An edge-based smart network monitoring system for the Internet of Vehicles

Pedro Manuel Augusto Ferreira
Instituto Superior Técnico, Universidade de Lisboa, Lisboa, Portugal
Email: pedro.m.a.ferreira@tecnico.ulisboa.pt

January 2021

***Abstract*—Internet of Vehicles is the future of transportation. It will be present everywhere and will have a huge impact on our lives. Notwithstanding, there are plenty of aspects to take into consideration while studying these networks, such as: data dissemination, cybersecurity threats and vulnerabilities. For the network to work efficiently, data has to be able to spread through the network efficiently, therefore, to tackle the data dissemination problem, a cluster-based routing algorithm was developed, R-privo. Which, uses a machine learning clustering algorithm as well as a routing algorithm based on social relationships between nodes. The R-privo, as shown by its performance in simulations, obtains high delivery rates with low overhead. Besides, due to the amount of data needed for the IoV, it also requires a good cooperation between nodes. In this sense, a misbehaving node might have a huge impact on network performance. In light of the above, a deep learning based monitoring system that is capable of detecting anomalies in the network and identify known misbehaviours was implemented. The monitoring algorithm is capable of identifying the sybil attack and the id spoofing attack with a high success rate, 68% and 95% for the sybil respectively. Besides, the algorithm was also built capable of detecting other misbehaviours without labelling them.**

Keywords: Internet of Vehicles, Machine Learning, Clustering, Deep Learning, Network Monitoring

I. Introduction

The Internet of Vehicles (IoV) aims to fully use the information and communication technologies for achieving the coordinated development of human, vehicle, and environment, which can alleviate traffic congestion, enhance transportation efficiency, and existing road capacity, [1]. For the IoV to work at its fully potential, a large amount of data has to be able to spread throughout the network. In addition, such amount of data leads to numerous cybersecurity threats and vulnerabilities. Such threats can be from merely data breaches where the attacker will gather data from other nodes to threats that can impact the environment itself. To detect these attackers an Intrusion Detection System (IDS), a system to detect any anomaly or intrusion in a system, can be deployed to the network. Due to the amount of data and possible vulnerabilities most systems are exposed, nowadays, many IDSs use machine learning approaches.

For the IoV to be used, the security of all data, nodes and users has to be ensured. Therefore, a monitoring system must manage all the data about vehicles, traffic, routes, signs, car crashes and people who interact with the network.

Besides, this systems has to be flexible enough to work everywhere and adapt to every city, village or any other place it might be deployed. It has to learn to accomplish all goals it was designed to achieve. Therefore, due to the amount of data in the networks, a Deep Learning (DL) algorithm will be used. In addition, such algorithm will require a lot of computer power, notwithstanding, in such network, the response time is also of utmost importance. Hence, to bring part of the computer and storage power closer to the end users. The algorithm will be deployed in the edge layer of the IoV.

Nevertheless, there are some aspects to take into consideration. What if the algorithm stops working? What if the algorithm does not share with the vehicles the correct data? What if some nodes act in a way that may harm the others? This may occur if the network is attacked. Therefore, such algorithm should also be able to detect and act to eliminate these threats.

On the light of the above, three objectives were outlines to this work, the design and implementation of a clustering algorithm stable in dynamic networks, design and implementation of a routing algorithm complying with the paradigm of the IoV and being capable of propagating data throughout the network, and, develop a deep learning monitoring system on the edge layer to detect any anomaly and classify behaviours in the network, hence, helping mitigate the impact of misbehaviour nodes.

II. State of the art

Internet of Vehicles (IoV) is becoming the next transformation in the world of transportation. Its main goal is safety, comfort and prompt delivery of its occupants with minimum impact on the environment. With this goal in mind, there are several applications for this technology, such as: managing network traffic; reducing of traffic jams; alert the users about any hazard; in case of an accident, call for specific help and send information about the victims. The only way these goals can be achieved is through communication between vehicles, pedestrians, Road Side Unit (RSU)s, and public networks. This huge amount of data leads to some challenges on security and privacy of users and their data.

Most of the challenges in the IoV have also appeared in other fields of study such as the Internet of Things (IoT), with numerous similarities between the two areas. The same

architecture schema is being used to model both IoV and IoT, splitting them into tree layers; Vehicles (Things), Edge and Cloud.

The vehicles layer is mainly responsible for data collection and actuation to control the physical world. Most devices in this layer are resource constrained in terms of computational power, storage, and energy. The edge layer is introduced to help end devices. First, computation intensive tasks can be offloaded to edge devices. Second, the edge layer can mask communication heterogeneity among end devices and connect them to the Internet. Third, edge devices help manage end devices. At last, the cloud layer is utilised not only to store, process, and analysed the collected data but also to provide the additional support needed by many applications. [2]

One of the main differences between IoV and IoT is the existence of RSUs, which are stationary units with much more computation capacity than the vehicles and act as intermediary link between the vehicles and the cloud. They are one of the main components of the edge layers.

To achieve a IoV fully operational, an enormous amount of data has to be constantly shared between the different types of nodes, therefore, a routing algorithm has to be deployed to the network. Most traditional routing algorithms work based on the property of shortest path to destination, however, in IoV most nodes are mobile, which means that the algorithm has to be able to exchange data between mobile nodes and has to be delay tolerant. Examples of algorithms used on delay tolerant networks are the traditionals epidemic [3], the prophet [4] and the bubblerap [5]. There is also the Privo algorithm, an efficient PRIvacy-preserVing Opportunistic routing protocol for Vehicular Delay-Tolerant Networks. This protocol models a Delay Tolerant Network as a time-varying neighbouring graph where the edges correspond to the neighbouring relationship among pairs of nodes, the weight of an edge indicates the strength of a relation at a given time.

