Traffic congestion avoidance through a cross simulation between ns-3 and SUMO

Riccardo Castelletto[†], Davide Masiero[‡]

***Abstract*—Ns-3 (network simulator) and SUMO (Simulation of Urban MObility) are two programs that can simulate, respectively, network traffic and an urban mobility system. Taken singularly they have been used many times to give a precise opinion and analysis about the construction and the inner functioning of a network or the traffic information of a street system. The project presented in this paper aims to combine these two trusted simulators to analyse the traffic in a given snippet of a map, predict and avoid congestion, working in a vehicle-to-vehicle (V2V) environment. Ns-3 and SUMO are called to complete cooperation to give feedback about the traffic situation and to change the route of the involved vehicle when needed. This work will use SUMO's data to retrieve information and build network metrics in ns-3. This metrics, then, will be used to reorganize the route and the internal state of the single-vehicle when possible congestion is detected to lighten the load on the network or to make lower the travel time. An analysis will be done about how well this project works, and many comparisons with a normal run will be done to understand the validity of this concept.**

***Index Terms*—Vehicular communication, V2V, ns-3, SUMO, Tu Dresden, Network analysis, Simulation, Traffic avoidance, Traffic congestion.**

I. Introduction

Road traffic has always been a problem: since when the total number of vehicles on the streets was low until now when there are thousands of vehicles in the same city. The creation of traffic congestion is the first problem that we face when the number of vehicles rises. Other secondary effects are involved here, like environmental pollution and the general disappointment of every person stuck on the street.

Navigation systems, when they were introduced, were not able to understand and inform the user about any traffic congestion on the selected route, because this was not their primary objective. Then those systems evolved: they can now show how much traffic there's on a road and, by extension, on the entire route. Normally, this evolution will involve a global tracking system, like GPS, but this project aims to translate the mobility system into a network simulator, and use network metrics as ways to understand the quantity of traffic on the network itself.

Now the world is moving towards the idea of self-driving cars, and this change of environment requires the ability of the vehicles to communicate with others and with the road (RSU - roadside unit) in a trusted and efficient way. The two options described above are the typical situations where the researches are focused on, and they are V2V (vehicular-to-vehicular) and V2I (vehicular-to-infrastructure) and V2X that is the mix of the previous two. This new type of environment requires new protocols to allow the vehicles to communicate in every type of conditions (like high speed). This kind of situation leads to new standards in the communication systems like IEEE 802.11p, that is born for this purpose.

Then in our project, we are focused on a vehicle-to-vehicle environment where every vehicle has its network device installed, and anyone of these can send a certain number of UDP packets through a wireless channel (IEEE 802.11p), via broadcast. Each device in range can receive those packets and change its internal state basing its decisions on the information received and its analysis. Each device has its source rate.

Then the goal of our project is trying to understand by discrete-time simulations the capabilities of the network under some constraints and try to optimize it by using just network information, position and direction of the vehicles. In a normal scenario there's a huge number of vehicles involved, and so is for the network devices, then it could happen that some network characteristics will change, like delay and packet loss, bringing congestion in the communication. The final goal of this project is to minimize the impact of the number of vehicles to ease the exchange of data in the whole network, by the identification of network congestion and then with the "side effect" of having better routes for the vehicles in the scenario.

This project can be considered as feasibility study because the analysis was done without all the possible variables due to the long execution time for each simulation, but we have observed that in certain situations our approach can bring some interesting results. The results that we have obtained highlight some interesting behaviours like: with our approach on average delay and packet loss are lower than a run without our application, and this is what we are looking for. Instead, the throughput on average are comparable between each other, but in some situations where the channel is congested, i.e. in a bad situation, our method guarantees a higher throughput and lower delay and packet loss.

The remainder of this paper is organized as follows. In section II, we present the literature. In section III we describe the software (simulators) used in this work. Section IV contains the detailed implementation, while section V describes the analysis of the results obtained by the simulations. Finally, section VI concludes the paper. Vehicular communication, V2V, ns-3, SUMO, Tu Dresden, Network analysis, Simulation, Traffic avoidance, Traffic congestion.

[†]Universita' degli studi di Padova: riccardo.castelletto@studenti.unipd.it
[‡]Universita' degli studi di Padova: davide.masiero.5@studenti.unipd.it

II. Related Works

In this section, we are going to describe the recent and related articles about this topic. In fact, in recent years, the need for simulators for VANET environment increased a lot, and there are a lot of different techniques adopted to simulate them. In need of an open-source complete way to join ns-3 and SUMO, a German team of the University of Dresden developed an ns-3 module that allows the bidirectional coupling of the two simulators [1], so the obtained results led to a good complete simulation for a VANET environment.
As we can see in the following papers [2] [3] [4] [5], there are a lot of works that evaluate the routing protocols in a VANET environment. Our project acts differently, in fact, it aims to perform a study about the feasibility of the network, use its statistics as congestion indicators and then try to avoid the congestion of network and on the roads.
Instead, other kinds of projects try to make a specific application for specific purposes like [6] [7]; however, they can be built on top of a system like ours that guarantees to not overload the capacity of the channel.

