future internet

MDPI

Article

SDN-Based Secure Common Emergency Service for Railway and Road Co-Existence Scenarios

Radheshyam Singh [1,*,†], Leo Mendiboure [2,*,†], José Soler [1], Michael Stübert Berger [1], Tidiane Sylla [3], Marion Berbineau [4] and Lars Dittmann [1]

[1] Department of Electrical and Photonics Engineering, Technical University of Denmark, 2800 Kgs Lyngby, Denmark; joss@fotonik.dtu.dk (J.S.); msbe@fotonik.dtu.dk (M.S.B.); ladit@fotonik.dtu.dk (L.D.)
[2] COSYS-ERENA, University Gustav Eiffel, IFSTTAR, F-33067 Bordeaux, France
[3] Department GEII-ISA, University of Sciences, Techniques and Technologies of Bamako, 33067 Bordeaux, France; tidiane.sylla@univ-eiffel.fr
[4] COSYS-LEOST Lab, University Gustave Eiffel, IFSTTAR, 59650 Villeneuve d'Ascq, France; marion.berbineau@univ-eiffel.fr
* Correspondence: radsi@dtu.dk (R.S.); leo.mendiboure@univ-eiffel.fr (L.M.)
† These authors contributed equally to this work.

Abstract: In the near future, there will be a greater emphasis on sharing network resources between roads and railways to improve transportation efficiency and reduce infrastructure costs. This could enable the development of global Cooperative Intelligent Transport Systems (C-ITSs). In this paper, a software-defined networking (SDN)-based common emergency service is developed and validated for a railway and road telecommunication shared infrastructure. Along with this, the developed application is capable of reducing the chances of distributed denial-of-service (DDoS) situations. A level-crossing scenario is considered to demonstrate the developed solution where railway tracks are perpendicular to the roads. Two cases are considered to validate and analyze the developed SDN application for common emergency scenarios. In case 1, no cross-communication is available between the road and railway domains. In this case, emergency message distribution is carried out by the assigned emergency servers with the help of the SDN controller. In case 2, nodes (cars and trains) are defined with two wireless interfaces, and one interface is reserved for emergency data communication. To add the DDoS resiliency to the developed system the messaging behavior of each node is observed and if an abnormality is detected, packets are dropped to avoid malicious activity.

Keywords: SDN; DDoS; railways; roads; emergency service; Mininet-WiFi; ONOS; latency

Citation: Singh, R.; Mendiboure, L.; Soler, J.; Berger, M.S.; Sylla, T.; Berbineau, M.; Dittmann, L. SDN-Based Secure Common Emergency Service for Railway and Road Co-Existence Scenarios. *Future Internet* **2024**, *16*, 122. https://doi.org/10.3390/fi16040122

Academic Editor: Manabu Tsukada

Received: 11 March 2024
Revised: 20 March 2024
Accepted: 27 March 2024
Published: 2 April 2024

1. Introduction

Today, the cohabitation of rail and road environments is unavoidable in many countries, due to the existence of level crossings with varying levels of safety (presence or absence of barriers, for example) or the deployment of trains in an urban environment, cohabiting with cars, as can be seen today in China [1]. These points of intersection are proving to be accitentogenic, in particular, due to human error, especially for car drivers, major emergency braking constraints in the rail environment (several hundred meters for high-speed lines), and lack of synchronization between railway and car environments [2]. As a result, securing the coexistence of these two worlds has become a priority for many regions and public and private rail companies.

To achieve this goal, the first idea implemented was to define new services within each domain. For example, in the railway domain, the European Railway Traffic Management System (ERTMS) [3], based on the Global System for Mobile Communications—Railway (GSM-R) [4], supports the enhanced Railway Emergency Call (eREC) [5]. The eREC is based on the Voice Group Call Service (VGCS) with additional preemption features. When

a user initiates an eREC call, the request arrives at the serving GSM-R network as shown in Figure 1, which creates a group call to all the entities in the same area, as well as to the area of the dispatcher/controller. The call is created even if no resources are initially available for it, as the network disconnects any other, lower-priority, call to free resources and prioritizes the eREC communication. Similarly, in road communications, the European Telecommunications Standards Institute (ETSI) has defined two main message types: the Cooperative Awareness Message (CAM) and the Decentralized Environmental Notification Message (DENM) [6]. CAM messages provide periodic updates about the vehicle's status, while DENM messages transmit specific, often emergency-related, information. These messages can indicate obstacles, lane changes, or sudden slowdowns, and are disseminated quickly in the local area via vehicle-to-vehicle (V2V) communications and more broadly through roadside infrastructure. Therefore, it can be inferred that both sectors have mechanisms to manage emergencies but no unified system exists across them.

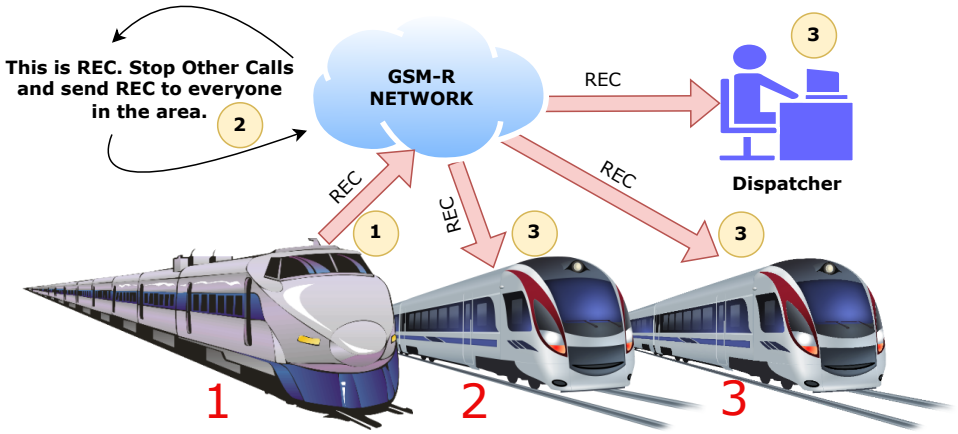

Figure 1. Railway Emergency Call [7].

However, a growing idea today is to design joint services, interconnecting road and rail services. Indeed, having a common service to distribute the emergency alert could offer several benefits. It could improve communication between different emergency services and it could also ensure that both the domains have the same information at the same time. This will help to build coordinated and efficient responses to emergency situations and to establish a common understanding of an emergency situation that can improve decision making and response time [8]. Along with these, sharing civil and network/telecommunication infrastructures for railway and road coexistence scenarios restricts duplication and minimizes direct investment towards marginal areas. In this way, it reduces the capital expenditure (CapEx) and operating expense (OpEx) [9,10]. However, effectively addressing and managing emergency or critical situations within the operational contexts of railways and roads remains an ongoing and open area of inquiry [11].

Software-defined networking (SDN) is one of the technologies that could guarantee fast, efficient distribution of information across shared or independent communication infrastructures, particularly for critical or emergency situations. SDN can oversee and manage the entire network as a single entity and eliminates the need for individual device configuration. It allows for precise routing and prioritization of traffic based on specific needs. It also offers advanced data traffic differentiation and centralized network control. SDN controllers facilitate centralized information collection and processing, unlike traditional networks that require comprehensive information exchange for security. Indeed, the centralized SDN controller introduces a vulnerability as it becomes a single point of failure for the entire communication network. To mitigate this risk, the deployment of multiple controllers, as suggested in [12–15], could be an effective solution. This approach enhances the reliability and robustness of the network by eliminating the single point of failure.

The SDN technology has been widely applied in the road environment, as demonstrated by [16,17], but also in the rail environment [18]. This demonstrates the potential of this solution to guarantee the implementation of a solution that (1) guarantees the independence of potential multiple network/infrastructure operators, (2) guarantees efficient message distribution, and (3) reinforces network security. However, such work has given little consideration to the idea of common services and network interconnections, even though such solutions have already been proposed and implemented in other environments [19].

That is why in this article we propose an SDN-based solution for implementing common emergency services for road and rail environments. In the complex interplay of road and rail systems, numerous scenarios necessitate the intervention of common emergency services. These include, but are not limited to, road accidents in proximity to railway tracks, infrastructure failures, and public-safety threats. A dangerous situation occurs frequently when a car is stacked across the railway track in a level-crossing area. If the car could send an emergency message not just to a car emergency system, but also to a rail emergency system, the approaching train could be alerted in time to prevent a potential collision. This is the case considered and presented in this paper. The aim of the proposed architecture and mechanisms, leveraging SDN capabilities, is to enable (1) coordinated management of emergency messages for the rail and road environments; (2) isolation of services intended solely for one of the two environments; (3) prevention of attacks that might seek to take advantage of this sharing of services by relying on the effective management of attacks that SDN is likely to enable [20–22]; and finally, (4) the applicability of the proposed solution in a real-life environment. The code for the proposed SDN application is available online [23], and this article aims to present the mechanisms implemented for traffic differentiation and malicious behavior detection, as well as to demonstrate how the implemented solution works.

