


Article

A Simulation Tool for Evaluating Video Streaming Architectures in Vehicular Network Scenarios

Pedro Pablo Garrido Abenza * , Manuel P. Malumbres , Pablo Piñol 
and Otoniel López Granado 

Department of Computer Engineering, Miguel Hernández University, Avda. Universidad, s/n, 03202 Elche, Spain; mels@umh.es (M.P.M.); pablop@umh.es (P.P.); otoniel@umh.es (O.L.G.)

* Correspondence: pgarrido@umh.es; Tel.: +34-96-665-8387

Received: 27 September 2020; Accepted: 20 November 2020; Published: 22 November 2020



Abstract: An integrated simulation tool called Video Delivery Simulation Framework over Vehicular Networks (VDSF-VN) is presented. This framework is intended to allow users to conduct experiments related to video transmission in vehicular networks by means of simulation. Research on this topic requires the use of many independent tools, such as traffic and network simulators, intermediate frameworks, video encoders and decoders, converters, platform-dependent scripting languages, data visualisation packages and spreadsheets, and some other tasks are performed manually. The lack of tools necessary to carry out all these tasks in an integrated and efficient way formed the motivation for the development of the VDSF-VN framework. It is managed via two user-friendly applications, GatcomSUMO and GatcomVideo, which allow all the necessary tasks to be accomplished. The first is primarily used to build the network scenario and set up the traffic flows, whereas the second involves the delivery process of the whole video, encoding/decoding video, running simulations, and processing all the experimental results to automatically provide the requested figures, tables and reports. This multiplatform framework is intended to fill the existing gap in this field, and has been successfully used in several experimental tests of vehicular networks.

Keywords: simulation tool; vehicular networks; video delivery; HEVC

1. Introduction

A vehicular ad hoc network (VANET) is a multi-hop wireless network in which messages are exchanged, both between mobile vehicles and with fixed infrastructure nodes. Several types of communication may take place: vehicle-to-vehicle (V2V), vehicle-to-infrastructure (V2I), or vehicle-to-everything (V2X). This type of network has many applications in the field of intelligent transportation systems (ITS), such as in traffic information systems, road safety, Internet access and entertainment applications (infotainment). Applications such as infotainment, security surveillance [1,2], health care assistance on-the-fly [3,4], video transmission for overtaking manoeuvres [5] and other applications related to video transmission require high bandwidth and low packet transmission delay, especially in real-time applications.

However, wireless networks are subject to several problems, such as limited bandwidth, the use of a shared medium in which transmissions from different communicating nodes can collide, and other phenomena such as signal attenuation with distance (path loss), the presence of obstacles (shadowing), and refraction and reflection (multipath) effects. Furthermore, in the specific case of VANETs, the communication window between vehicles is very limited due to their high mobility, which causes dynamically changing network topologies. Each of these problems can lead to a high rate of packet loss; this is especially true for video transmission applications due to the high bandwidth and bounded packet delay required, and particularly for real-time applications.

Simulation is the most commonly used technique for experimenting with vehicular networks, as it is easier to design a simulated network scenario than a real one or an experimental testbed. Furthermore, this approach makes it feasible to evaluate a scenario under different conditions and to collect statistics, and ensures that experiments are reproducible [6].

Research on video delivery in vehicular networks requires the use of many independent tools, such as traffic and network simulators, intermediate frameworks, video encoders and decoders, converters, platform-dependent scripting languages, data visualisation packages and spreadsheets, and some tasks are usually done by hand. A lack of tools to carry out all the necessary tasks in an integrated and efficient way formed the motivation for the development of the Video Delivery Simulation Framework over Vehicular Networks (VDSF-VN) framework. The main innovations of the proposed simulation framework are as follows: (i) a reduction in the probability of errors in the simulation chain; (ii) automation of the longer and more tedious processes (such as configuring utilities and file formats, and ensuring the interoperability of utilities); (iii) coordination and support for the supervision of all simulation experiments, from the experimental design step to the analysis of simulation results; and (iv) a user-friendly graphical user interface (GUI) that integrates all the required tools and allows researchers to easily define simulation setups, run simulations, visualise the results in a graphical and interactive way, and generate predefined technical reports from simulations.