III. Simulators

For the project, two simulators in combination were used, ns-3 and SUMO. Now a brief description of the two simulators and how this two were coupled to allow us to develop the final project.

A. NS-3

Ns-3 [8] is a discrete-event network simulator; its software is free and publicly available. It's primary intended use is for research and education. The goal of this simulator is to build a realistic open-source simulation environment suitable for networking research. Ns-3 offers a well documented, easy to use and easy to debug simulation core, which allows configuring any project in any part, to trace data and then analyse it. Ns-3 is kept updated by a worldwide team of volunteer maintainers to be able to support any new technology. It has also the possibility to generate packets that can be read by a physical network device, so it's also easy to include simulation in a real context.
This simulator is used in this work as a way to implement the network layers as the physical layer and MAC layer, and on top of that, we have installed the 802.11p standard and the internet stack. In this work for the channel, we used a log-distance path loss model, a Nakagami-m fading and random loss for the shadowing, to have a good reproduction of a realistic channel. Then whit this setup, we have installed a Wi-Fi device on each node (vehicle). And finally each device broadcasts its information at a certain rate into its area of transmission; instead, the packets obtained by a device are then analysed by the receiver subroutine.

B. SUMO

Simulation of Urban MObility, or SUMO [9] for short, is an open-source, microscopic, multimodal simulator to represent a context of vehicular mobility on a given map. It can simulate many vehicles moving and interacting through a given road network. The fact that SUMO is microscopical lays on the fact that each vehicle is modelled explicitly with its route, and moves individually. SUMO has its Graphic User Interface, from which it is possible to follow the entire simulation directly and intuitively. In our work, SUMO is used to download and adapt a snippet of a map through Open Street Map; then, a script is invoked to generate randomly all vehicles and routes.
The network, composed by junctions and streets, the vehicles and the routes are written in an XML like syntax, so it's easy to parse and to analyse. It is also possible to personalize each one of these files with the needed information.
In this work, SUMO is used to simulate a realistic environment of vehicles with the correct mobility pattern, based on a map chosen by us. For the map, we have chosen a little portion of the Oklahoma City, sited in Oklahoma in the United State of America. This city was chosen by chance because we got some problems with other big city caused by one-way routes or by the absence of good alternatives paths for the cars. We haven't chosen a big part of the city, instead we choose a very small one, in a way to have simulations that don't take too long time to been simulated, because to see the effect of our project it's needed a quite long simulation (we use 90 seconds simulations), and with an high concentration of nodes in a small part of the map. Another important characteristic is that with SUMO we can keep constant the number of vehicles in the simulation, then allow us to make a comparison between different simulations with different parameters and varying the number of vehicle at our leisure.

C. NS3-SUMO-COUPLING

The coupling between ns-3 and SUMO isn't already present, and then we used an external module [1] developed by the University of Dresden (TU Dresden), in Germany. This module for ns-3 was created to implement a bidirectional coupling between the network simulator and SUMO. It can parse the XML configuration files of the SUMO simulation and it has some built-in methods to obtain any kind of information from these files. To allow that, the module is built on top of the TraCI API provided by the creators of SUMO to allow the communications with other programs. Simply, the module creates a TCP client which interfaces with a TCP server that is implemented by the TraCI API of SUMO, and the tasks of the client are to allow the bidirectional communication with the correct implementation of all the features provided by the SUMO's API.
This coupling module takes any vehicle from the traffic simulator and associates it to a node in the network simulator, transferring the mobility model of the vehicle to

the node. After that, each node can send, receive and handle any packet to simulate an exchange of information through the network channel set in ns-3.

IV. Implementation

After the creation of the network topology, the map and the coupling of these two-phase, we have started to implement our decision algorithm. The main execution unit of each vehicle is a set of three operations:

1) Send: at this moment the device builds the UDP packet to be sent via broadcast, gathering all the needed information with some dedicated functions of the coupling module. The information sent to other vehicles are the ID of the sender, the road ID where the sender is, the time (in microseconds) in which the packet is built, the frequency at which the packets are sent by the sender and a value that is different from zero if the vehicle is stuck in the traffic, that will be updated with the results of the other operations. The sending procedure is scheduled at a regular interval, that can be changed in the function of some parameters in a way that we're going to see later, in the decision part.