When monitoring networks, the most common method is to develop an Intrusion Detection System. These systems can be divided into two areas, the anomaly detection systems, which detect deviations to the normal behaviour of the network, and misuse detection systems, which identify specific behaviours. Traditionally, anomaly detection cannot identify any behaviour and a misuse detection system cannot detect any anomaly without previous knowledge of. Therefore, the hybrid intrusion detection systems were created, to merge the two and take advantages of the best parts of each one. Several IDS are created with machine learning algorithm with the goal automatically inferring and generalising a learning model from sample data. One of the main approaches is through Neural Networks, which, with the right architecture will create an image of the raw data that will be transformed into a representation at a higher and more abstract level, which is known as Deep Learning.

In addition, the authors of [2], present EdgeSec, a novel security service for the IoT that with a few changes can be deployed to the IoV. The EdgeSec is also deployed on the edge layer, that has much more resources than the things layer, in addition, it is much closer to the nodes than the cloud. Besides, the edge layer has more information than the end devices about the network as a whole and it keeps a stable relation with the nodes, which can be beneficial to establish trust between nodes [2].

In order to design and implement an edge-based smart network monitoring system for the Internet of Vehicle, it is necessary to model this network and all the case studies necessary to develop such system. Considering that vehicles and roads are not yet equipped with devices capable of testing these networks, the ONE simulator was used [6]. This simulator was developed for Delay Tolerant Networks and some modifications were required. The most significant changes were the addition of the edge layer where some of the most costly tasks are supposed to run; the implementations of RSUs; the implementation of a new routing algorithm based on privo [7]; and the addition of the deep learning algorithm.

III. Design and Implementation

This section provides a detailed description and discussion of everything that was implemented in this work. It will start with the network architecture. Then, clustering and routing algorithms will be explained and analysed. At last, the monitoring of the network will be discussed.

A. Network Architecture

1) System Model: The proposed IoV network architecture has three different layers, vehicles, edge and cloud. It is also assumed that all vehicles connect with each other using an 802.11p Wireless Access in Vehicular Environments (WAVE) interface, that, using the same values as [8], such as a transmission range of 100 m and a data rate of $10Mbps$. In addition, each vehicle is also equipped with a more powerful interface that can only be used to connect with RSU. This interface has a bigger range, 250 m but has the same data rate.

There are different types of nodes with different parameters, the RSU, cars and buses. However, the only difference between them is the buffer sizes, an RSU has 1024MB while a car has a buffer size between 64MB and 256MB and all buses have 256MB.

As far as the offloading of data from the edge to the vehicles is concerned, a node uploads data to the nearest RSU to be processed, and, once the computation is finished, the output will be sent through vehicles in the direction of the desired node. Nevertheless, the RSUs are all connected through the edge, thus, some data can be shared between them.

2) RSU Placement: In this work the ONE [6] simulator was used, therefore, the addition of RSU and the representation of the edge layer had to be done. The RSUs are supposed to be stationary and their position should be near a road. Therefore, the implementation of the RSU movement/position was based on two movement models that the simulator already had, *StationaryMovement* and *MapBasedMovement*.

The RSUs had to be spread across the city, in this sense, two different approaches were developed.

The first one, 7 circles with the same centre but different radius are placed centred in the city centre and the nodes are distributed through the circles. The problem of this method was that some important locations of the city were not covered. To tackle this problem, another solution was developed. Consequently, the map was studied and the RSU were placed one by one into strategic points, such as main interceptions, main accesses to some parts of the city and city centre. The simulator has data of some public transports routes that were also taken into consideration.

This last node distribution used only a portion of the number of nodes to cover all the city and presents much better results. However, to implement this approach some information about the city is required, therefore, in a simple simulation in a new scenario, the first distribution approach would be advisable.

With more data the second approach could be transformed into an optimisation problem, reducing the number of nodes and improving results of the network. In a real-world application this should be done since, as will be shown, these nodes have a huge impact on the network.

B. Clustering Based Routing

The data dissemination is one of the utmost problems when the IoV is concerned, in this sense, a clustering baser routing algorithm was developed.

There are countless options to cluster nodes in a network, the most trivial one is clustering based on their position. However, this leads to a problem, the stability of the network. The aim of the clustering algorithm is to create clusters of nodes in IoV, where most nodes are moving in completely different directions. For example, at a given time, a picture of all cars in Lisbon was taken, and, according to that they were assign to clusters based on their position. How different would these clusters be if the photo was taken 1 minute later?

In light of the above, another metric was used, social relationships. In this way, every time a node connects with other node their social strength will be updated. On the other hand, this metric will decrease if the two nodes do not connect during a given period. All nodes will update their social relations every hour and they will be clustered according to the similarity of their relations, as explained below.

Assuming that all nodes keep a map that link the node's social strength with its neighbours and the value assigned to it. In this sense, considering that the RSUs also store their social strengths. If the edge merges all these data, it could be presented as an Euclidean space with N dimensions, where N is the number of RSUs and each node would be represented by its social strength, as shown in Figure 1, where node A has never connected with RSU 2 and node B has already connected with all RSUs.

Based on this representation, and keeping in consideration that the algorithm's complexity, for a number of nodes much larger than the number of dimensions (i.e. number of nodes much larger than number of RSUs), is suitable for live computation the k-means algorithm [9] was used to compute the clusters. The algorithm will run 3 iterations each time it is called. The number of iterations is a balance between resource consumption and the accuracy of the algorithm.

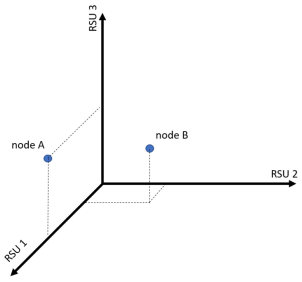

Fig. 1. Simple Euclidean space with 3 RSU

Just after the k-means finishes, the occupation of all clusters will be checked, and, in the case that one or more are empty, the centroid of this(ese) cluster(s) will be placed over a point of the largest cluster and the algorithm will run a few more iterations, then, there will be no empty clusters and a balance between their size.