The main goal of our simulation framework is to allow researchers to work on specific parts of a vehicular video delivery system, and to evaluate the performance of their proposed schemes not only when working in an isolated simulation environment, but also (and most importantly) when working on a part of the whole system. In this way, interactions between their proposed systems and other parts of the video delivery system can be better analysed to improve the overall performance of the system. In our opinion, our simulation framework will be useful for researchers working on specific parts of a vehicular video delivery system, such as the design of new video coding tools, the use of innovative video packet forwarding approaches or new error concealment (EC) approaches, among others.

The remainder of the paper is organised as follows. In Section 2, we carry out a literature review of existing frameworks in the field of vehicular network simulation, with a particular focus on those intended for the evaluation of video streaming. In Section 3, we present an overview of the proposed VDSF-VN environment, including the different elements contained in the framework. Finally, in Section 4, some conclusions are drawn and directions for future research are highlighted.

2. Related Work

A literature review shows that several frameworks or tools have been developed with the aim of making it easier to conduct experiments with VANETs by means of simulation. As explained below, many configuration files and tools need to be used when building a VANET network scenario with Simulation of Urban MObility (SUMO) [7]. Despite the set of utilities provided by SUMO, it is sometimes necessary to manually write or edit configuration files, which is often an error-prone and time-consuming task, especially for a novice user. In view of this, several applications have been developed with the aim of making SUMO more accessible by including a GUI to reduce the time and effort required for these tasks. Examples of such GUIs include eNetEditor [8], TrafficModeler [9,10], SUMOPy [11], CityMob for Roadmaps (C4R) [12], and OSMWebWizard. However, the majority of these applications are intended to be used only with SUMO, and numerous errors may appear when running simulations with the OMNeT++ simulator [13] or the VEHICLES In Network Simulation (Veins) framework [14].

The need to use several individual components (simulation packages, custom scripts, etc.) and the difficulties in visualising and analysing the obtained results are described in [15]. In order to overcome existing drawbacks, a framework called WGL-VANET (Web-based Visualization Tool for VANET Simulations) was developed, which offers an interactive and easy-to-use web interface for the ns-3 [16] simulator and the SUMO traffic simulator. In [17], a framework named ELVS (Efficient Large-scale VANET Simulator) was proposed, based upon the JiST/SWANS [18] and SUMO simulators and some

other third party software. This is an open-source framework that was developed in Java and has a graphical interface that allows the user to visualise vehicles in realistic scenarios. It was intended to allow users to view the internal states of their components (simulators) and to facilitate analysis of the output data. In [19], another framework was developed for a specific application in VANET simulations, and was also mainly based on the JiST/SWANS simulator.

In addition to these frameworks, several others have been developed for the specific purpose of evaluating video transmission over vehicular networks in cases where more elements need to be used (video encoders/decoders, packetisers, etc.). For example, EvalVid [20] performs a quality evaluation of video transmissions encoded with MPEG4 [21] based on a calculation of the peak signal-to-noise ratio (PSNR) and other network metrics such as delay, jitter, and loss rate. This framework has been extended to support several other network simulators, such as ns-2 [22], ns-3, and OMNeT++ with the Castalia framework [23]. The latter was implemented in a simulation framework based on EvalVid, called Mobile Multi-Media Wireless Sensor Networks (M3WSN) [24]. Another framework based on EvalVid is QoE Monitor [25], an extension that supports the ns-3 simulator.

The main differences between our simulation framework and the other simulation tools available in the literature are as follows: (i) our proposal includes the most accurate and realistic vehicular simulation models by means of the OMNeT++ and Veins simulation frameworks; (ii) it includes a state-of-the-art video coding system based on the High Efficiency Video Coding (HEVC) standard [26], and can be easily updated with the future video coding standard H.266/VCC; (iii) it is the only system to fully integrate all of the tools required for the video delivery simulation chain (such as those for the definition of vehicular scenarios, traffic mobility, video packetisers, video codecs, interactive graphic plots, report generators, etc.), and to be defined in a modular way that allows for easy integration of new modules into the simulation framework (e.g., a forward error correction (FEC) module); (iv) it was designed as a multiplatform tool for different operating systems (Windows, Linux and Mac OS X); and (v) it includes multi-threading support to allow several tasks to be run in parallel, in order to reduce user response times.

The existing difficulties in carrying out experiments with vehicular networks and the absence of an integrated environment that met our needs formed the motivation for the development of our new framework, VDSF-VN, which is described in detail in the next section.