This paper is organized as follows: Section 2 gives information about some related work. Section 3 presents an overview of the mechanism to implement the secure and common emergency service. Section 4 provides information about the proof of concept (PoC) to implement the SDN application. Validation of the developed system to distribute the common emergency messages is presented in Section 5 and tackling DDoS attacks is presented in Section 6. Finally, Section 7 concludes the paper with a critical analysis.

2. Related Work and Motivation

Existing SDN-based work aimed at improving coexistence between rail and road environments can be divided into two categories:

- **Work focused on a single domain:** This work aims to enable more efficient dissemination of information in either the rail or road environment. The idea is, therefore, to try to ensure global safety without implementing exchanges between road and rail servers/terminals. Most of these solutions have been developed for the road environment. Indeed, the recent and numerous studies on automated and connected vehicles have encouraged the development of proposals designed to guarantee a high level of safety. In this road context, we can cite the following as interesting examples [24–26]: they aim to manage and anticipate the mobility of cars in the best possible way to guarantee the dissemination of critical messages with minimum latency. Similar work can be found in the railway environment, such as [18]. These projects have two similar features: (1) mobility management adapted to both rail and road environments, and (2) traffic differentiation designed to guarantee faster dissemination of critical messages. However, they do not enable the interconnection of rail and road environments, and therefore, cannot ensure global safety.
- **Work seeking to propose a solution common to both domains:** The idea here is to ensure that safety is shared by both the road and rail domains, and thus, to guarantee the transmission of information between these two environments. So far, this idea seems to have received little consideration in the context of SDN-based network

architectures and, as far as we know, our article [27] was the first to mention it. However, the existence of work in a non-SDN context, aimed at enabling the exchange of critical information [8], both in research articles and in European projects, seems to demonstrate the relevance of this approach. Furthermore, the existence of SDN-based applications in each of the environments also seems to be an argument in favor of implementing such solutions interconnecting both environments.

Therefore, existing research in the field has primarily focused on narrow aspects of SDN-based common emergency services for vehicular networks and railways. Most researchers have considered only vehicular networks in which numerous network-based solutions are currently being developed. Prior to the documentation of our current practical study, no previous studies were found that explored the implementation of a unified messaging for emergency services using a centralized SDN-based networking application.

In addition, in order to limit the risks that the interconnection of these two environments could generate, we also propose integrating elements designed to reinforce system security. Indeed, the opening of doors between the two communication systems could be used by attackers to carry out denial-of-service (DoS) attacks against the rail and road environments, with the impact maximized by the duplication of messages to both environments. An attack that floods a network or server with data or internet traffic is referred to as a DoS/DDoS attack [28]. We have, therefore, integrated a message filtering mechanism into our architecture, taking our inspiration from the solutions described in the literature for managing DoS attacks in SDN-based systems [29,30]. On this point, our contribution consists of adapting these solutions to the environment under consideration (interconnection of heterogeneous rail–road systems) and proposing an open-source implementation that can be reused by other researchers in the future, both in rail and road environments and in other environments.

3. Description of the Proposed SDN-Based Service for Rail and Road Coexistence

The aim of this section is to present key information related to the SDN-based service we have implemented for the coexistence of road and rail environments, particularly in emergency scenarios. The proposed solution aims to address the limitations of existing solutions in the literature (see Section 2).

3.1. Objectives of the Implemented Service

The implementation of an SDN-based solution relies on the implementation of an SDN application capable of managing the entire road–rail emergency service. The SDN application that we developed can be decomposed into two parts: (1) Differentiating the emergency messages from common messages to distribute them to both rail and road emergency servers, and (2) detecting DoS/DDoS attack scenarios that could disrupt system operation:

1. **Managing emergency messages:** Cars and trains can only communicate with their own servers and with devices/nodes that belong to the same VLAN of the same physical network (cars/trains). However, during emergency situations, the application must have the capability to distribute emergency data messages to every node in the network, including assigned emergency servers. This means that regardless of whether it is a car or a train, all nodes will receive the emergency message. Detailed information about common emergency service implementation is presented in Sections 4.1 and 4.2. The application addresses two specific cases:
    - **Case 1: Sending emergency messages to assigned servers**
      When an emergency message is sent from cars to the car emergency server, the message is also forwarded to the rail emergency server. Similarly, if an emergency message is sent from trains to the rail emergency server, it is also transmitted to the car emergency server. In this case, an emergency message is first sent to the centralized SDN controller and the controller distributes the

message to assigned servers. The servers, upon receiving these emergency messages, disseminate them to the cars and trains located in the affected area. This mechanism ensures the comprehensive distribution of emergency messages among the relevant vehicles.

- **Case 2: Sending emergency messages directly to cars, trains, and to assigned servers** In this case, our SDN application enables direct communication of emergency messages between cars and trains. It allows cars and trains to send and receive only emergency messages to each other, without the need for routing through controller and intermediate servers. To enable this mechanism, all the nodes (cars and trains) are defined with two wireless interfaces, where one interface is reserved only for emergency data communication.

2. **Handling DoS/DDoS attacks:** To avoid DoS, a mechanism is developed based on monitoring the sudden change in the messaging behavior during the transmission of the emergency messages. More information about the mechanism to handle the DoS/DDoS situation is presented in Section 4.3. A mechanism to address the DDoS situation has also been evaluated.

3.2. Tools Used for Service Implementation and Evaluation

To develop an SDN application designed to meet the objectives described in Section 3.1 and evaluate its performance in a realistic scenario, the following tools were used:

- **Open Network Operating System (ONOS) SDN controller:** An open-source project aimed at creating a software-defined networking operating system for communications service providers that is designed for scalability, high performance, and high availability. This tool is widely used in both academic and industrial environments [31].
- **Mininet-WiFi network emulator:** A software-defined network emulator tool that has the capability to define different network topologies, where nodes/hosts can be configured with multiple wireless interfaces [32]. It can, therefore, reproduce complex rail and road environments by directly injecting real traffic data.
- **Scapy tool:** A packet manipulation tool for computer networks that can forge or decode packets, send them on the wire, capture them, and match requests and replies. It will be used in the implemented system to generate common/emergency messages [33].
- **Matt's traceroute (MTR) tool:** A commonly used tool to measure the latency of the developed network. It uses the ping and traceroute to calculate the latency and jitter in the network [34]. In our context, it will be used to demonstrate the real-life applicability of the proposed solution. Detailed information about these tools is available in [27].

3.3. Considered Scenario: Level Crossing

To demonstrate and emulate the common emergency services for railway and road coexistence scenario, a level-crossing situation has been selected, where railway tracks are perpendicular to roads, as shown in Figure 2. Nevertheless, the proposed mechanism could be applied to any type of shared or dedicated scenario/situation aiming at interconnecting rail and road services. In the considered scenario, railways and roads have shared access networks and both entities have a common backhaul core.

A Python script is written in Mininet-WiFi to develop the selected network topology, which can mimic the level-crossing scenario [23]. In the selected topology, the ONOS [31] SDN controller is used to programmatically control the forwarding element of the network topology. The defined access points and switches are SDN-based devices that operate and control via the OpenFlow13 protocol [35]. The rail emergency server (RailEmer) and car emergency server (CarEmer) are designed to manage the exchange of emergency messages between the server and nodes. Each emergency server has primary and secondary servers, as shown in Figure 2. The secondary servers are included to ensure system redundancy and fail-safety. In addition, clusters of primary and secondary servers can be considered

to enhance system security and prevent system failure. The RailServer and CarServer are defined to manage and exchange the normal messages between the nodes and server.