2) Receive: this function is activated every time a packet arrives at the receiver. In this function, each arrived packet is parsed and decomposed in each of its fields. Once this decomposition is over, every information extracted (the information into the packet are about the vehicle status), are saved into two different data structure. After that, the current time is taken to calculate the packet delay in the most accurate way and then are calculated some statistics (this time network statistics), that are saved into another data structure. All the information take in this part are going to be used in the decision part;

3) Statistics calculation and decision: this function calculates many metrics like mean throughput, mean delay, packet loss and others, using the information that had been extracted in the receive callback. These metrics will be then used to understand if the vehicle is stuck and if it has to implement congestion control policies. The strategies chosen to apply this concept are: perform a rerouting operation, change the interval between each packet to send and the way how these two policies are combined.

Now that we have described briefly the three main part of the program, now we are going to do a detailed description of the three main approaches that we used to decide for the congestion control for the network.

A. Rerouting

The first thing and maybe the more obvious is to change the route of a vehicle, in this way we can move a node from a very congestion situation towards a better one. This technique goes to analyze different kind of situations and requires a lot of tuning parameters. To apply this technique we decompose the process in more phases: vehicle status identification i.e. if it is stuck into the traffic, congestion's road identification and rerouting process.
The first big problem is to understand the situation only by the metrics of the network because can bring a lot of misunderstandings if we don't do that carefully. To avoid this problem we split the decision into more parts. We have a decision tree where a branch takes into account only statics statistics, instead, the other one takes care of the evolution of the statistics through time. Another important part is the voting system.

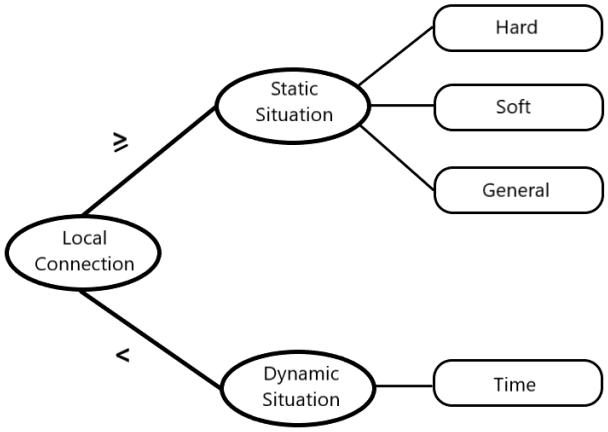

Fig. 1: Representation of the decision tree.

Now we are going to describe in more details the decision tree that is represented in `Fig.1`. The first condition it's to decide if we take care of the static part or the dynamic part, this decision is made by taking into account the number of "local" connection. What do we mean by local connections? The local connection in this application is considered only if the vehicle that sends the packet is in the same road of the receiver that can be understood by the road ID put into the packet by the sender. We have seen that this constraint allow us to understand if there is a good connection, that is a good number of vehicles in the same road, probably they are moving slow and then we can capture a lot of numbers of packets. Then if this constraint is satisfied (number of local connection greater than 4 as the default value), we are in a static situation. Now we have other three conditions that bring the vehicle in a situation of congestion if no one is satisfied we consider the vehicle in an optimal situation. The three constraints are one hard, one soft and one general. The hard constraint represents a very bad situation, where we have both a high "local" throughput compared to the minimum throughput generate by a single car, and high

”local” delay compared to the minimum delay registered by the car in the interval time. The soft one considers the high ”local” throughput or the high ”local” delay but in this case, we consider another statistics that is the percentage of the velocity of the car compared to the max velocity allowed by the road. If it is under a certain threshold it means that the vehicle probably is blocked on the traffic. The last constraint indicate that the node is in a general congestion situation, i.e. we have a very high general throughput or a very high delay.
Regarding the dynamic part, we are going to consider a fixed size window (that can be chosen, the default value is 4), that takes into account the evolution of the system. The decision to set the status of the vehicle as congested is pretty simple because we consider the evolution of the ”local” delay and ”local” throughput. We consider two sub-window the old one in time and the new one. The congestion identification is made by a simple constraint: if the values in the newest window are bigger than the old one multiplied by a settable value it means that the vehicle is congested. This constraint means that the node is going in a bad situation.
The last part to understand the vehicle congestion status is the voting system. This part, in the decision process, is the last one. After the decision of the vehicle for itself, the node takes care of the other vehicles status, thanks to the fact that every vehicle sends its current status in the UDP packet. In this part we count the number of ”local” nodes (i.e. in the same road), that tell us that they are congested; if the half plus one of the number of vehicles (taking itself into account), in the road told us that there is a congestion, then we set the state of the vehicle as congested.
After the identification of the vehicle state, we can identify the roads congested by the analysis of the UDP packets, because we have the status of the nodes and the roads where they are. Then we set the travel time cost of each congested road to a higher value, and trigger a rerouting subroutine that tries to avoid the congestion, and after that, we reset the normal travel time cost of each road.