3. VDSF-VN Simulation Framework

In this section, we describe the VDSF-VN framework. It consists of traffic and network simulators, intermediate frameworks, video codecs, packetisers, platform-dependent scripting languages, data visualisation packages, spreadsheets, and additional third party software packages, which are managed via two graphical applications, GatcomSUMO and GatcomVideo, as shown in Figure 1. GatcomSUMO is used in the setup of a network simulation scenario, including all the configuration parameters needed to properly run a network simulation. This application can be used to manage and configure: (i) the OMNeT++ network simulator with the Veins framework, which controls the simulation of the data exchange between the network nodes; (ii) the SUMO traffic mobility simulator, which is responsible for the design and coordination of the vehicle mobility patterns in a particular vehicular network scenario, and includes many console commands and scripts that extend its functionality, such as importing external maps, computing vehicle routes, etc.; and (iii) other third party software utilities like OpenStreetMap (OSM), which provide real city street maps that can be imported by SUMO as part of a network scenario. Through the use of GatcomSUMO, the user can define the network scenario to be used and the details of the vehicle mobility patterns. The resulting configuration files will be used when the next application starts the OMNeT++ network simulator.

The second application, GatcomVideo, can be considered the front-end application of the VDSF-VN framework, since it allows the user to configure, coordinate and manage all the steps in the vehicular video delivery process from content generation at the source node to video rendering at the destination node. The only configuration task it delegates to GatcomSUMO application is the

definition and configuration of the network scenario and the traffic mobility of the vehicular node. GatcomVideo is composed of three main steps (modules): (i) the pre-process module, which is used to encode the source video via the HEVC video encoder, to convert the resulting bit stream into a Real-time Transmission Protocol (RTP) video packet sequence (the packetisation process), and to provide a traffic trace that is readable by the OMNeT++ network simulator; (ii) the OMNeT++ simulation module, which is used to launch simulations for a predefined network scenario (using configuration files from GatcomSUMO) and to provide the results of all simulations for analysis; and (iii) the post-process module, which gathers all the simulations results to (a) decode the received bit stream and render the resulting decoded video; (b) interactively plot the desired simulation results; and (c) export the results in the form of graphics files and technical reports.

Both the GatcomSUMO and GatcomVideo applications were developed in Java and offer a user-friendly graphical interface (front-end). However, as explained below, both applications invoke several other command-line programs (back-end) to perform their tasks. In the following subsections, we will analyse these graphical applications in detail.

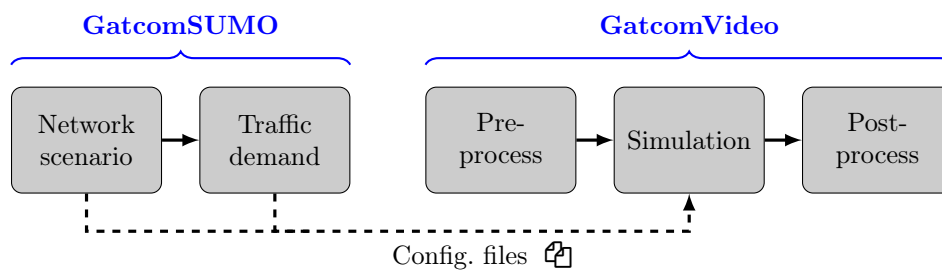


Figure 1. Video Delivery Simulation Framework over Vehicular Networks (VDSF-VN): General view.

3.1. GatcomSUMO

A network scenario in the SUMO simulator is composed of several Extensible Markup Language (XML) files and some other plain text files that must follow strict syntax rules. SUMO supports maps from different sources, such as real maps downloaded from OpenStreetMap [27] in XML format (*.osm.xml), and abstract or synthetically generated maps with various geometric structures (grid, spider or random). Maps downloaded from OpenStreetMap need to be imported into SUMO by means of the `netconvert` utility, which generates a network file (*.net.xml). The obstacles file (*.poly.xml) can be generated from previous files via the `polyconvert` utility, in order to include information about different obstacles (e.g., buildings) that can affect communications. Abstract scenarios can be generated with the `netgenerate` utility. In addition to the network files, SUMO also requires a traffic demand file (*.rou.xml), which can be generated with the utilities that SUMO provides, such as `duarouter`, or the Python scripts `randomTrips.py` and `dua-iterate.py`. All of these files must be specified in a global configuration file (*.sumo.cfg) in order to be used with SUMO, as shown in Figure 2. When simulations are run with OMNeT++ and SUMO, the Veins framework is needed, as this implements the traffic control interface (TraCI) [28] application program interface (API) that allows for bidirectional communication between the two simulators. Communication is possible if the SUMO server program (`sumo` or `sumo-gui`) is running in the background (daemon), and this requires a knowledge of the SUMO simulation setup. This setup is written into another configuration file in XML format (*.launchd.xml), which is defined in the `omnetpp.ini` file (see Figure 2) and includes all the parameters necessary for network communications and the numerous other parameters needed by OMNeT++.