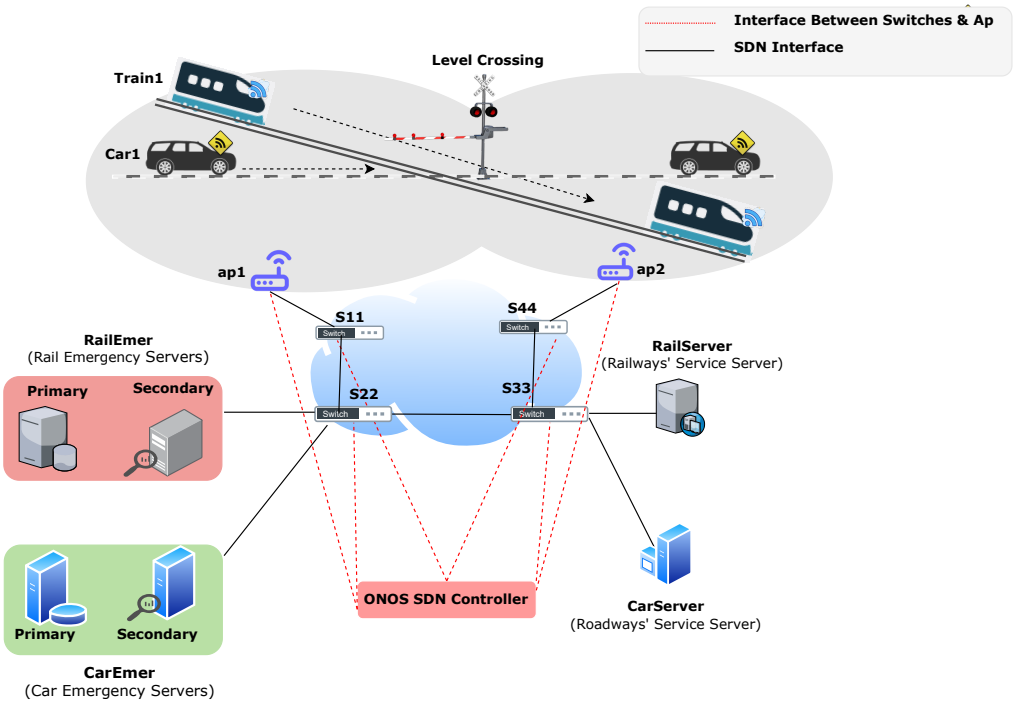

Figure 2. Considered scenario: rail and road coexistence at level crossing.

To implement the common emergency service, network-side emergency service implementation is carried out using SDN and network slicing with virtual local area network (VLAN) tagging. In this empirical work, we are not creating the whole eREC mechanisms nor multipoint calls but only demonstrating how a common emergency service can be implemented and triggered from both domains. The "start-eREC" message can be sent by cars or trains in the specific area of the level crossing. In this implementation, it is ensured that data traffic is differentiated between roads and railways based on its source, destination, VLAN tag, and port number and provides common emergency services for both trains and cars. Detailed information about developed common services to differentiate the data traffic and emergency message delivery is presented in Sections 4.1 and 4.2.

Figure 3 shows the developed network topology, where the nodes Train1, Train2, Car1, Car2, and New node are connected to access point Ap1. The nodes Train3 and Car3 are connected to access point Ap2. Access point Ap1 is connected to switch S11 and access point Ap2 is connected to switch S44. The RailEmer and CarEmer with their primary and secondary servers are connected to switch S22 and RailServer and CarServer are connected to switch S33. Figure 4 shows the Mininet-WiFi-generated graph with all the nodes connected to access points. For simplicity, the demonstration involves three open virtual switches, two access points, and seven nodes (representing cars and trains). However, the developed system is capable of handling more complex prototypes with an increased number of switches, access points, and nodes. The complexity of the prototype depends on the capacity of the system where the SDN-based solution is implemented.

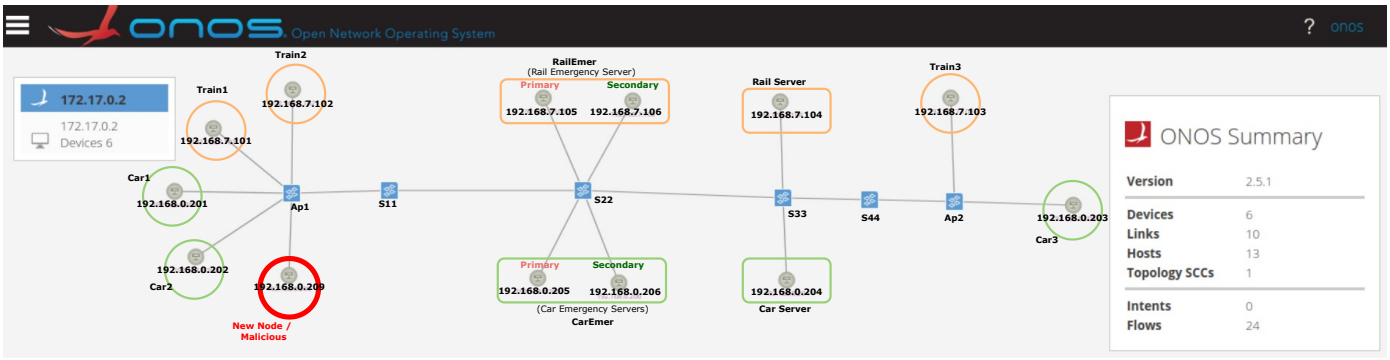

Figure 3. Considered scenario: ONOS perspective.

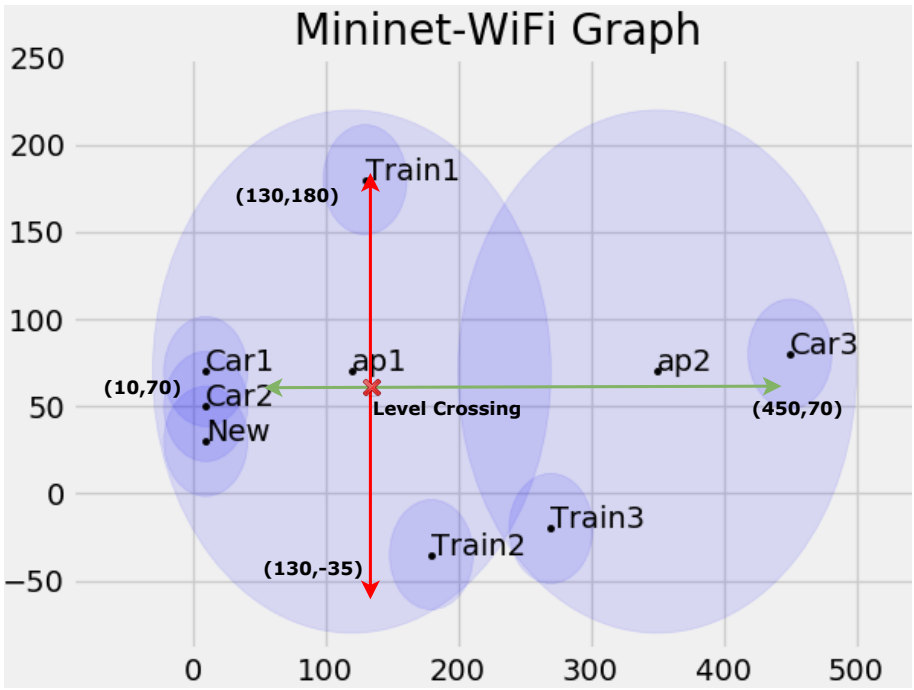

Figure 4. Considered scenario: Mininet-Wifi perspective.

4. Implementation of the Proposed SDN-Based Service for Rail and Road Coexistence

In this section, based on the elements introduced in Section 3 (objectives, tools, scenario), we present the algorithms that we implemented for the secure management of emergency messages.

4.1. Packet Differentiation and Emergency Service Delivery

Figure 5 shows the logical flow diagram of the SDN application that has been implemented to differentiate the data traffic from railways and roads, which is mainly based on VLAN tagging, as described in [27]. It also aims to differentiate emergency data packets/messages and non-emergency data packets/messages and forwarding of emergency messages to the RailEmer and CarEmer servers.

When a User Datagram Protocol (UDP) data packet arrives at any access point/switch, this network device looks into its forwarding rules/table, and if there are no forwarding rules installed at the current access point/switch for the arrived source and destination IP pairs, the data packet is sent to the SDN controller for treatment as an OpenFlow PacketIn message. The developed SDN application checks the destination port assigned in the UDP data packet and if the destination port is equal to 135 (the port defined as the emergency port in this proof of concept), the controller marks this data packet as an emergency data packet and tags this packet with VLAN ID 5 (VLAN associated with emergency here). For

the implementation of the application, any port and any VLAN ID can be used. Here, VLAN ID 5 is used to differentiate the emergency data packet and port 135 is set to receive this data packet via UDP.