Once all nodes are divided into clusters, the edge will choose the cluster head, a node that represents the cluster and will gather the messages from the cluster to deliver to the somewhere else in the network, reducing the overload of the network. This choice is based on two metrics, the similarity between each node and the centroid that represents the cluster, and the ego betweenness centrality of each node in the network. The value that represents the similarity is usually between 1 and 0. However, the value that represents the ego[1] is not, thus, a function based on a logistic function was used to map the ego values to values that could be compared with the similarity. This function was chosen since only a small percentage of the nodes have an ego larger than 7.5 and those are the nodes that should be considered to cluster heads. Then, the cluster head is the node whose metrics produces the lower value from equation 1.

$$normalized\ ego * 2 + similarity \tag{1}$$

Notice that the cluster head formula, equation 1, gives and extra weight to the ego in order to increase the importance of a smaller normalized ego over the similarity to the centroid.

As stated before, it is essential to fully use the information and communication technologies for achieving the coordination of human, vehicle and environment. Therefore, a reliable routing algorithm is mandatory.

This algorithm must comply with the IoV paradigm presented above. Therefore, there are some aspects to take into consideration while designing this algorithm.

First of all, it is assumed that the messages are transported via nodes and not through the edge. This is important since even though the RSUs share this connection, it is just used to share data about the state of the network.

[1]metric concerning the centrality of a node in a graph of all nodes and their connections

As far as the impact of the cluster in the routing is concerned, there are a few constraints to ensure that the connecting point between clusters are the RSUs and preferable the cluster heads.

In addition, it is important to refer that if two nodes are within range and one has a message for the other one, even though they may belong to different clusters, the message will be delivered.

In light of the above, the RSU based privo (R-privo) was developed.

The routing algorithm is based on the privo algorithm [7] and is built upon four mechanisms, inter-contact time, forwarding policy, routing metric and average separation period metric.

The inter-contact time metric is estimated for each pair of nodes using an exponential weighted moving average to update values based on previous data.

There are three forwarding policies available, direct single copy, direct multi copy and limited multi copy. In the first option, each message can only be sent from node to node without the sender keeping a copy. While on the second one, all nodes keep copies off all messages that pass through them. Finally, the last policy has a limited number of times a event/message can be copied.

As far as the routing metric is concerned, it depends upon three smaller metrics that are similarity to destination, betweenness centrality and average separation period.

At last, the average separation period is represented by the mean time to encounter.

Here, *MTTE*, *Sim* and *EgoBC*, refer to the metrics presented by [7], Mean time to encounter, Similarity and Ego betweenness centrality respectively.

The R-privo algorithm is all about complying with the rules of the paradigm used. Therefore, each node will follow the IoV routing rules to filter which of the nodes in range can receive the message. With this set of nodes, will run an algorithm to check if the other node is preferable comparing to the current node.

According to the offloading method chosen, the connection between RSUs is only used to share which is the closest to the destination node and which is the RSU with the largest ego betweenness centrality, it will never be used to send messages from one to another. This information will be used to forward the messages in the direction of the best RSU or to the most central one when the destination node has never made contact with any RSU.

There is also one more aspect to take into consideration about the proposed routing algorithm. When the limited multi copy policy is used, there are times that instead of just transferring that message, the node will create a replica to the other node and keep the message. This change divides the copies of the messages among nodes with a higher probability of meeting the destination node.

As far as the implementation is concerned, the decision is based on the type and cluster of the node who carries the message. If the node is a RSU or it is from the same cluster of the destination node, it will always create a replica before transferring the message. In addition, if the similarity between the carrier of the message and the destination is larger than a threshold, the message will also be replicated and transferred. Otherwise, the node will just transfer the message.

C. Intrusion Detection System

In the near future, IoV will be present in our lives, they will change the way we see transportation and mobility. However, this responsibility brings a huge risk. What if something goes wrong? This leads to a need of an entity monitoring the network, overseeing all the nodes and raising flags if one of them is not responding as it should, or a node is intentionally misbehaving.

In this section, a deep learning approach that runs on the edge layer will be described. This algorithm runs based on data from each node, retrieving a probability of a misbehaviour by the given node, and if this output is high, it will give an indication of the most probable behaviour.

It is important to mention that the objective of this project is only to detect and, if possible, identify the malicious behaviour. The goal of this algorithm is to help the network manager by giving alerts, it will not affect any node or the network directly.

The data given to the algorithm are the type of node (normal node, cluster head or RSU); ego in the network; number of nodes in range; number of messages received; number of messages sent; buffer size; node location; and social strength with its three strongest connections. The location is given as a coordinate X,Y.

These data will be stored in blocks of three arrays that form a 2D array that contains data of a given node at three different times. The number of samples in each block of data was chosen to take into consideration the past of the node without a huge impact on the performance of the algorithm. Therefore, the input format will be 9×3.

All data used is sent by the vehicles to the edge every time a nodes connects with an RSU. If a node does not share its data, it means that it is not connecting with the edge, consequently, its role in the network will be less important and the impact of its actions will be lower.

These data will enter in a convolution auto encoder that will use the convolution property to simulate a dependency of data during time. The aim of this algorithm is to encode the input data into a smaller space than the original, and then, from the encoded data, decode it to restore the original input. The performance of the algorithm is measured by the mean squared error between the input and the output data. Therefore, the train of the algorithm is all about finding a robust analysis of the data, thus, building a robust anomaly detector.

