symp. Integr. Netw. Manag. (IM)*, May 2015, pp. 1202–1207. [Online]. Available: https://ieeexplore.ieee.org/document/7140467

[7] M. Silva, P. Teixeira, C. Gomes, D. Dias, M. Luís, and S. Sargento, "Exploring software defined networks for seamless handovers in vehicular networks," *Veh. Commun.*, vol. 31, Oct. 2021, Art. no. 100372. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2214209621000413

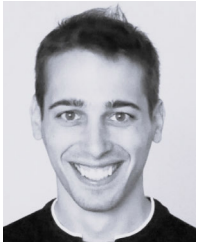
[8] *Final Draft ETSI EN 302 637-2 V1.3.1*, European Standard EN 302 637-2, ETSI, Sep. 2014. [Online]. Available: https://www.etsi.org/deliver/etsi_en/302600_302699/30263702/01.03.01_30/en_30263702v010301v.pdf

[9] D. Dias, "A software defined vehicular network using cooperative intelligent transport system messages," M.S. thesis, Dept. de Eletrónica, Telecomunicações e Informática, Univ. Aveiro, Aveiro, Portugal, Dec. 2021. [Online]. Available: https://ria.ua.pt/handle/10773/34072

[10] N. Cardona, E. Coronado, S. Latré, R. Riggio, and J. M. Marquez-Barja, "Software-defined vehicular networking: Opportunities and challenges," *IEEE Access*, vol. 8, pp. 219971–219995, 2020.
[11] B. Alaya and L. Sellami, "Toward the design of an efficient and secure system based on the software-defined network paradigm for vehicular networks," *IEEE Access*, vol. 11, pp. 43333–43348, 2023.
[12] W. Quan, N. Cheng, M. Qin, H. Zhang, H. A. Chan, and X. Shen, "Adaptive transmission control for software defined vehicular networks," *IEEE Wireless Commun. Lett.*, vol. 8, no. 3, pp. 653–656, Jun. 2019.
[13] L. Nkenyereye, L. Nkenyereye, S. M. R. Islam, C. A. Kerrache, M. Abdullah-Al-Wadud, and A. Alamri, "Software defined network-based multi-access edge framework for vehicular networks," *IEEE Access*, vol. 8, pp. 4220–4234, 2020.
[14] H. Li, M. Dong, and K. Ota, "Control plane optimization in software-defined vehicular ad hoc networks," *IEEE Trans. Veh. Technol.*, vol. 65, no. 10, pp. 7895–7904, Oct. 2016.
[15] X. Yin and L. Wang, "A fast handover scheme for SDN based vehicular network," in *Mobile Ad-Hoc and Sensor Networks*, vol. 747. Singapore: Springer, 2018, pp. 293–302.
[16] S. M. Raza, D. S. Kim, and H. Choo, "Leveraging PMIPv6 with SDN," in *Proc. 8th Int. Conf. Ubiquitous Inf. Manage. Commun.* New York, NY, USA: Association for Computing Machinery, Jan. 2014, pp. 1–8, doi: 10.1145/2557977.2558056.
[17] R. Duo, C. Wu, T. Yoshinaga, and Y. Ji, "SDN-based handover approach in IEEE 802.11p and LTE hybrid vehicular networks," in *Proc. IEEE SmartWorld, Ubiquitous Intell. Comput., Adv. Trusted Comput., Scalable Comput. Commun., Cloud Big Data Comput., Internet People Smart City Innov. (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI)*, Oct. 2018, pp. 1870–1875.
[18] A. Kaul, K. Obraczka, M. A. S. Santos, C. E. Rothenberg, and T. Turletti, "Dynamically distributed network control for message dissemination in ITS," in *Proc. IEEE/ACM 21st Int. Symp. Distrib. Simul. Real Time Appl. (DS-RT)*, Oct. 2017, pp. 1–9.
[19] S. Correia, A. Boukerche, and R. I. Meneguette, "An architecture for hierarchical software-defined vehicular networks," *IEEE Commun. Mag.*, vol. 55, no. 7, pp. 80–86, Jul. 2017. [Online]. Available: https://ieeexplore.ieee.org/document/7981530
[20] *OpenFlow Switch Specification*. Accessed: Nov. 2021. [Online]. Available: https://opennetworking.org/wp-content/uploads/2014/10/openflow-switch-v1.5.1.pdf
[21] B. Pfaff and B. Davie, *The Open vSwitch Database Management Protocol*, document RFC 7047, RFC Editor, Dec. 2013. [Online]. Available: https://datatracker.ietf.org/doc/html/rfc7047.txt
[22] M. W. Garrett and W. Willinger, "Analysis, modeling and generation of self-similar VBR video traffic," in *Proc. Conf. Commun. Archit., Protocols Appl.*, Oct. 1994, pp. 269–280.
[23] P. Rito, A. Almeida, A. Figueiredo, C. Gomes, P. Teixeira, R. Rosmaninho, R. Lopes, D. Dias, G. Vítor, G. Perna, M. Silva, C. Senna, D. Raposo, M. Luís, S. Sargento, A. Oliveira, and N. B. de Carvalho, "Aveiro tech city living lab: A communication, sensing and computing platform for city environments," *IEEE Internet Things J.*, vol. 10, no. 15, pp. 13489–13510, Mar. 2023.