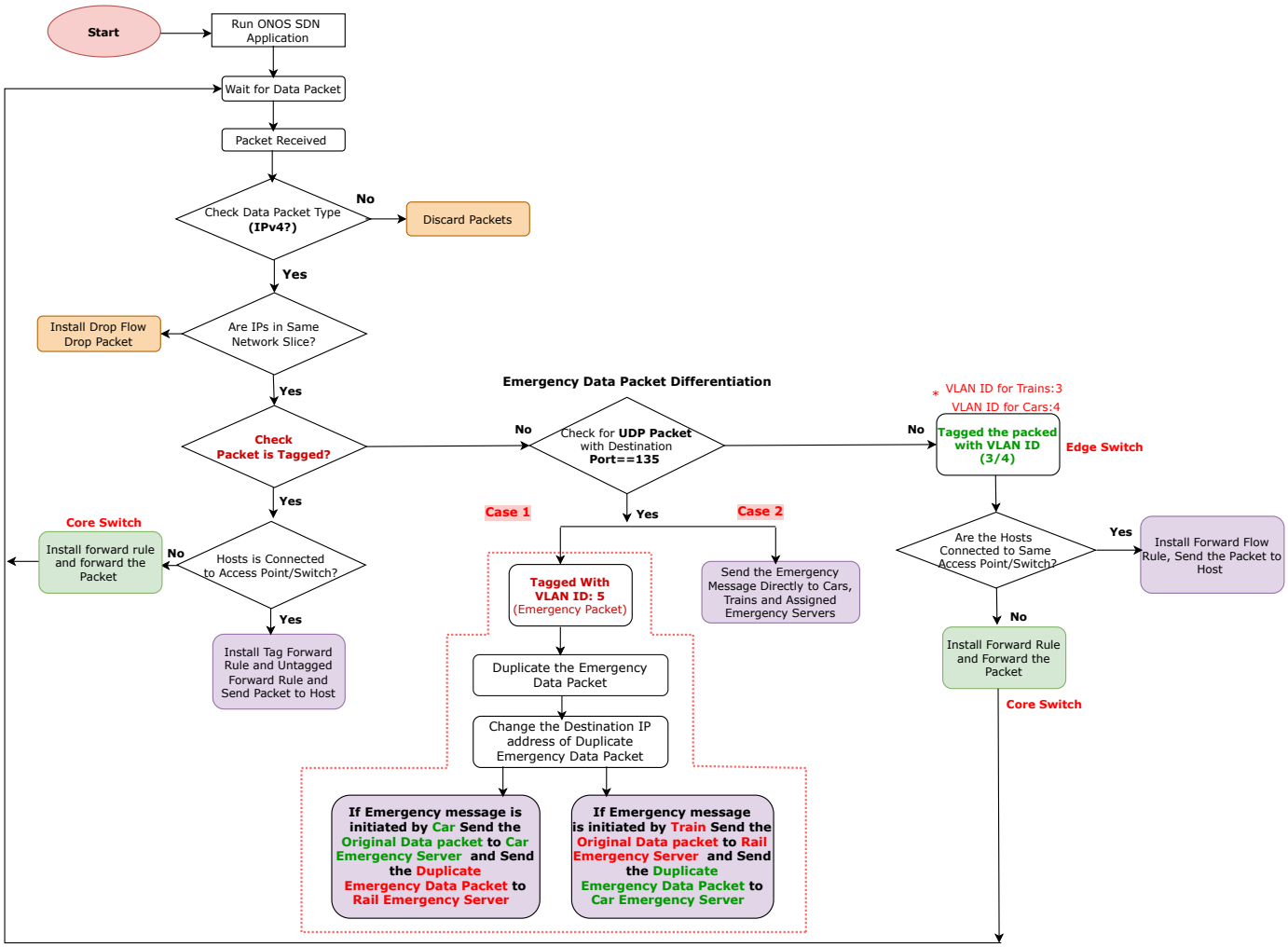

Figure 5. Flowchart.

If the developed SDN application is operating in case 1, the SDN application duplicates the emergency UDP data packet and changes the destination IP address of the duplicated emergency data packet. If the emergency message is initiated by the car, the original emergency data packet is sent to the car emergency server and the duplicated emergency data packet is sent to the rail emergency server. On the other hand, if the emergency message is initiated by a train, the original emergency data packet is sent to the rail emergency server and the duplicated emergency data packet is sent to the car emergency server. If the developed SDN application is operating in case 2, the SDN application distributes the emergency data packet to trains and cars on the secondary wireless interface which is reserved for emergency data communication.

If the incoming UDP data packets have a destination port other than 135, they are designated as non-emergency data packets. VLAN ID 3 is assigned to the data packet to/from the rail, RailServer, and RailEmer server, while VLAN ID 4 is assigned to the data packet to/from the car, CarServer, and CarEmer server. It is worth noting that any VLAN ID can be used to tag the packets for traffic differentiation.

4.2. Emergency Data Packet Duplication

The developed ONOS application ensures that emergency data packets can be differentiated from non-emergency data packets and, when an emergency or critical situation occurs, both the emergency servers, i.e., RailEmer and CarEmer receive the emergency message [23]. For the demonstration purpose, we used the UDP protocol to send and receive emergency and non-emergency data packets. Figure 6 shows the emergency data packet VLAN tagging and duplication mechanism of the emergency data packet. In the given Figure 6, Train1 and Car1 are connected to access point Ap1. Suppose Car1 is stuck at the level crossing or has some issue and Train1 is about to reach the level crossing.

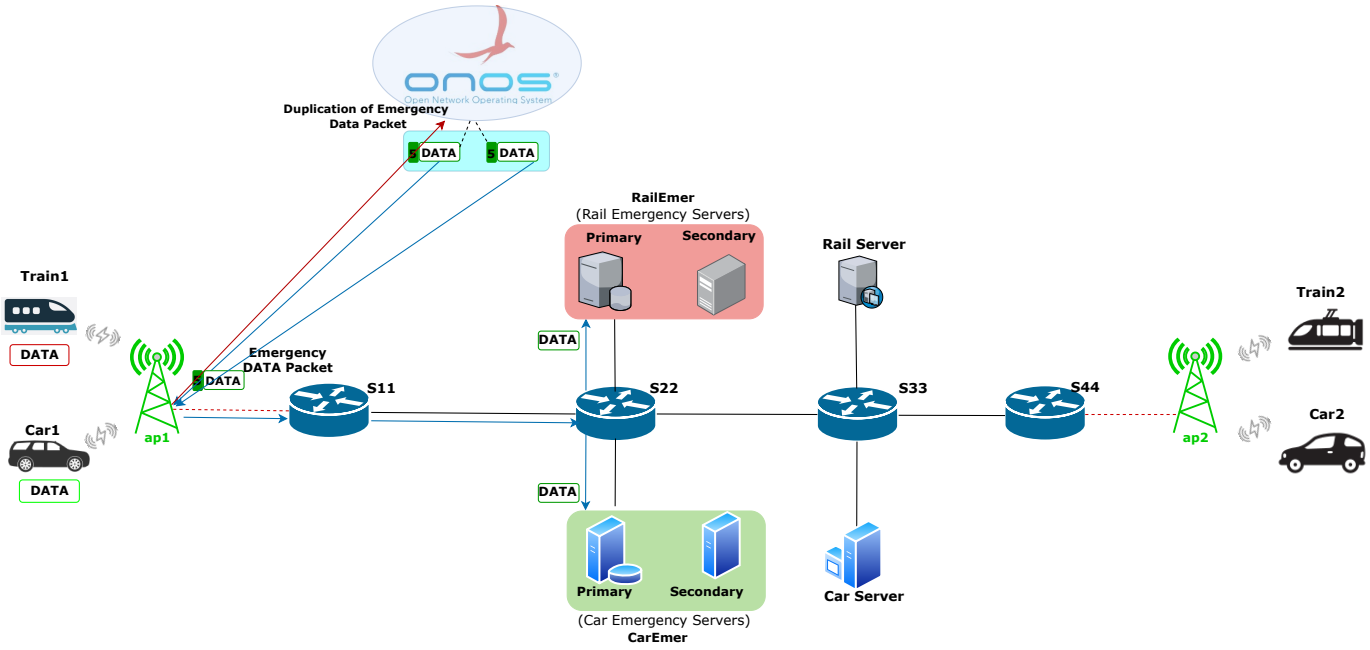

Figure 6. Emergency packet tagging and duplication.

In this considered situation, the emergency data packet is generated by Car1. When this emergency data packet comes to access point Ap1, the SDN application tags this data packet with VLAN ID 5 and forwards it to the SDN controller. The SDN controller duplicates the emergency data packet and sends the original data packet to access point Ap1. Along with these, the controller installed the forwarding rules using the OpenFlow13 protocol at the access point Ap1, switch S11, and switch S22. In this selected scenario the original emergency data packet is sent to the car emergency server (CarEmer) since the emergency message is initiated by Car1. The SDN application replaces the destination IP and MAC address of the duplicate emergency data packet with the IP and MAC address of the rail emergency server (RailEmer) and sends this duplicate emergency data packet to RailEmer. In Figure 6, it can be seen that an emergency data packet is duplicated and distributed to both emergency servers. In this manner, the SDN-based application informs the emergency servers of both domains about the critical situation in the railway and road coexistence scenario.