The auto encoder used is formed by convolution layers. These layers receive a 3D input and convolve it with a set of kernels, or filters, and applies an activation function to the filter outputs, in this case, a Rectified Linear Unit. Each kernel has a localised support in the first two spacial coordinates and a full range on the depth on the input (third coordinate). It will

compute the value of each neuron of the next layer according to this paradigm.

All layers of the algorithm have the same parameters, a kernel of 3×2 and a stride of 1. The kernel can be seen as a filter, where its size is the filter size. The amount of the filter shift is given by the stride.

At each iteration, the kernel will analyse the data it is covering and compute a single value to represent it. In the end, the data is reshaped according to equation 2, that has to be applied to both dimensions.

$$output\_vol = \frac{(input\_vo - kernel\_size)}{stride} + 1 \qquad (2)$$

In light of the above, the encoder consists of two convolution layers that will transform the input data into 5 neuron arrays of 7×2 and then 3 arrays of 5×1. Then, the decoder is two layers of the reverse function with the same parameters, as shown in Figure 2.

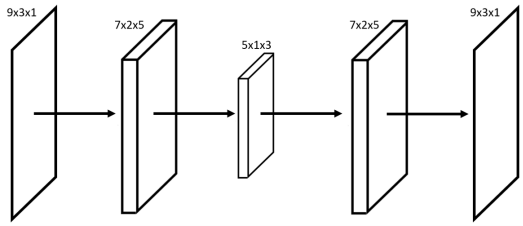

Fig. 2. Architecture of the Convolution Auto Encoder used

To conclude, in the algorithm used (Figure 2), each neuron of the second layer will be computed based on the first layer neurons covered by a kernel (3×2) that will shift one position every time (stride) it is applied. In this case, the depth of the first layer is just one, thus, the kernel will only consider 6 neurons each time. Notwithstanding, for the other layers, as their depth is larger than one, the kernel will cover more neurons, for example, between layer 2 and 3, at each iteration, 30 ($3 \times 2 \times 5$) neurons will be considered.

To implement this neural network, a library called DeepLearning for Java [10] was used. It was trained in a different program, only the testing part runs at the same time as the simulations. As mentioned before, the algorithm will issue the mean squared error between the input and the output, therefore, values closer to 0 represent nodes whose behaviour is closer to the normal. When this value is larger than a threshold, the node will be labelled as misbehaving. This threshold will be adjusted during the simulation to be adapted to each case. Its initial value will be the largest value obtained while testing the algorithm. The mean score of this test as its relation with the maximum value will also be stored and will be used as a guideline to the update of the threshold.

During the simulation, every time the algorithm runs, the mid value will be adjusted as shown in equation 3.

$$mid\_value = mid\_value * 0.9 + new\_value * 0.1 \qquad (3)$$

Then, the relation obtained during testing will be applied to this value, thus, the threshold will be updated.

As mentioned above, there are plenty of behaviours or malfunctions that may have a huge impact on the network. Therefore, the following behaviours were implemented.

The node not working behaviour consists of a node dropping or simply not sending any of the messages it receives. In addition, the node may have a malfunction on its 802.11p communication interface, therefore, it can only communicate with RSUs. These actions were implemented on the three different types of nodes, RSUs, cluster heads and normal nodes.

As far as the **identity spoofing** is concerned, it is expressed by a node that successfully identifies itself as another node, or has having a different role in the network. for example, a node claiming that it is a RSU or a cluster head when it is simply a normal node.

The implementation is essentially lying when the node connects with any other node by changing its identification to the desired one or replacing the nodes identification with a RSU's.

The sybil attack occurs when a node, the attacker, subverts the reputation system by creating a large number of pseudonymous identities and uses them to gain influence in the network. In this case, by creating these replicas, the attacker would rise in the network and in case of success would become a cluster head. The impact of this node would also be noticed outside of its cluster hence the attacker would occupy a central node in the network.

This attack lies on the creation of a group of nodes that will be always in range of the attacker. For the other nodes, these nodes will appear as normal vehicles.

One aspect to take into consideration when creating this behaviour is the fact that the replicas and the attacker have to share the same resources (computer power and buffer size), therefore, the replicas of the attacker usually do not have as many resources as a normal node.

Besides the detection of a misbehaviour, the edge should also be able to label the above described actions. Therefore, when the previous algorithm raises a flag, the data will be forwarded to a new neural network.

This deep learning algorithm will have the same input as the previous one but will be a classification problem, thus, it will have only 5 different outputs: sybil attack; node not working; identity spoofing; normal behaviour; and new unknown behaviour. It will start with a convolution layer to simulate the dependency of data over time and will consist of 3 dense layers and a softmax function to discern the labels. The choice of the number of layers and the number of neurons was a balance between the flexibility of the network, as with more layers the network can synthesise a wider variety of nonlinear functions with fewer neurons, and the difficulty to train given that a deeper (more layers) and denser (more neurons) network requires more computation.

The Data set used to train this network was divided into two, the training set and the validation set. This model uses

the validation set to implement the early stopping, a technique used to ensure that the network is not over-fitted to the trained data.

Another aspect to be taken into consideration is the fact that the network cannot be trained to detect new unknown behaviours, as that would be a paradox. Therefore, the neural network is trained to label the 4 known behaviours, and, when the confidence on the result is low, the behaviour will be classified as an unknown one. The output of this algorithm is an array with four positions, one for each label, that represent how confident is the network in assigning the label to the data set. The sum of the four positions of the array is always one. Therefore, given a data set, if there is not a confidence larger than 0.5 for any label, the data can be labelled as unknown behaviour.

IV. Results and Analysis

In this section, the results of the algorithms and implementation mentioned in the previous section will be discussed. Starting with the performance of the clustering based routing algorithm will be presented and finishing with the results obtained for the monitoring system will be analysed.