B. Interval change

The idea behind changing the interval between each packet is generally to lighten the load on the receiver, so less packet is sent by each device in this situation; in this way, it’s possible to decrease the delay and the packet loss, because a device has less packet to analyse. A side-effect is that in this way also the throughput is affected because sending fewer packets in the same amount of time means less information. It is to understand if it’s more likeable to reduce the throughput but being able to receive and correctly analyse the packets arrived. More analysis will be done on this topic.
The way how in this project this strategy is brought is by analysing the value of the packet loss, that describes the network state at that moment. In particular, if the packet loss of the node is currently greater than a certain fixed high value, we assume that this device is surrounded by many other possible sources of packets, which are increasing the network congestion in the wireless channel of every vehicle involved: in this case, the interval is doubled, so the new packet frequency of the device is the half of the previous value. If instead, the packet loss is low, this variable is reduced by half until it returns at its normal value.
To avoid having too many or too few packets sent, the interval is lower bounded and upper bounded, respectively, by 7 and 50 milliseconds.
The value of the variable about the current packet rate of a device is used to precisely know how many packets have to be received by another one, because in congestion situations the intervals will be greater, few packets will be sent, and counting the expected number of received packets, avoiding to take in account the personal rate of each device, will be a wrong strategy.

C. Combination of rerouting and interval change

Combine these two strategies means that a device while entering congestion, will try to understand how bad that situation is because its metrics will recognize that a lot of vehicles are stuck in its way. If the vehicle recognises itself stuck, by the second strategy, the intervals change to have fewer packets to send and solve the congestion, and this happens until the limit is reached. At the same time, under the correct conditions, the vehicle will try to change its route and avoid the situation, by the first strategy. The device, in general, with the combination of these two approaches, has more chances to avoid congestion in the network.
Effectively, this combination is expressed by adding a condition on the interval change strategy, which takes in account the state of the vehicle, e.g. if it’s in congestion, then we reduce its packet rate. On the other end, reducing the packet rate of the node changes the metrics involved in the calculation of its vehicular state, e.g. congested or not.

V. Results

In this section, we present the results obtained in the several simulations performed. These runs have been differentiated by various parameters choices. By varying only one parameter per time it is possible to see how good is our implementation. In particular, we focused on the variation of four different parameters: the aggressiveness of the thresholds, the size of the window that considers moments in the past, the number of vehicles in the scenario at the same time and the strategy adopted. All simulations were performed on a total running time of 90 seconds, the value was chosen because it was enough, in our opinion, to generate an appreciable number of situations in the scenario. Where not specified, the simulations were done with the two strategies combined. The descriptions of the various variables variations are the following:

1) By varying the multipliers of the thresholds, it’s possible to understand how the network behaves in a situation where the devices have a different limit to declare that their vehicle is stuck in the traffic. What follows is that

lower multipliers will lead to lower thresholds and every vehicle will have less margin before declaring itself in congestion, and vice-versa for the higher thresholds. This situation can be seen in the figures `Fig.2` and `Fig.3`. The comparison with the baseline, represented

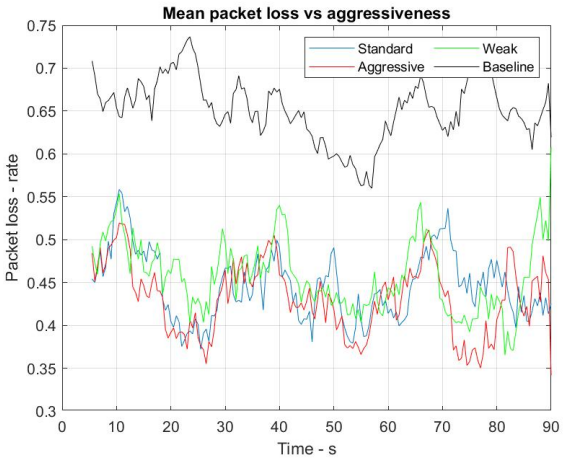

Fig. 2: Packet loss per variable aggressiveness.