4.3. Security Improvement: DoS/DDoS Mitigation

Some of the most famous DDoS attacks include the one on Google in 2017, which was the largest to date, and the one on AWS in 2020. GitHub also experienced a significant attack in 2018. In 2016, an attack on Dyn disrupted many major sites, and in 2013, an attack on Spamhaus slowed down the entire internet [36]. These attacks have led to a greater focus on improving network security.

In the context of DDoS attack mitigation, blackholing and sinkholing are two commonly employed strategies. Blackholing is a technique that diverts all traffic towards a specific IP or IP range to a null route or 'black hole' during a DDoS attack. While this method effectively blocks malicious traffic from reaching its intended target, it may unintentionally impact legitimate traffic as well. Sinkholing, conversely, filters out malicious traffic while allowing legitimate traffic to pass through. It uses a decoy, or 'sinkhole' IP address, to identify malicious nodes without disrupting traffic from legitimate sources. This method primarily targets IP addresses or domains associated with known threats. However, it may not be effective against new or previously unidentified malicious nodes. Moreover, establishing and maintaining a sinkhole infrastructure for traffic diversion and analysis can be resource-intensive. Sinkholing is often implemented at the DNS level, i.e., application layer. Despite their effectiveness, both techniques have certain limitations [37].

To address these challenges, a mechanism has been integrated into the developed SDN application, as depicted in Figure 7. This mechanism operates at the second layer of the OSI model, enabling early detection and blocking of malicious traffic. It functions in both proactive and reactive modes, as every emergency data packet is routed through the SDN controller. The application is designed to detect both new and previously unidentified malicious nodes. This approach ensures a more robust defense against DDoS threats.

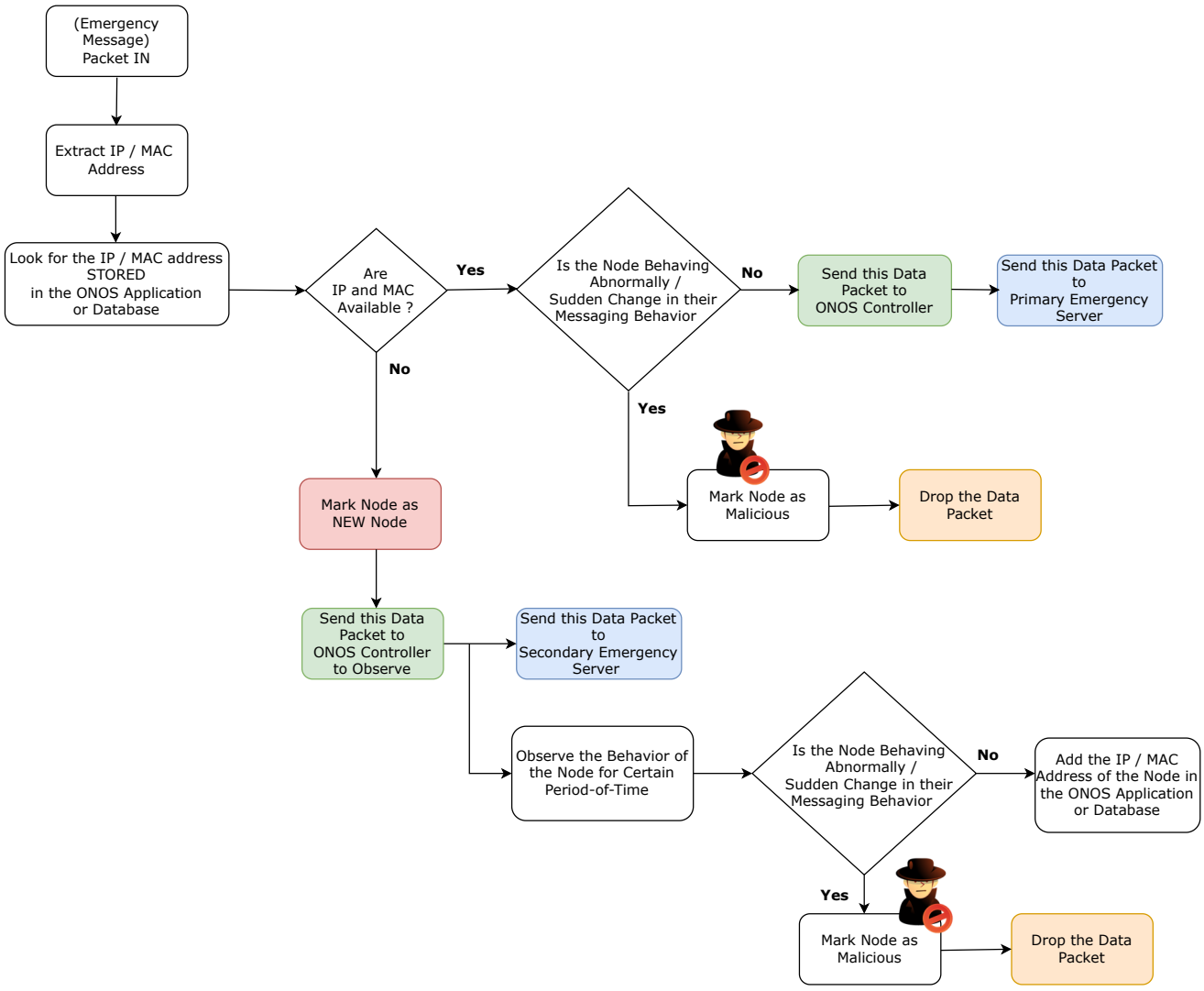

Figure 7. Flowchart: reducing DDoS attacks.

This mechanism works by first differentiating emergency messages from common ones and extracting the IP and MAC addresses of the source node initiating the emergency message. The application then checks these addresses against those in the SDN application/database. If a match is found, the node is authenticated. Every emergency message is first sent to the SDN controller, which monitors the node's behavior. If any abnormal behavior is detected, such as changes in messaging behavior or attempts to send more messages than allowed, the node is marked as malicious, and all its data packets are dropped. If no abnormalities are detected, the emergency packet is forwarded to the ONOS controller, which then sends it to the primary emergency server.

If the extracted IP and MAC are not found in the ONOS application/database, the node is marked as a new node and the node is considered new. The packet is first sent to the ONOS controller for observation and then forwarded to the secondary emergency server. In the meantime, the system continues to observe the node's messaging behavior for a certain period to monitor for any changes in behavior that might indicate malicious intent. If abnormal behavior is detected during the observation period, the node is classified as malicious and all the data packets to/from the new node are dropped and an alert is sent to the controller to block the node. If no abnormal behavior is observed during the observation period, the IP and MAC address of the new node are added to the ONOS application/database and the node is considered ethical and the data packet is processed normally. After adding the node's information into the ONOS application/database, the new node is served by the primary emergency server. This process ensures the continuous operation and security of the developed architecture, providing a robust defense for reducing potential DDoS attacks.

Overall, the given flowchart in Figure 7 outlines a security mechanism for identifying and isolating potentially malicious nodes on a network. It leverages information about the node's IP and MAC address, its past behavior, and its messaging activity during a monitoring period to make decisions about how to handle incoming data packets. It is important to note that the specific details of this process, such as the criteria for abnormal behavior detection and the duration of the observation period, may vary depending on the specific implementation and security requirements of the network.

5. Validation of Developed Application for Common Emergency Services

This section presents the validation of the developed SDN-based application for common emergency services for railway and road coexistence scenarios. Emergency and non-emergency messages are sent for demonstration purposes from Car1 to CarEmer and from Train1 to RailEmer. For validation purposes, these data packets were captured in Wireshark [38].

5.1. Analysis of Data Packet in Case 1

In case 1, we conducted tests to validate the developed SDN application. Firstly, an emergency message was sent from Car1 to CarEmer, and the data packet was captured at the CarEmer and RailEmer nodes using Wireshark. Additionally, to measure latency, 500 UDP data packets were sent from Car1 to CarEmer on port 135. The average latency observed was 30.9 ms, as shown in Figure 8.