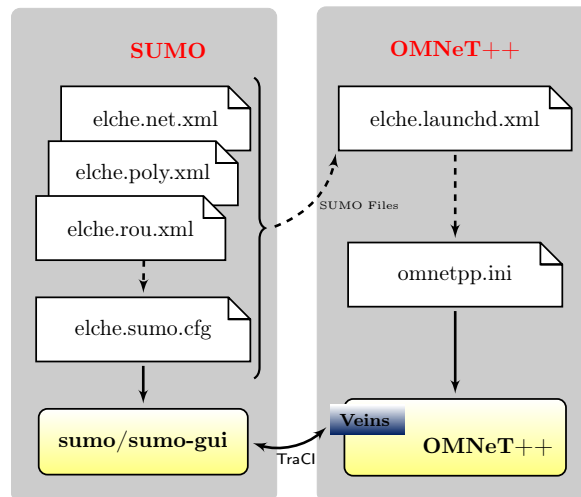


Figure 2. Configuration files for Simulation of Urban MObility (SUMO) and OMNeT++.

As can be seen, many configuration files are required in order to build a VANET network scenario; the task of building these scenarios including vehicle mobility is tedious, and mistakes are very frequently made during the configuration of the simulations, which can cause unexpected execution failures. The motivation for the development of the GatcomSUMO [29] application was therefore to offer a simple GUI to automate as many tasks as possible for the creation of scenarios and traffic demand for VANET environments, in order to smooth the learning curve for the proper use of the simulators and utilities associated with SUMO, OMNeT++ and Veins.

GatcomSUMO allows for the creation of both abstract and real network scenarios (Figure 3), including obstacles, and even for abstract grid networks, which are not supported by the netgenerate utility. In terms of traffic demand, the user can manually define both a list of vehicles and routes for them by selecting each edge in a sequential fashion. Once an edge has been selected, the subsequent edge in the route is then selected from a list that shows only the adjacent edges, and the route is visualised on the selected network map. In this way, the possibility of defining a discontinuous route is avoided. Another option is to automatically generate any number of routes between a specific or randomly chosen source and the destination edges, and possibly to include a list of intermediate edges that all routes must include. This is known as a trip in SUMO terminology. To properly generate the traffic demand, GatcomSUMO (i) invokes the utilities provided by SUMO for this task, thus avoiding the use of the system command line; and (ii) ensures the validity of the generated traffic demand, hence preventing simulation crashes at run-time. As mentioned above, when a route is manually created in an interactive way using GatcomSUMO, there is no possibility of inserting an unknown edge or creating a discontinuous route; that is, each edge in a route is followed by an adjacent edge.

Further issues that should be taken into account when generating valid traffic routes include the following: (i) a single route needs at least two edges, even if the vehicle's trip is very short or if the vehicle has stopped; (ii) the order within the routes file (*.rou.xml) is important, and all vehicles specified in this file must be sorted based on their departure time; (iii) it is necessary to check vehicles leaving the area defined for the network scenario, since run-time errors will arise if this condition is met; (iv) it would be helpful to define filters at route creation time, in order to select the desired lengths of the routes (i.e., less or greater than a specified value), or to specify the type of vehicle that is allowed to cross the edges, as a run-time error occurs if a vehicle tries to move along an edge that does not allow that type of vehicle (e.g., a pedestrian street).