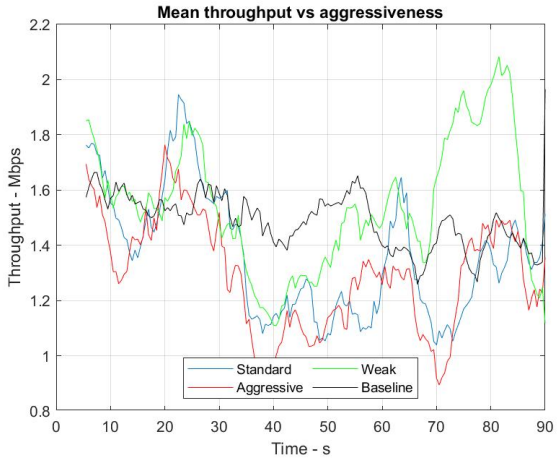

Fig. 3: Throughput per variable aggressiveness.

in the images by a black line, is clear while talking about the packet loss, where the work performs better with all the threshold values. Between the various levels of aggressiveness, as expected, the one that seems to give a better packet loss is the one with more aggressive variables, represented by a red line. It's a different situation for the throughput because the baseline, in general, has similar values than the other plots. The only difference is in the stability of the network metric during the custom simulations, where they are more unstable: here, the best thresholds, that guarantee a higher throughput, sometimes even better than the baseline, are the weak ones. It must be noticed that there are moments when the project simulations are worse than the baseline. In general, the best approach should be the one with loose thresholds, because the throughput can be way better, and the packet loss is very low compared to the baseline, but not so high compared to the other threshold values. About, the delay, it's very bad in the baseline, while considering the projects, it's very low, without any considerable difference between the threshold values.

2) The variation of the evolution window is a good way to understand the differences between a static analysis of the system and a more dynamic approach. As already pointed before, if the window is small, the consequent analysis is more static, while if the window is bigger, the system counts for the variations of the metrics and in this way the analysis is more dynamic. This difference can be analysed in the figures `Fig.4` and `Fig.5`.

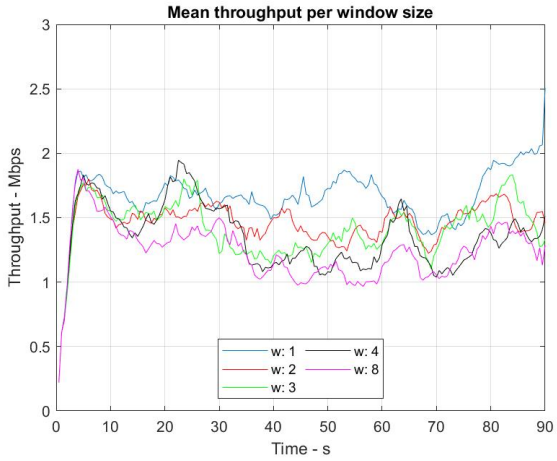

Fig. 4: Throughput per window size.

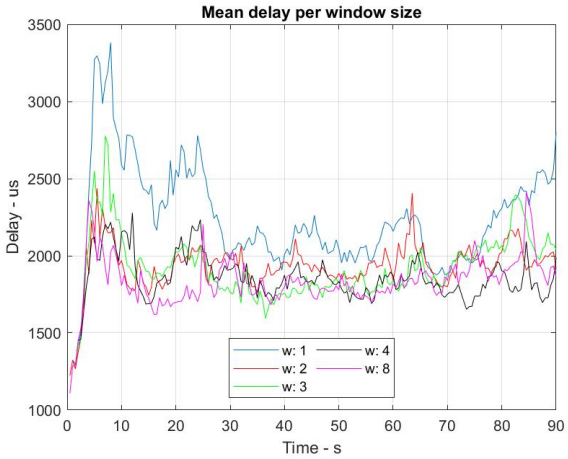

Fig. 5: Delay per window size.

Generally, if the window is bigger (black and purple lines), the throughput is lower, so less information is transmitted through the network, but at the same time, the delay is lower. In particular, in the static case, when

the window is one (blue line), the delay is higher than the other plots that are all similar. The packet loss behaves similarly than the other metrics. The size of the window appears to be a good way to perform a trade-off between throughput and packet reception, and even in this case, it looks like that the best compromise is to take a mid-low window different than one, like two or three, so the resulting analysis is dynamic. In this way, the throughput is not so affected but the delay is lower and this results in a better overall behaviour of the network.