A. Clustering Based Routing

1) Clustering Evaluation: As discussed before, the network will divide the nodes into clusters. The main goals of the approach used are covering as many nodes as possible and creating the clusters as stable as possible. In addition, the larger the number of clusters, the more specific information a cluster gives about its nodes. Nevertheless, the size of each cluster shall be taken into consideration, i.e. the number of clusters shall be much lower than the number of nodes.

Thus, the clusters were made using a k-means algorithm based on the social relations of the nodes. To evaluate the performance of the algorithm, the number of nodes eligible for clustering and the number of cluster changes per hour were measured during several simulations.

The following test was done by simulating 56 nodes over 48 hours. The 56 nodes are 5 of each group of home, office and meeting spots plus 2 buses for each of the 8 routes. In addition, 30 RSUs were hand placed to improve the performance, as shown in the previous section. Each simulation was done 3 times with different seeds and the results presented refer to the average of the results obtained.

Regarding the number of nodes covered by the algorithm, it is important to mention that only nodes that already made contact with an RSU are known by the edge and therefore eligible for clustering. In addition, this result does not depend on the number of clusters. The algorithm is based on the relations with the RSUs, therefore it only depends on their position. Nevertheless, it is an important metric that should be taken into consideration when studying the clusters.

Several tests were done and as expected, during the first two days, the nodes start to made contact with the RSUs which leads to an increase in the number of nodes covered. The most significant changes occur in the morning and in the evening that is when the nodes are supposed to move between home, office or meeting spots (following the workday movement model). During the work time, the values are much more stable. This values and changes should be taken into consideration when studying the stability of the algorithm.

To further analyse the clustering algorithm, a cluster changes metric is defined where all cluster changes count as one, when a node does not have a cluster and joins one and when a cluster changes from one cluster to another, both count only as one cluster change.

In the simulation with only one cluster, all nodes eligible for clustering belong to the same cluster, therefore all changes will be nodes that were added to the clustering set. In addition, the number of clusters was varied between one and eight. Given that all clusters have exactly the number of nodes covered, with the objective of presenting clear data, the changes related with new nodes eligible for clustering were subtracted from the number of cluster changes per hour. In Figure 3, each colour represents a different simulation scenario, each vertical bar shows the sum of nodes who had a cluster change during each hour. For example, during the second hour, there were a total of 65 cluster changes, but when using two clusters, there was just an average of 7 changes. In addition, as it can be seen, there is not any cluster change counted in the first hour, this is because all the changes in this period were nodes making the first contact with the RSUs.

To give a clearer image of when the network starts to converge, an average of all these simulations were done and it can be seen in Figure 4. In addition, in this figure, in contrast with the previous one, all cluster changes are represented, both nodes who connect for the first time with an RSU and nodes who change from one cluster to another. That is why, in the second figure, there are values on the first hour of the simulation, all these cluster changes are nodes who were not registered for clustering and made the first contact with one RSU during this time slot.

As seen in Figure 4, it takes an average of six/seven hours to reach a stable point. Furthermore, with the increasing of the number of clusters a decrease of the stability can be seen (Figure 3). However, as it can be also be seen that even for the less stable configurations, after 48 hours the number of changes per hour tends to less than 1, which shows that the approach developed leads to a stable clustering solution as desired.

In addition to the above studied approach, the stability of clustering based on proximity was also tested. The goal of this test was only to give a comparison with the most trivial approach to the problem, therefore, only a superficial analysis was done. In here, the contacts that a node had during two consecutive time slots were compared and the number of changes counted. During the first slot the number of changes was 11, then dropped to 8 and every time the node moved the values were similar, when the node was not moving these values were near 1. Nevertheless, these values are just for one node, which means that it represents only 1/50 of the cluster changes. Therefore, the difference between the stability of the

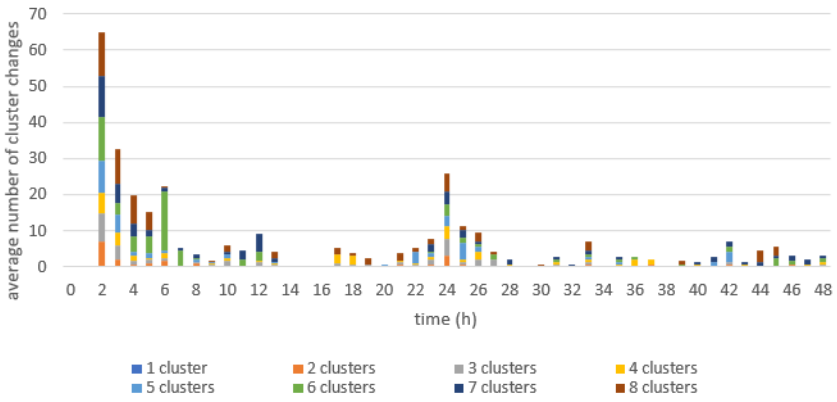

Fig. 3. Sum of the average cluster changes over time

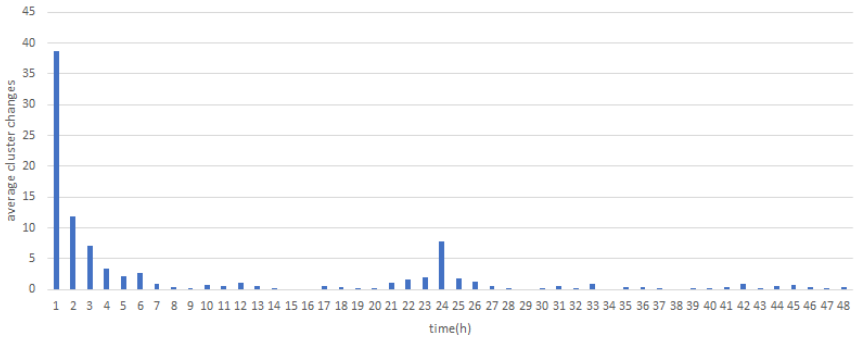

Fig. 4. Average cluster changes over time

social based approach and this one is evident.