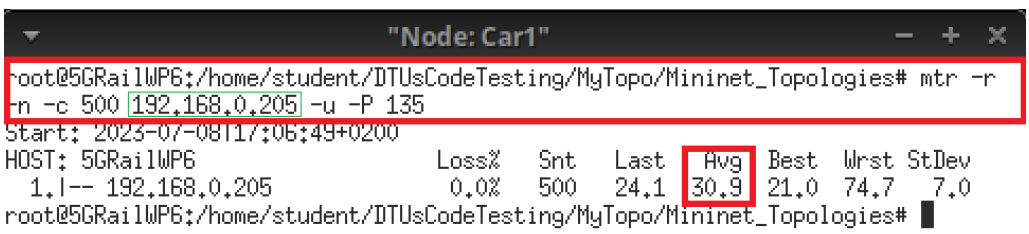

Figure 8. Latency test between Car1 and CarEmer.

After receiving the emergency data packet, CarEmer forwarded it to Car2 and Car3, resulting in an average latency of 7.5 ms, as shown in Figure 9. In this case, the end-to-end latency was (30.9+7.5) 38.4 ms. Similarly, 500 UDP data packets were sent from Train1 to RailEmer, and RailEmer to Train2 and Train3. The average end-to-end latency for this test was 36.9 ms.

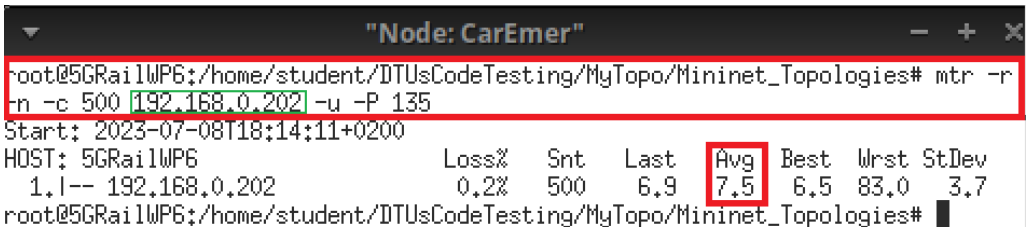

Figure 9. Latency test Between CarEmer and Car2.

In addition to these tests, non-emergency data packets were also sent between cars and trains and their respective servers. The results of these tests are presented in Table 1. From Table 1, it can be concluded that during non-emergency data transmission, cars are only able to communicate with other cars and their assigned servers, while trains can only communicate with other trains and their assigned servers. This distinction ensures the appropriate differentiation of data traffic in scenarios where both road and railway transportation coexist. However, during an emergency situation, the emergency data packets were shared with every node in the network.

Table 1. Data differentiation and distribution of emergency data packet.

	Non-Emergency Message (VLAN 3 or VLAN 4)				Emergency Message VLAN 5			
	Trains	**Cars**	**Train Servers (Emergency & Service)**	**Car Servers (Emergency & Service)**	**Trains**	**Cars**	**Train Servers (Emergency & Service)**	**Car Servers (Emergency & Service)**
trains	✓	x	✓	x	✓	✓	✓	✓
cars	x	✓	x	✓	✓	✓	✓	✓

5.2. Analysis of Data Packet in Case 2

In this scenario, Car1 directly sends emergency data to every available car and train at the location where the emergency situation occurs. This communication is facilitated through a second wireless link specifically dedicated to emergency data transmission. For the purpose of demonstration, Car1 sends 500 UDP data packets to Train1, and the measured average end-to-end latency is 7.5 ms, as shown in Figure 10.

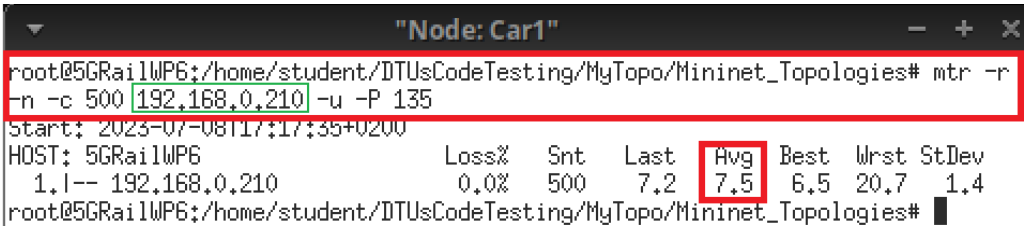

Figure 10. Latency test between Car1 and Train1 using dedicated wireless interface.

In comparison to case 2, case 1 exhibits higher end-to-end latency. This disparity arises due to the process followed in case 1, as explained in Section 4.2. Initially, all emergency packets are transmitted to the SDN controller. Upon receiving the emergency data packets, the controller duplicates them and sends packets to the designated emergency servers, namely, CarEmer and RailEmer. Subsequently, the emergency servers disseminate the emergency packets to all the nodes present at the specified location.

6. Validation of the Developed Application to Avoid DDoS

This section validates the developed SDN application to handle a DDoS attack. To validate the developed application to tackle a DDoS attack, the scenario presented in Figure 2 is considered. For the demonstration purpose, one new node acts as a malicious node and is attached to access point Ap1, as presented in Figure 11. The information (IP and MAC) of all the nodes presented in this figure is saved in the ONOS application/database, only the information of the new node is not available in the ONOS application/database.

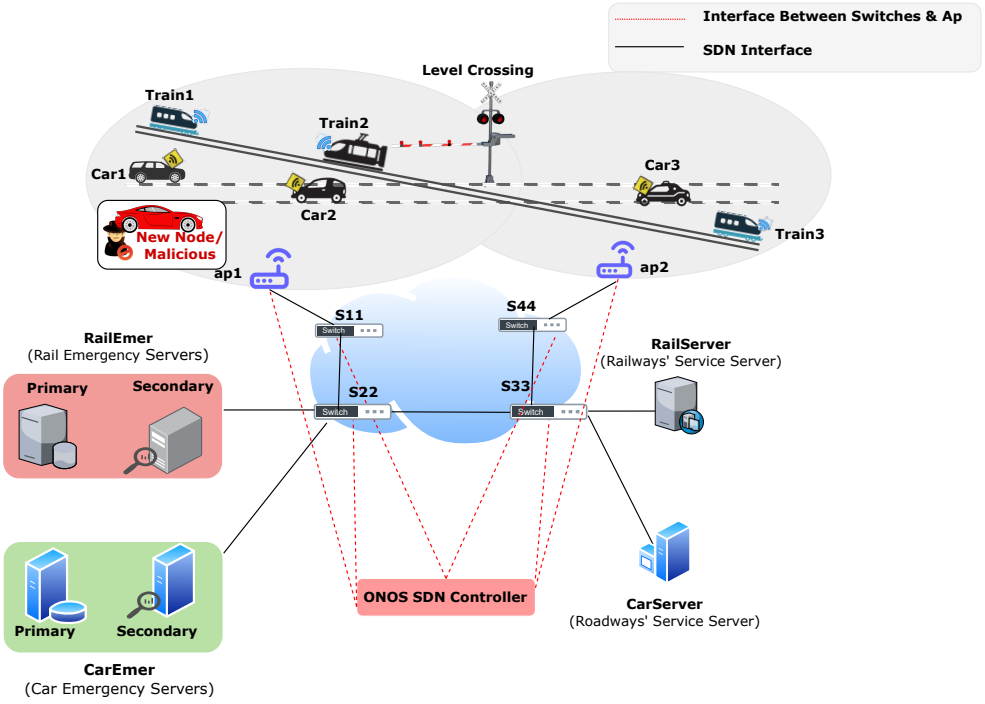

Figure 11. Considered scenario: shared access network and shared core; track perpendicular to road.

6.1. When an Authenticated Node Sends Emergency Messages

Figure 12 shows the emergency message transfer by the node Car1 to the emergency server. As Car1 is an authenticated node, the packet is sent to the controller. After that, it follows the steps described in Section 4.2 and sends the data to the primary servers. For validation purposes, a limit is set on the number of emergency messages that can be sent by the nodes: a maximum of ten messages per minute or six hundred messages per hour. The intervals between the sending of emergency messages are tracked by the controller. If this limit is exceeded by a node, this behavior is observed by the controller and the packets start being dropped. As shown in Figure 13, as the authenticated node sends the eleventh packet, exceeding the allowed limit, the SDN application tracks this abnormality of the node and starts dropping the packets.

```
10:01:38.891 INFO [AppComponent] Legit Node, Send this Packet to Primary Server:: 1,192.168.0.201,192.168.0.205
10:01:43.985 INFO [AppComponent] Legit Node, Send this Packet to Primary Server:: 2,192.168.0.201,192.168.0.205
10:01:49.053 INFO [AppComponent] Legit Node, Send this Packet to Primary Server:: 3,192.168.0.201,192.168.0.205
10:01:54.084 INFO [AppComponent] Legit Node, Send this Packet to Primary Server:: 4,192.168.0.201,192.168.0.205
10:01:59.156 INFO [AppComponent] Legit Node, Send this Packet to Primary Server:: 5,192.168.0.201,192.168.0.205
10:02:04.208 INFO [AppComponent] Legit Node, Send this Packet to Primary Server:: 6,192.168.0.201,192.168.0.205
10:02:09.269 INFO [AppComponent] Legit Node, Send this Packet to Primary Server:: 7,192.168.0.201,192.168.0.205
10:02:14.332 INFO [AppComponent] Legit Node, Send this Packet to Primary Server:: 8,192.168.0.201,192.168.0.205
10:02:19.397 INFO [AppComponent] Legit Node, Send this Packet to Primary Server:: 9,192.168.0.201,192.168.0.205
10:02:24.442 INFO [AppComponent] Legit Node, Send this Packet to Primary Server:: 10,192.168.0.201,192.168.0.205
```

Time Stamps | Authenticated Node is Served by Primary Server | Msg Count | Source/Node IP | Primary Server IP

Figure 12. Serving authenticated node: screenshot of developed SDN application.