3) The following analysis is a simple check about the network capabilities with the variation of the number of vehicles in the scenario. The relative graphs are displayed in `Fig.6`, `Fig.7` and `Fig.8`. It's easy

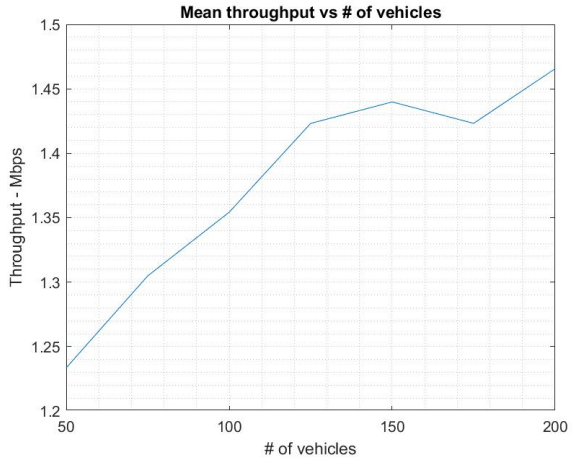

Fig. 6: Throughput per number of vehicles.

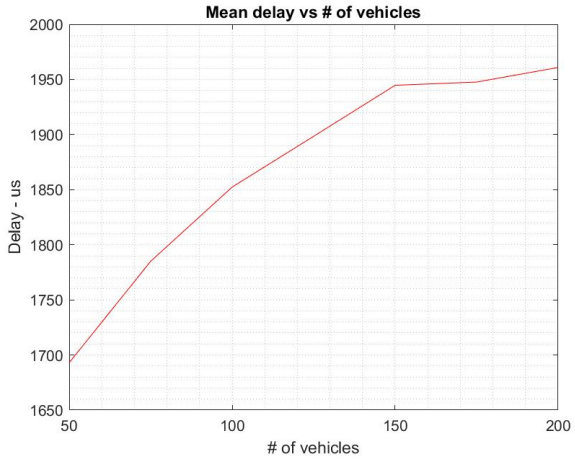

Fig. 7: Delay per number of vehicles.

to understand that a high number of vehicles will bring more information on the overall network, so the throughput is higher. More vehicles mean more inter-vehicular communication, and the increase of the

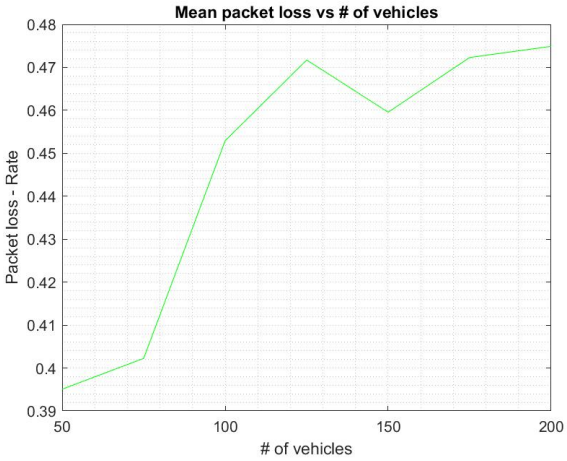

Fig. 8: Packet loss per number of vehicles.

number of packets raises the delay and the packet loss linearly with regards to the number of vehicles.

4) In this section, it's possible to see the impact on the statistics of the strategy selected for this project:
   - **TT**: means that both rerouting and interval change are used;
   - **TF**: means only rerouting;
   - **FT**: means only interval change;
   - **FF**: is the baseline, without any strategy;

   Each bar in every graph is the mean value for every vehicle for every second of the simulation. The through-

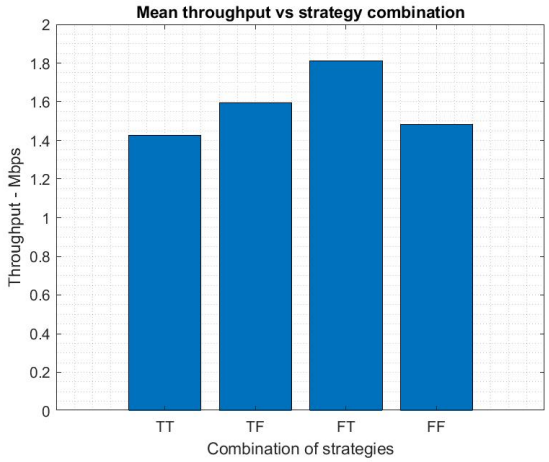

Fig. 9: Throughput per strategy.

put plot, `Fig.9`, shows that the only case where the project has a lower value than the baseline is when the strategies are applied in combination. This can be due to the way how the vehicle behaves: if it's stuck, lowers its packet rate and only if many vehicles tell it that they are in congestion, it tries to go away from there. In this way, it continues to receive packets at a low rate or stops receiving them because of the change of the route, and the throughput goes down. In other words, the two

strategies ”get in each other’s way”, and they can not take full advantage of their benefits. In the other two cases, it’s possible to see a rise in the throughput, due to the strategies adopted singularly. Considering delay