All in all, when clustering a network of vehicles, there are some aspects to take into consideration, such as, what are the clusters representing, the size of the clusters and the network, and the stability that the algorithm used will provide. As presented, clustering based on social relations leads to stable results, and it is advisable to be used when the nodes are always moving and have some movement patterns, such as in the IoV case.

2) R-privo: As far as the routing algorithm is concerned, it should be noticed that the algorithm is an adaptation of another one built to Delay Tolerant Networks, the Privo algorithm [11].

To evaluate the performance, the following data was collected over several simulations: the delivery probability, the overhead ratio, the average latency, the average hop count and the average buffer time.

This study will focus on the delivery rate and the overhead ratio of each solution and the scenario used to this simulation was the same scenario used for the clustering tests.. A Limited multi copy policy of 6 copies was chosen to boost the delivery rate with a small impact on the overhead. In addition, the algorithm will be evaluated by comparing the delivery rate and the overhead ratio with four different algorithms, epidemic, prophet, bubblerap, and privo. Moreover, all simulations have both vehicles and RSUs, however, while in the R-privo, the RSUs have a different behaviour than the other nodes, on the other algorithms, the only difference between them is the buffer size and the movement, the RSUs are static.

In the first instance, the delivery rate of each algorithm was compared. As shown in Figure5, the privo have the best performance, immediately followed by the R-privo. Then the bubblerap and the prophet reach nearly 45% and the epidemic could only achieve 30%.

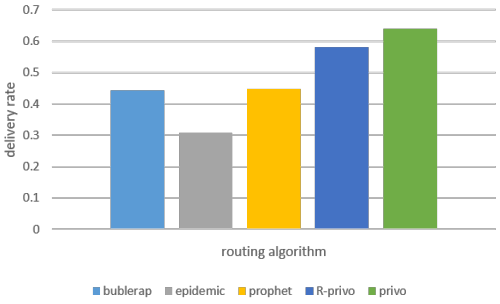

Fig. 5. Delivery rate of the 5 routing algorithms

As far as the overhead is concerned, there is an enormous difference between the algorithms, with values ranging between 1311 and 5, with the R-privo and the privo algorithm presenting the best overheads by far. This disparity of values was expected hence the epidemic and the prophet do not assign any limit to the number of copies, the first one copies the

message to every node it makes contact with and the other one copies to all nodes that have a better metric than him. As far as the bubblerap, has its own restrictions on who to copy to. However, with these configurations, the privo and the R-privo have a limit on the number of copies of 6. Figure 6 shows the overhead of each algorithm as a table printing the values. To encompass all values in the same figure, a logarithmic scale was used.

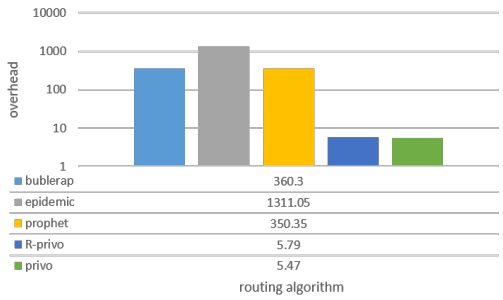

Fig. 6. Overhead ratio of the 5 routing algorithms

As seen in the figure above, the R-privo presented worse results than the privo. This is due to the fact that one of the main differences between these algorithms relies on the restrictions to transfer a message, the R-privo will only share the messages within its cluster while the privo will share them with any node with a greater chance to meet the destination.

To conclude, the R-privo is an algorithm that was developed to take advantage of the architecture of the IoV, it can be seen as an evolution of the privo algorithm to this paradigm. The R-privo has a delivery rate in the same range of the other standard routing algorithms with an overhead much lower. In addition, this algorithm needs, in average, only 1.8 hops to deliver a message when the others need 3.0, 4.7, 3.8 and 1.9 for bubblerap, epidemic, prophet and privo respectively. On the other hand, both the latency and the average buffer time of this algorithm are much worst that the others, notwithstanding, this is to be applied to a network of vehicles where each node have more resources that a usual node in a network where these algorithms are used.

B. Intrusion Detection System

As presented before, the monitoring of the network consists of two different algorithms, one that detects anomalies and the other that tries to identify that anomaly.

This study was done over several simulations when some nodes were programmed to misbehave and the edge would try to detect. Every time the first algorithm detects an anomaly, the data will also pass through the second one to classify the behaviour.

In this work, a true positive is when an attacker situation is correctly labelled and a false positive when a normal is classified as a misbehaving one. A true negative is when a normal node is labelled correctly and a false negative is when an attacker situation is mislabelled.

Before using it in the real simulation, the algorithms were trained and tested. The training of the algorithms consisted of uploading a data set and the algorithm updated its weights to best perform on the given data. As mentioned in the previous section, this data was obtained in several simulations with different parameters.

During all the following tests, the attackers will behave maliciously from the beginning until the end of the simulation. Their data will be monitored every time they made contact with an RSU and the algorithm will not take into consideration any label given to that node before, i.e, in this study, instead of testing a node based on all its history, the tests will only take into consideration the data sample acquired at each moment. Therefore, they are presented below as different attack situations. Usually, each simulation has 5 misbehaving nodes and during the simulation time, each node is inspected by the algorithm 20 to 30 times, depending on its movement patterns.

After all the training, the algorithms were deployed to the simulations. In the first instance, the algorithm was used to monitor a network where all nodes had normal behaviour. To this analysis, 1500 tests were done over 106 nodes. From these 1500 tests, 117 were labelled as a potential anomaly, however, from this 117, the second algorithm labelled 108 as normal behaviour. This indicates that from 1500 samples, only 9 were false positives, only 0.6%.