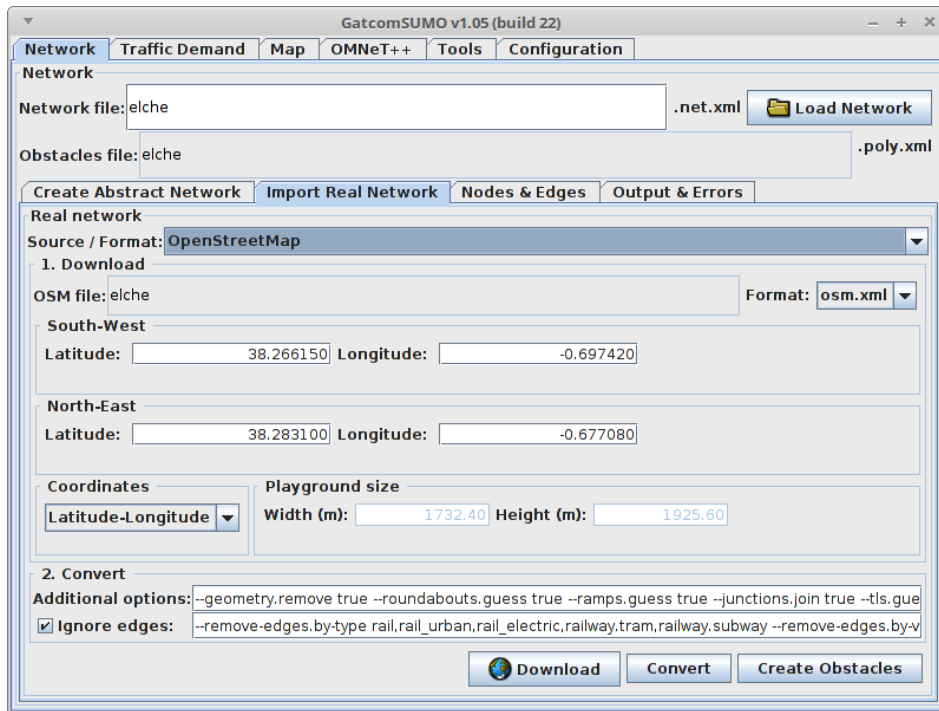


Figure 3. GatcomSUMO: Real network from OpenStreetMap.

All of these conditions are checked by GatcomSUMO in a way that is transparent to the user. The application is able to generate any number of random routes, and to ignore those that do not match all the conditions and filters that have been set up. Finally, all the generated routes can be pre-visualised on the map (see Figure 4).

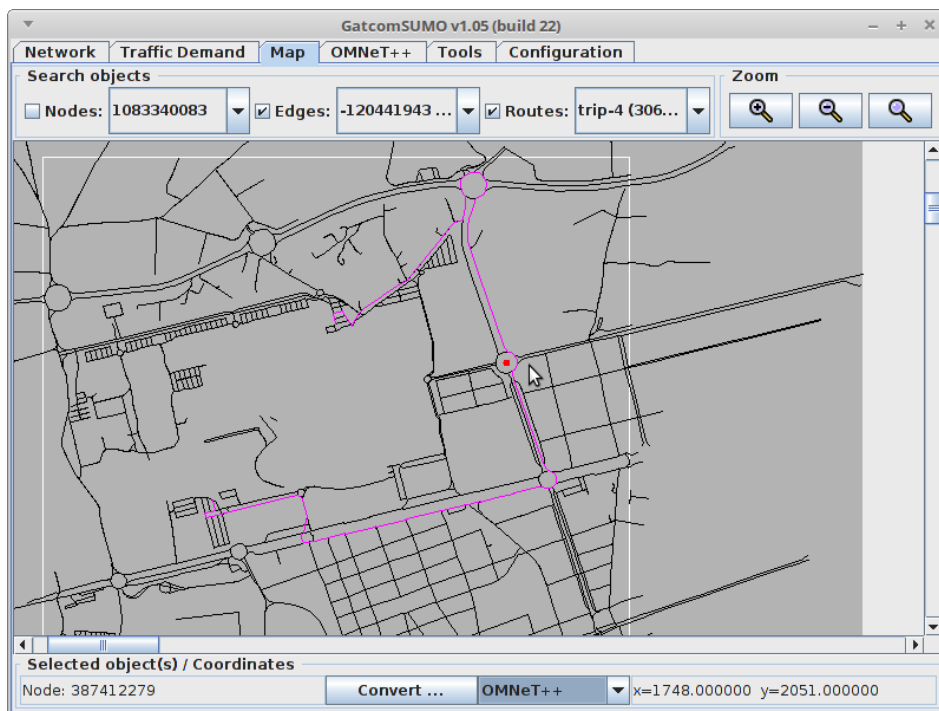


Figure 4. GatcomSUMO: Elche road network.