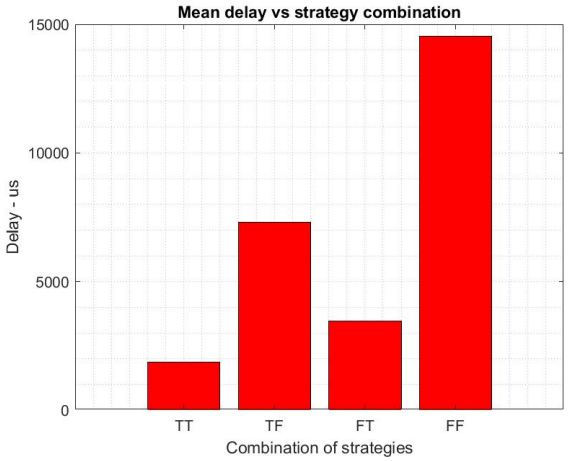

Fig. 10: Delay per strategy.

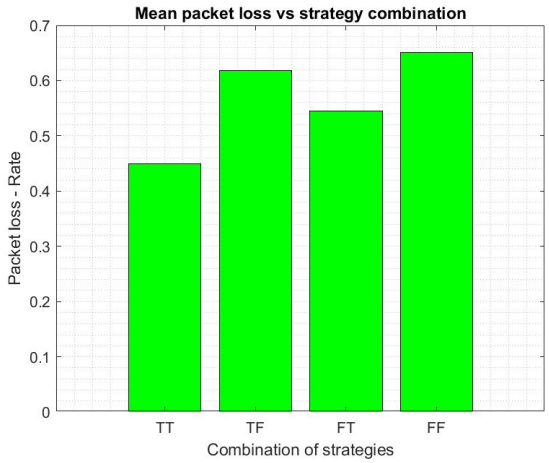

Fig. 11: Packet loss per strategy.

and packet loss, it’s possible to see in the respective figures, `Fig.10` and `Fig.11`, that the baseline has on average higher values compared to those coming from this work. In particular, the best strategy to lower these metrics is the interval change, which values are very low. This can be explained by the description of the strategies themselves: in the rerouting-only case, if a vehicle cannot change its route, which is possible, will stay stuck and will keep receiving packets at the same rate, so the number of packets rises and delay and packet loss get worse. If the interval between a packet and the other can be augmented, fewer packets will be sent, so each device will be able to process them in a better way.

One last analysis has been made: the one considering the union of the results about the metrics and the number of vehicles in the scenario and the results about the differences between the strategies and their combination. The plot regarding through-

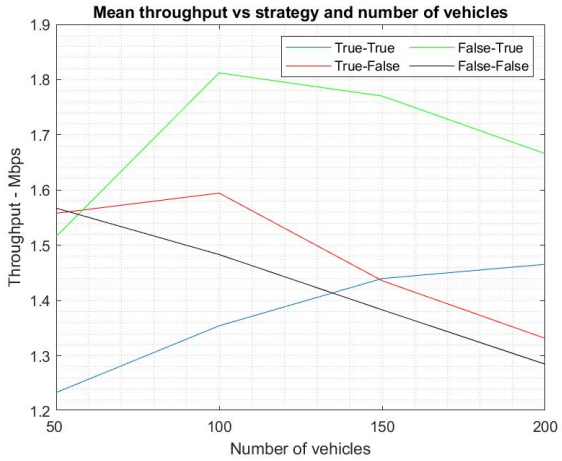

Fig. 12: Throughput vs strategy and number of vehicles.

put, `Fig.12`, shows that the strategies that increase the interval are those that maintain a good throughput even while the number of vehicles is rising. With rerouting, instead, the throughput goes down when more vehicles are entering the scene. In `Fig.13`, the plot about the delay, it’s clear that

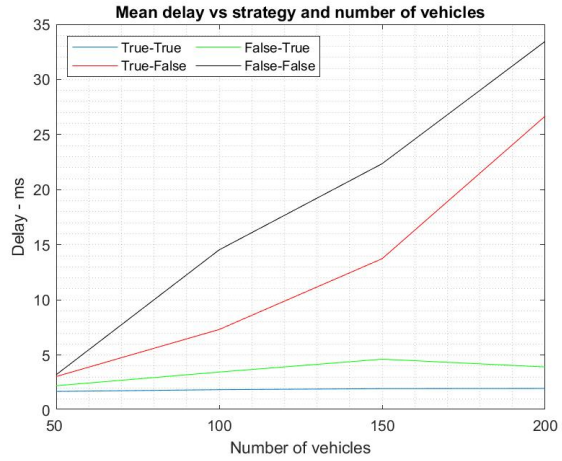

Fig. 13: Delay vs strategy and number of vehicles.