Considering that in the beginning of the simulations all the nodes signatures are the same, if the algorithm is applied, all nodes will have the same classification. Over the simulation, the signatures will become more and more distinct. Therefore, in the first instance, the time that the systems need to warm up, i.e. there is a difference between the signature of a normal node and an attacker, was studied. With this objective, several simulations focusing the first hours of the simulation were done, these simulations concern the first two behaviours (sybil and id spoofing) with distinct parameters. In this study, the evolution of the true positives, false positives and false negatives were collected and once the values of the true positives is larger than the values of false negatives and false positives, it indicates that the system is warmed up and the algorithm is ready to run. For example, in Figure 7 is presented the evolution of the classification metrics for the id spoofing attack.

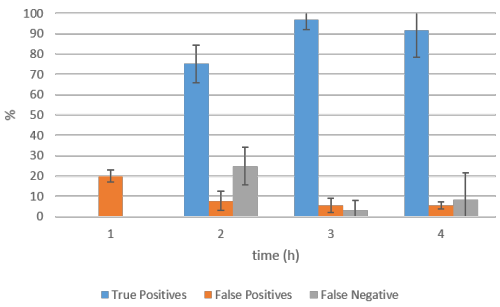

Fig. 7. Evolution of classification metrics for the id spoofing attack

During these tests, the classification metrics have started to stabilise after 3 hours, therefore, 10800 seconds was chosen as warm up time.

1) Sybil Attack: With all the tests and the simulations for a normal situation done, it was time to start testing against misbehaviour. At first, the sybil attack was studied. For this, several simulations were done, varying the movement model of the attacker and the number of replicas it creates. Among all these simulations, the monitor system had run 597 attack situations, from these, 407 were labelled as having an anomaly and were studied by the behaviour identifier who labelled all 407 as sybil attackers. This means that once an attacker is captured by the anomaly detector, the behaviour identifier will be able to label this attack correctly. In addition, this false negative rate (approx. 32%) derives from the similarity of an attacker with a node with an important role in the network, i.e. high Ego betweenness centrality. Moreover, during this study, a total of 8000 nodes were monitored and only 16 were mislabelled as sybil attackers which leads to a false positive rate of only 0.22%.

During the study above, the fact that when the attacker creates more replicas, it is easier to detect was confirmed. When testing with 3 replicas, the false negatives were significantly higher than the most used configuration, 5 replicas.

2) Identity Spoofing: As far as the id spoofing attack is concerned, when a node identified as an RSU makes contact with the edge, and this node is not registered as an RSU, the monitor system already knows that something not normal is occurring. Therefore, it will always run the behaviour identifier.

During the testing period, the algorithm run the data of 18000 tests, from which 389 were attack situations. Considering the attackers, 370 were labelled as attackers. In addition, during these tests, there were 100 false positives, i.e. nodes that were labelled as attackers and had normal behaviours.

3) Node Not Working: Regarding the detection of nodes not working, the difference between a node not working and a node whose role in the network is not important is small. I.e. a node that at a given time stopped working and a node who went to an isolated place and cannot connect to anyone result in a similar behaviour. This behaviour was not tested on RSUs hence if one of these nodes was not working it would be unable to connect with the edge and therefore, it was assumed that the edge would be able to detect this anomaly without the algorithm.

Regarding the results obtained, for he node not working behaviour the true positive rate was only around 50%. As mentioned before this behaviour has a signature similar to the normal behaviour and even when the anomaly detector captures these nodes, it is not guaranteed that the behaviour identifier will not label it as a normal node with a normal behaviour. To have a greater certainty about this behaviour, it would be advisable to wait for at least one more iteration on the algorithm and check if the classification stays the same.

On the other hand, the false positive rate on this behaviour, just like on the others, is close to zero, in a total of 5000 evaluations, only 33 were misclassified as a node not working.

In Table I, is shown a summary of the performance of the monitoring systems in this scenario.

TABLE I
SUMMARY OF THE MONITORING SYSTEM

	sybil	idspoofing	node not working
true positive (%)	68	95	52
false positive (%)	0.22	0.6	0.8
true negative (%)	98.78	99.4	99.2
false negative (%)	32	5	48

The system was also tested in two different scenarios where a strong dependency on the data used to test was proven. In addition, comparing the influence of the data deviation on the performance of the algorithms, the behaviour identifier showed a smaller influence.

To conclude, the monitoring algorithm has a high success rate identifying the attacks above and despite the results obtained for the nodes not working, the algorithm can be a huge help when managing a network. As expected, the anomaly detection system has a strong dependency on the training scenario, while the behaviour identifier has more flexibility. Nevertheless, as shown above, the performance improves with the training of the algorithm, and, for better results, the algorithm should be trained with data of the network where it is deployed.

In a real world application, the algorithm should be collecting data and from time to time and add the data collected to the training set. By doing so, the algorithm would learn more about the network it is supervising and it will be more suitable to detect any anomaly on that system. Moreover, to increase the certainty of a classification, the algorithm could assign the different classifications to the respective node, thus, combining consecutive measurements to reduce the uncertainty.

In addition, an increment on the deviation to the normal behaviour could be seen on all nodes when there is a node misbehaving, proving that even one node misbehaving has an impact on the network as a whole.

V. CONCLUSIONS

A. Achievements and Contributions

The main objective of this thesis was to develop an edge-based monitoring system to the internet of vehicles. To achieve this, a clustering algorithm for these networks, a routing algorithm to ease the communication between nodes and a deep learning algorithm to detect and identify possible attacks and misbehaviour were also developed.

On the light of the above, a routing algorithm that could comply with the paradigm of IoV was needed. With that objective, the R-privo was developed, an algorithm based on the privo algorithm to take advantage of the edge layer and the location of the IoVs.