Another interesting feature of GatcomSUMO is the possibility of computing suitable coordinates for placing fixed objects such as a road-side units (RSUs). A fixed object can be placed in two ways:

(i) by defining a “mobile” vehicle without mobility in SUMO (with the maximum speed set to zero); and (ii) by defining an object within OMNeT++ (by specifying its static position in the `omnetpp.ini` configuration file). In the first approach, it is not possible to place the RSU in an exact position, as it will be located at the beginning of the first edge of the assigned route and oriented towards the second (a route therefore needs at least two edges). In addition, the main problem with this option is that the object will be considered a vehicle in the same way as the others; its presence may therefore affect the other vehicles and it may cause a traffic jam. Although SUMO allows vehicles to be parked alongside the road, Veins does not support this feature, and a run-time error arises when the simulation is run. In the second alternative, since the RSU is defined as an object in OMNeT++ (as an instance of the `BaseMobility` class), rather than as a Veins object, it can be placed in any position (e.g., a location with real coordinates). However, SUMO and OMNeT++ use different coordinate systems, which differ from real geodetic coordinates (latitude and longitude) and UTM coordinates. GatcomSUMO allows for a conversion between all of these coordinate systems, taking into account the bounding box and the necessary projection and translation of the corresponding values. This can be done by typing in the exact values to be converted, or simply by clicking on the map. As shown in Figure 4, the coordinates obtained for OMNeT++ can be written directly into the `omnetpp.ini` configuration file. One minor inconvenience is that the RSU is not shown in the graphical interface of SUMO (`sumo-gui`), so it is difficult to validate its position. However, this issue is solved by creating a SUMO point of interest (POI) using the ‘TraCIDemo11p’ module, which is shown as a small circle.

In addition to the network scenario and the traffic demand, GatcomSUMO also generates the remaining configuration files needed by the SUMO and OMNeT++ simulators, as shown in Figure 2. Finally, the application also includes a set of utilities that are useful for setting the appropriate values for certain parameters in the `omnetpp.ini` configuration file, such as unit converters (dBm-Watts, speed, etc.), and a communication range calculator that supports two physical models, based on the free-space path loss (FSPL) and two-ray approaches (Figure 5).

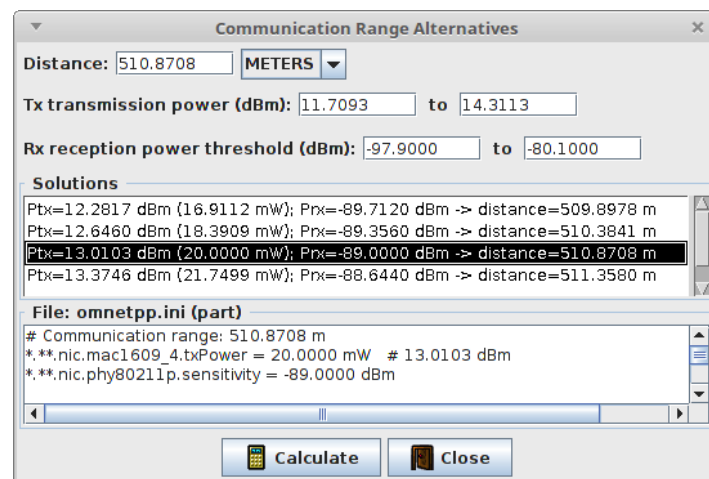


Figure 5. GatcomSUMO: Communication range utility.

GatcomSUMO is a general purpose application that is useful whenever SUMO is used, although it is particularly valuable when used in conjunction with OMNeT++ and Veins. If the scenario built with GatcomSUMO is intended to be used for the specific purpose of evaluating video transmission over VANETs, additional steps need to be taken before and after running the simulations in order to perform a realistic and detailed evaluation study of all the elements that can be found in a vehicular video delivery system. This formed the motivation for the development of GatcomVideo, which is introduced in the next subsection.

3.2. GatcomVideo

When evaluating the video transmission in a vehicular network, in addition to defining the network scenario and simulation models, other tasks need to be done. Firstly, the source video sequences must be encoded and packetised, to generate the corresponding video packet trace file to be used in the network simulations (pre-process). Then, after running the simulations, the video sequence needs to be reconstructed and decoded (post-process). All three of these steps, pre-process, simulation, and post-process, can be done with the GatcomVideo application, as illustrated in Figure 6. The graphical interface includes a set of tabs: one for each of these steps, plus an additional tab for the general configuration of GatcomVideo.