a solution without the interval change strategy brings more delay than the others. As already explained, this is possible because reducing the number of packets sent in a congestion situation results in a better network situation. In the last image, `Fig.14`, as it is possible to see, the best strategy combination is the one with both rerouting and interval change, because a vehicle has more chance to get away from the congestion and correctly receive more packets. Connecting all the result from this plots, the best strategy for our parameters and our condition is the one about changing the interval between each packet because it can have a great throughput (in relation with the initial parameters, of course) and low delay and packet

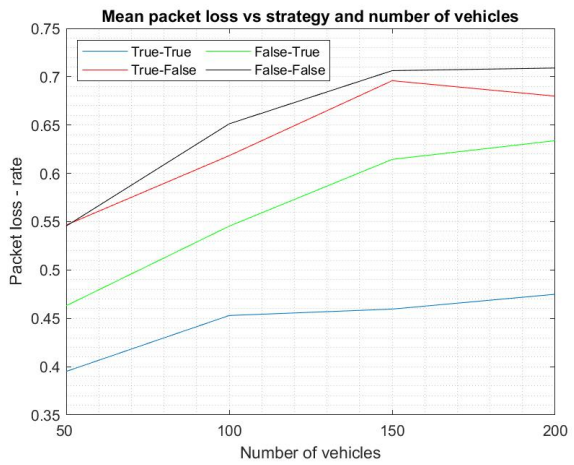

Fig. 14: Packet loss vs strategy and number of vehicles.

loss. If the throughput can be sacrificed, the approach with both strategies has better values for the other metrics.

VI. Conclusion and Future Works

With this paper, we have tried to understand the limits of a VANET network and then how to use this limits to maintain the network in a good condition to allow the vehicles to have a good communication also in a congested situation. Our project is only a preliminary study, in fact, the long-time required by the simulations and the big amount of parameters, we couldn't explore each aspect of this topics, but thanks to our first analysis we have seen some interesting trends in the evolution of the throughput, delay and packet loss.

We can conclude that, obviously this three parameters are affected by the number of vehicles in the scenario, but we can say that with a good management of the packet rate with regards to the network congestion, and a good manipulation of the metrics with regards to the vehicular position, we can avoid congestion both for the network and the streets, but at the same time maintaining a good performance in the throughput but increasing the reliability of the channel, with the decrease of the delay and the packet loss.

As we said before, this is only a first exploration of the argument, and then there is a lot of work to do to improve this research. To make this work more authoritative and relevant in this research area, we need a lot more data to confirm our first analysis. Then, we need to try our program in different maps, with a bigger variation of the number of vehicles and a finer granularity in the threshold parameters. Over this kind of analysis, we can take into account other parameters that are only touched in this paper, like the vehicles out of any communication range, to improve the application rate and the packet size, in a way to have better general throughput and a further improvement of the delay and packet loss. Then, if we do more tests the trend can be confirmed or not. We hope that this study can help somehow, and for this reason, we link below our Github repository: `https://github.com/XWerer/NAS_Project`.

References

[1] S. K. Patrick Schmager, "ns3-sumo-coupling,"
[2] T. M. P. M. Chitraxi Raj, Urvik Upadhayaya, "Simulation of vanet using ns-3 and sumo," April 2018.
[3] E. S. L. B. M. I. K. U. Vladi Kolici, Tetsuya Oda, "Performance evaluation of a vanet simulation system using ns-3 and sumo," March 2015.
[4] A. M. Gamal Sallam, "Performance evaluation of olsr and aodv in vanet cloud computing using fading model with sumo and ns3," April 2015.
[5] Y. S. E. S. M. I. L. B. Vladi Kolici, Tetsuya Oda, "Performance evaluation of a vanet simulation system using ns-3 and sumo considering number of vehicles and crossroad scenario," October 2015.
[6] J. M. David Smith, Soufiene Djahel, "A sumo based evaluation of road incidents' impact on traffic congestion level in smart cities," 2014.
[7] M. K. Adwitiya Mukhopadhyay, S Raghunath, "Feasibility and performance evaluation of vanet techniques to enhance real-time emergency healthcare services," Sept. 2016.
[8] "Network simulator: https://www.nsnam.org/,"
[9] "Simulation of urban mobility: http://sumo.sourceforge.net/,"