In addition, the algorithm also divides the network in clusters by their social relationships, grouping the nodes that most likely will share the same connection. While studying

the clustering approaches on the IoV, two metrics were used, similarity and ego betweenness centrality. During this approach, higher importance was given to the similarity when choosing the cluster head. All in all, clustering based on social relationships between nodes was elected as an advisable approach to this problem, providing a stable solution that due to the choice of the IoVs as anchors, is also robust to an increment of the number of nodes.

In addition, the R-privo, even though being built with more constraints to comply with the IoV paradigm, was able to beat the traditional algorithms (epidemic, prophet and bubblerap) and reach the same performance levels of the privo algorithm, presenting itself as an ideal algorithm for these networks. Furthermore, the R-privo's routing rules are strongly related to the cluster of each node, therefore, an increment on the number of clusters will decrease the delivery rate of the algorithm

At last, with the environment set to properly simulate an IoV network, the edge based monitoring system was developed. This system is formed by two different deep learning algorithms, one to detect anomalies in the network and the other to identify those anomalies.

The first one, the anomaly detection system, has the purpose of detecting deviations from a normal behaviour in the network. It is very useful to identify nodes who may be misbehaving, however, this algorithm cannot distinguish the different misbehaviours.

The second algorithm, the behaviour identifier, is able to identify any known attack, i.e. the algorithm has to be taught how to detect each studied attack, therefore, the identifier cannot detect any new attack.

The monitoring systems is built to take advantage of both algorithms. In addition, the choice of the data to be collected and used to evaluate each node was also studied as well as the architecture of each algorithm.

At last, in a scenario similar to the one used to train, the monitoring system is able to detect and identify 68% of the sybil attacks, 95% of the id spoofing attacks, and 52% of the nodes not working. In addition, the behaviour identifier has able to correctly classify all the Sybil attackers that were sent by the anomaly detector. Nevertheless, the results obtained for the nodes not working is due to the fact that this behaviour and a node who is in an isolated zone have similar signatures. Notwithstanding, the monitoring system is able to identify the above mentioned behaviours and can be a great tool for any network manager.

To conclude, in this thesis the ONE simulator was adapted to work with IoVs, an algorithm was developed to comply with the network paradigm and to take advantages of what the IoVs add to the system, and an edge-based monitoring system capable of identifying several behaviours was developed, fulfilling all the objectives presented in the introduction.

B. Future Work and System Limitations

With more time and resources some improvements could be done.

As far as the location of the RSUs is concerned, a optimisation problem could be developed to find the optimal location of each RSU.

Regarding the clustering algorithm, the G means algorithm [12] could be tested and compared with the one used. The G means algorithm has the advantage that it will choose the most suitable number of clusters, providing greater flexibility to the algorithm.

At last, the monitoring system could be retrained from time to time with live data, adapting the system to the network it is monitoring. In addition, the classifications of each node could be saved to have the evolution of each node over time as it would also increase the certainty of a given classification by comparing it with the previous ones. The algorithm could also be able to act to reduce the impact of a misbehaving node in the network. Moreover, some behaviours could be added to the system, for example, the black hole or grey hole attack. Finally, the complexity of the deep learning algorithms could be greater, which would improve the performance of the monitoring systems but would require more computation power.

REFERENCES

[1] F. Yang, J. Li, and T. Lei. Architecture and key technologies for internet of vehicles: a survey. *Commun. Inf. Netw. 2, 1–17 (2017)*, 2017.

[2] K. Sha, R. Errabelly, W. Wei, T. A. Yang, and Z. Wang. Edgesec: Design of an edge layer security service to enhance iot security. *2017 IEEE 1st International Conference on Fog and Edge Computing (ICFEC)*, pages 81–88, 2017.

[3] Amin Vahdat, David Becker, et al. Epidemic routing for partially connected ad hoc networks. *Technical Report CS-200006, Duke University*, 2000.

[4] Anders Lindgren, Avri Doria, and Olov Schelén. Probabilistic routing in intermittently connected networks. In Petre Dini, Pascal Lorenz, and José Neuman de Souza, editors, *Service Assurance with Partial and Intermittent Resources*, pages 239–254, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg.

[5] P. Hui, J. Crowcroft, and E. Yoneki. Bubble rap: Social-based forwarding in delay-tolerant networks. *IEEE Transactions on Mobile Computing*, 10(11):1576–1589, 2011.

[6] Ari Keränen, Jörg Ott, and Teemu Kärkkäinen. The ONE Simulator for DTN Protocol Evaluation. In *SIMUTools '09: Proceedings of the 2nd International Conference on Simulation Tools and Techniques*, New York, NY, USA, 2009. ICST.

[7] N. Magaia, C. Borrego, P. R. Pereira, and M. Correia. eprivo: An enhanced privacy-preserving opportunistic routing protocol for vehicular delay-tolerant networks. *IEEE Transactions on Vehicular Technology*, 67(11):11154–11168, 2018.

[8] N. Magaia and Z. Sheng. Refiov: A novel reputation framework for information-centric vehicular applications. *IEEE Transactions on Vehicular Technology*, 68(2):1810–1823, 2019.

[9] John A Hartigan and Manchek A Wong. Algorithm as 136: A k-means clustering algorithm. *Journal of the royal statistical society. series c (applied statistics)*, 28(1):100–108, 1979.

[10] Deep learning for java. Available at https://deeplearning4j.org/ (2020/12/10).

[11] N. Magaia, C. Borrego, P. Pereira, and M. Correia. Privo: A privacy-preserving opportunistic routing protocol for delay tolerant networks. In *2017 IFIP Networking Conference (IFIP Networking) and Workshops*, pages 1–9, 2017.

[12] Greg Hamerly and Charles Elkan. Learning the k in k-means. In *Advances in neural information processing systems*, pages 281–288, 2004.