From Figure 12, it can be inferred that when Car1 sends ten emergency messages per minute, it is successfully sent to the primary emergency servers. However, as soon as the allowed limit is reached, the application begins to drop the data packets.

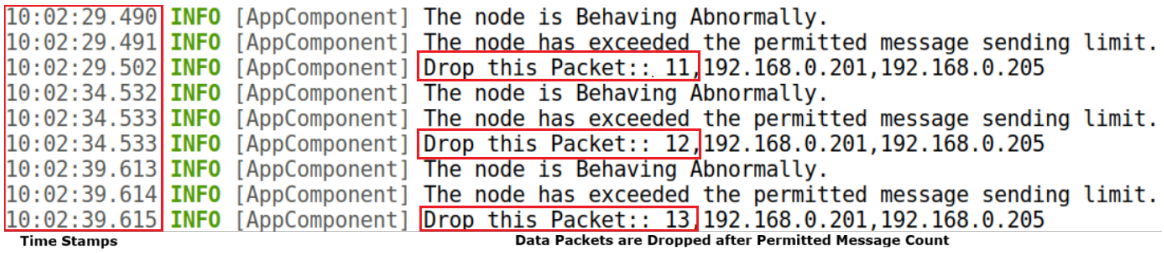

Figure 13. Dropping data packets.

6.2. When a New/Malicious Node Sends Emergency Messages

When a new node sends emergency messages, the data packet is sent to the controller and secondary servers, as shown in Figure 14. While the new node sends messages within the allowed limit it is served by the secondary server. The secondary server initially handles the new node, monitoring its messaging behavior for a specified duration. In the context of this application, the new node's behavior is observed for one hour. It is important to note that this observation period is merely for demonstration purposes within this application, and the actual duration would depend upon the specific application and its functional requirements. Following the observation period, if the new node's behavior is deemed acceptable, the IP and MAC addresses of the new node are incorporated into the ONOS application/database. This action authenticates the new host.

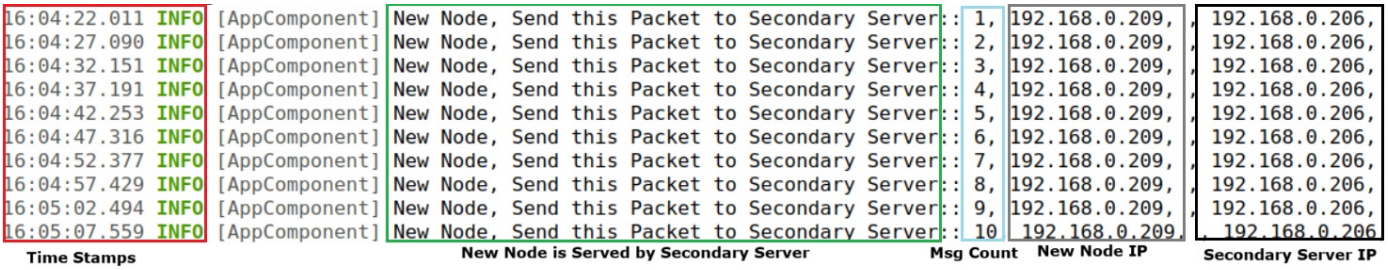

Figure 14. Dropping data packets.

Suppose the new node is malicious and it starts flooding the system by sending emergency messages, in this case, the ONOS application marks the new node as malicious, drops all the packets, and blocks it, as shown in Figure 15.

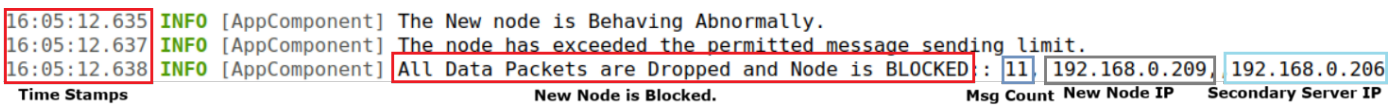

Figure 15. Dropping data packets.

7. Conclusions

In this paper, we have proposed an SDN-based solution for common emergency service for railway and road coexistence scenarios. To the best of our knowledge, such a solution had never been proposed before and its open-source implementation is available online [23]. Moreover, we aimed to emulate and validate the developed SDN-based application through a realistic level-crossing scenario and different evaluations, involving sending emergency messages and non-emergency data packets between cars, trains, and their respective emergency servers.

To validate developed common emergency services two cases were considered. Case 1 is beneficial in the scenario when no cross-communication is available between the road

and railway domains. In this case, emergency data are first sent to emergency servers, and emergency servers distribute to all the nodes with the help of the SDN controller. In this case, the end-to-end average latency was in the range of 36.9 to 38.4 ms. In case 2, the emergency data are directly shared with cars and trains, leveraging the dedicated wireless interface to send and receive the emergency messages. In this case, the average end-to-end latency was 7.5 ms. This facilitates efficient and timely transmission of critical information during emergencies. Case 1 has higher latency compared to case 2, this is due to the additional steps involved in case 1, where the emergency packets were first sent to the SDN controller, duplicated, and then forwarded to the assigned emergency servers before being distributed to all nodes at the location.

Additionally, tests with non-emergency data packets revealed that cars could only communicate with other cars and their assigned servers, while trains could communicate with other trains and their assigned servers. This shows that the developed SDN application is capable of differentiating the data between the road and railway domains. However, during an emergency situation, the emergency data packets were shared with every node in the network.

To minimize the risk of DoS/DDoS situations, and improve security, the SDN-based application checks the sender's IP and MAC addresses with the SDN database and authenticates if a match is found. The node's behavior is monitored for any abnormalities like excessive messaging. If found, the node is marked as malicious and its packets are dropped. Otherwise, the emergency packet is forwarded to the primary server. If a node's IP and MAC are not available in the ONOS database, it is marked as new. Its packet is observed by the ONOS controller and sent to a secondary server. The system watches the node's behavior. If the new node behaves maliciously, such as by flooding the system with emergency messages, the ONOS application identifies this harmful behavior: its packets are dropped, and it is blocked. If it behaves normally and sends messages as per the allowed limit, it is added to the ONOS database, and its packets are processed as usual.

In summary, the proposed SDN-based solution leverages the capabilities of SDN, such as centralized network control, precise traffic routing, and advanced data traffic differentiation based on VLAN tagging. This solution effectively differentiates data traffic and mitigates the risk of DoS/DDoS attacks. Future work includes developing edge emergency servers based on node geolocation. Concurrently, additional security aspects will be explored, and functionalities will be enhanced to ensure the delivery of emergency data packets during busy hours or network congestion.

Author Contributions: Conceptualization, R.S., J.S., T.S., L.M., M.S.B. and M.B.; methodology, R.S., J.S., L.M. and M.S.B.; software, R.S. and J.S.; validation, R.S., J.S., L.M., T.S., M.S.B. and M.B.; formal analysis, R.S., J.S., L.M. and M.S.B.; investigation, R.S., L.M. and J.S.; resources, M.S.B., L.D. and R.S.; data curation, R.S., M.S.B. and L.M.; writing—original draft preparation, R.S., L.M., J.S. and M.S.B.; writing—review and editing, J.S., R.S., L.M., J.S. and M.S.B.; visualization, R.S.; supervision, M.S.B., L.D. and J.S.; project administration, J.S., M.S.B., L.D., L.M. and M.B.; funding acquisition, J.S., L.D., L.M. and M.B. All authors have read and agreed to the published version of the manuscript.