DUARTE DIAS received the M.Sc. degree in computer and telematics engineering from the University of Aveiro, Portugal, in 2021. Since 2020, he has been a Researcher with the Instituto de Telecomunicações (IT), Aveiro, Portugal, exploring areas, such as software defined vehicular networks, machine learning in vehicular networks, and quality-of-service. Currently, he is associated with research projects, such as IMMINENCE (Celtic-NEXT Program) and Route 25 (PRR Agenda). His current research interests include vehicular communications, network management, virtualization, and smart cities.

MIGUEL LUÍS received the M.Sc. and Ph.D. degrees in electrical and computer engineering from the Faculdade de Ciências e Tecnologia, Universidade Nova de Lisboa, Portugal, in 2009 and 2015, respectively. He is currently an Assistant Professor with the Instituto Superior Técnico/Universidade de Lisboa (IST/UL) and a Researcher with the Instituto de Telecomunicações. He has been involved in several national and European research projects targeting new communications for mobile networks. Currently, he is the Coordinator of "MH-SDVanet: Multihomed Software Defined Vehicular Networks," a national funded research project, and he contributes to several other research projects, such as POWER (P2020 Program), and Route 25 and New Space (PRR Agenda), to name a few. He has published more than 100 scientific works, including three book chapters and 48 publications in peer-reviewed international journals. His research interests include medium access control for wireless systems, routing and dissemination mechanisms for mobile networks and management, orchestration, and softwarization of future networks.



PEDRO RITO (Member, IEEE) received the M.Sc. degree in electronics and telecommunications engineering from the University of Aveiro, Portugal, and the Ph.D. degree in electrical engineering from Technische Universität Berlin, Germany, in 2011 and 2019, respectively. In 2012, he joined IHP GmbH, Frankfurt (Oder), Germany, as a Researcher. In 2018, he joined Cisco Optical GmbH, Nürnberg, Germany, as a Research and Development Engineer, investigating and developing state-of-the-art high-efficient and high-bandwidth electro-optical interconnects for datacenters, metro, and long-haul communications. Since 2020, he has been with the Network Architectures and Protocols Group, Instituto de Telecomunicações (IT), Aveiro, Portugal, as an Assistant Researcher. During his research activity, he has published more than 50 publications (20 of which in peer-reviewed journals) and he holds one U.S./EU patent. His current research interests include radio access networks, software-defined networking, vehicular communications, edge computing, and smart cities.

SUSANA SARGENTO (Member, IEEE) is currently a Full Professor with the University of Aveiro and a Senior Researcher with the Instituto de Telecomunicações, where she is also leading the Network Architectures and Protocols Group. She was a Visiting Ph.D. Student with Rice University (2000–2001) and a Guest Faculty Member of Carnegie Mellon University, in 2008. She has been leading and involved in several European projects, CMU and MIT-Portugal projects and in several PRR agendas on autonomous mobility, space, two-wheels vehicles, tourism, and different testbeds. Her main research interests include self-organized networks, intelligent transportation systems, 5G, and beyond networks and services, with more than 450 publications, with two large-scale communication and sensing platforms, in Porto and Aveiro, with Aveiro TechCity Living Laboratory. She has co-founded a vehicular networking company, in 2012, Veniam (www.veniam.com). She is the Winner of the 2016 EU Prize for Women Innovators, the Winner of Femina 2020 Prize in Science, and one of the nominated of Prize ACTIVA Inspiring Women, in 2021. She was the Co-Coordinator of the National Initiative of Digital Competences in the Research Axis INCoDe.2030, belonged to the evaluation committee of the Fundo200M (www.200m.pt) and is one of the Scientific Director of CMU-Portugal Programme. She regularly acts as an Expert of European Research Programmes.