Funding: This empirical study was conducted as part of the '5G for Future RAILway Mobile Communication System' (5GRAIL) project. The project receives funding from the European Union's Horizon 2020 research and innovation program, with grant agreement number 951725.

Data Availability Statement: The data presented in this study are available in this article.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Kostrzewski, M.; Eliwa, A.; Dawood, A. Autonomy of urban LIGHT rail transport systems and its influence on users, expenditures, and operational costs. *Transp. Probl. Int. Sci. J.* **2022**, *17*, 165–175. [CrossRef]
2. Kohda, T.; Fujihara, H. Risk analysis of level crossing accidents based on systems control for safety. *Proc. Inst. Mech. Eng. Part O J. Risk Reliab.* **2008**, *222*, 419–429. [CrossRef]

3. ERTMS Provides the European Union with a Unique Opportunity to Create a Seamless Railway System. Available online: https://www.ertms.net/ (accessed on 1 February 2024).
4. The Railway System for Mobile Communication. Available online: https://uic.org/rail-system/gsm-r/ (accessed on 1 March 2023).
5. Enhanced Railway Emergency Call Specification. Available online: https://uic.org/IMG/pdf/erec_specification_o-3151-2.0.0.pdf (accessed on 1 February 2024).
6. Santa, J.; Pereñíguez, F.; Moragón, A.; Skarmeta, A.F. Experimental evaluation of CAM and DENM messaging services in vehicular communications. *Transp. Res. Part C Emerg. Technol.* **2014**, *46*, 98–120. [CrossRef]
7. Sniady, A. Lecture on GSM-R. DTU's Course 34345-Signalling Systems and Technology for Railways. Available online: https://lifelonglearning.dtu.dk/en/electro/single-course/signalling-systems-and-technology-for-railways/ (accessed on 10 March 2024).
8. RAIL INCIDENTS: Guidance to the Emergency Services for Access to the Railway Infrastructure. Available online: https://safety.networkrail.co.uk/wp-content/uploads/2022/09/Rail-Incidents_Guidance-to-the-Emergency-Services-for-Access-to-the-Railway-Infrastructure.pdf (accessed on 1 February 2024).
9. Infrastructure Sharing in Broadband Networks: Impact on Telecommunications Operators and Consumers. Available online: https://www.frontiersin.org/research-topics/36510/infrastructure-sharing-in-broadband-networks-impact-on-telecommunications-operators-and-consumers/m (accessed on 1 February 2024).
10. Cross-Sectoral Infrastructure Sharing for Broadband. Available online: https://repository.unescap.org/bitstream/handle/ (accessed on 1 February 2024).
11. Future Railway Mobile Communication System. Available online: https://uic.org/rail-system/frmcs/ (accessed on 1 February 2024).
12. Ahmad, S.; Mir, A.H. Scalability, Consistency, Reliability and Security in SDN Controllers: A Survey of Diverse SDN Controllers. *J. Netw. Syst. Manag.* **2021**, *29*, 9. [CrossRef]
13. Mao, J.; Chen, L.; Li, J.; Ge, Y. Controller Backup and Replication for Reliable Multi-domain SDN. *Ksii Trans. Internet Inf. Syst.* **2020**, *14*, 4725–4747. [CrossRef]
14. Yu, T.; Hong, Y.; Cui, H.; Jiang, H. A survey of Multi-controllers Consistency on SDN. In Proceedings of the 2018 4th International Conference on Universal Village (UV), Boston, MA, USA, 21–24 October 2018; pp. 1–6. [CrossRef]
15. Hu, T.; Guo, Z.; Yi, P.; Baker, T.; Lan, J. Multi-controller Based Software-Defined Networking: A Survey. *IEEE Access* **2018**, *6*, 15980–15996. [CrossRef]
16. Ghazi, M.; Khattak, M.; Shabir, B.; Malik, A.; Ramzan, M. Emergency message dissemination in vehicular networks: A review. *IEEE Access* **2020**, *8*, 38606–38621. [CrossRef]
17. Nikbakht Bideh, P.; Paladi, N.; Hell, M. Software-Defined Networking for Emergency Traffic Management in Smart Cities. In *Vehicular Ad-Hoc Networks For Smart Cities: Third International Workshop, 2019*; 2020; pp. 59–70. Available online: https://link.springer.com/chapter/10.1007/978-981-15-3750-9_5 (accessed on 10 March 2024).
18. Zaballa, E. 4 SDN-Based Slicing and Network Resource Distribution in Train-to-Ground Railway Communication. In *Reports of the DLR-Institute of Transportation Systems Volume 38*; 2021; p. 27. Available online: https://www.dlr.de/fs/Portaldata/16/Resources/dokumente/berichtsreihe/Volume_38_3rd_SmartRaCon_Scientific_Seminar_2021.pdf#page=36 (accessed on 10 March 2024).
19. Hassan, M.; Gregory, M.; Li, S. Multi-Domain Federation utilising Software Defined Networking: A Review. *IEEE Access*. **2023**, *11*, 19202–19227. [CrossRef]
20. Sultana, R.; Grover, J.; Tripathi, M. Security of SDN-based vehicular ad hoc networks: State-of-the-art and challenges. *Veh. Commun.* **2021**, *27*, 100284. [CrossRef]
21. Akhunzada, A.; Gani, A.; Anuar, N.B.; Abdelaziz, A.; Khan, M.K.; Hayat, A.; Khan, S.U. Secure and dependable software defined networks. *J. Netw. Comput. Appl.* **2016**, *61*, 199–221. [CrossRef]
22. Ali, S.T.; Sivaraman, V.; Radford, A.; Jha, S. A survey of securing networks using software defined networking. *IEEE Trans. Reliab.* **2015**, *64*, 1086–1097. [CrossRef]
23. DTU5GRail, 5GRail_WP6. Available online: https://github.com/DTU5GRail/5GRail_WP6/tree/main/DTU_Code (accessed on 1 February 2024).
24. Mendiboure, L.; Chalouf, M.; Krief, F. Load-aware and mobility-aware flow rules management in software defined vehicular access networks. *IEEE Access* **2020**, *8*, 167411–167424. [CrossRef]
25. Hussein, D.; Askar, S. Federated Learning Enabled SDN for Routing Emergency Safety Messages (ESMs) in IoV under 5G Environment. *IEEE Access* **2023**, *11*, 41723–141739. [CrossRef]
26. Abbas, M.; Muhammad, A.; Song, W. SD-IoV: SDN enabled routing for internet of vehicles in road-aware approach. *J. Ambient. Intell. Humaniz. Comput.* **2020**, *11*, 1265–1280. [CrossRef]
27. Singh, R.; Soler, J.; Sylla, T.; Mendiboure, L.; Berbineau, M. Coexistence of Railway and Road Services by Sharing Telecommunication Infrastructure Using SDN-Based Slicing: A Tutorial. *Network* **2022**, *2*, 670–706. [CrossRef]
28. What Is a DDoS Attack? Available online: https://www.cloudflare.com/learning/ddos/what-is-a-ddos-attack/ (accessed on 1 February 2024).

29. Eliyan, L.; Di Pietro, R. DeMi: A Solution to Detect and Mitigate DoS Attacks in SDN. *IEEE Access* **2023**, *11*, 82477–82495. [CrossRef]
30. Dridi, L.; Zhani, M. SDN-guard: DoS attacks mitigation in SDN networks. In Proceedings of the 2016 5th IEEE International Conference on Cloud Networking (Cloudnet), Pisa, Italy, 3–6 October 2016; pp. 212–217.
31. Open Network Operating System (ONOS®). Available online: https://opennetworking.org/onos/ (accessed on 1 February 2024).
32. Mininet-WiFi Emulator for Software Defined Network. Available online: https://mininet-wifi.github.io/ (accessed on 1 February 2024).
33. Introduction to Scapy? Available online: https://santandergto.com/en/guide-using-scapy-with-python/ (accessed on 1 February 2024).
34. What Is MTR and Why Is It Useful? Available online: https://www.comparitech.com/net-admin/what-is-mtr/ (accessed on 1 January 2024).
35. OpenFlow Switch Specification. Available online: https://opennetworking.org/wp-content/uploads/2014/10/openflow-spec-v1.3.0.pdf (accessed on 1 February 2024).
36. Famous DDoS Attacks | The Largest DDoS Attacks of All Time. Available online: https://www.cloudflare.com/learning/ddos/famous-ddos-attacks/ (accessed on 1 March 2024).
37. Blackholing. Available online: https://www.imperva.com/learn/ddos/blackholing/ (accessed on 1 March 2024).
38. Wireshark. Available online: https://www.wireshark.org/ (accessed on 1 February 2024).