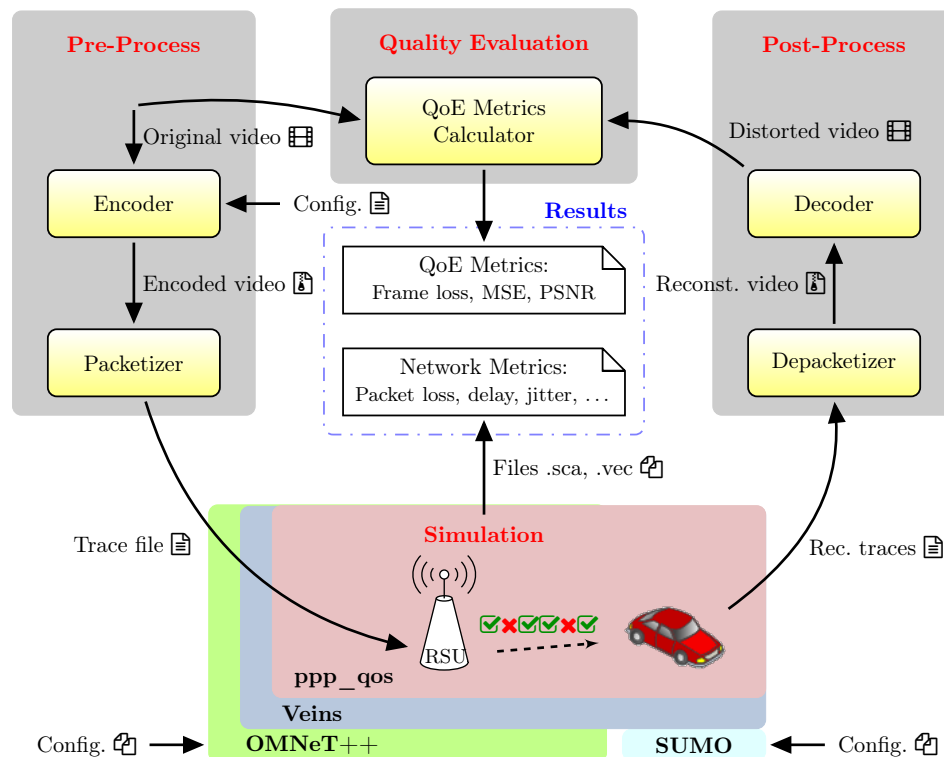


Figure 6. GatcomVideo: Workflow.

The pre-process step is intended to encode the video sequences with the HEVC standard. We have selected the HEVC codec because it is the last video coding standard and it is able to compress a video sequence yielding half the bit-rate than the previous standard H.264/AVC [30] at the same perceptual quality. However, the future video coding standard called Versatile Video Coding (VVC/H.266) is close to be finally approved and, in that moment, we aim to give support for this new video codec. Specifically, the HEVC reference software HM (HEVC Test Model) [31] is used, and it has been modified to include a module that performs an RTP [32] packetisation of the output video bit stream. As a result, a video packet trace file is generated [33], providing the information about each packet to be transmitted: A correlative packet number, the frame type which it belongs to (I, P, or B), the playback time, the packet size, the frame offset of the packet payload, and the total number of packets of the frame which it belongs to. All of these external programs are invoked from the GatcomVideo application, which is managed by the user in an interactive way. GatcomVideo allows to encode a video sequence with different parameters, such as the encoding mode (All-Intra, Low-delay P, etc.), the Quantization Parameter (QP), or the number of tiles per frame used [34] (see Figure 7). The input video sequences must be loaded in YUV video format, the most popular video format for uncompressed video. As YUV format does not contain video headers, additional info should be provided to properly read the YUV video sequence like (a) the total number of frames, (b) spatial frame

resolution (width \times height pixels), (c) bit-depth (number of bits per pixel), and (d) temporal resolution (frames per second). For our purpose, we have used several video sequences recommended by the HEVC common tests conditions reference [35]. As this step requires a great amount of computation, GatcomVideo is able to launch several tasks in parallel. So, depending on the number of available CPUs (cores), the required time may be significantly reduced. Even though the simulation framework is used in an interactive way from the GUI, these long duration task will be run in background.

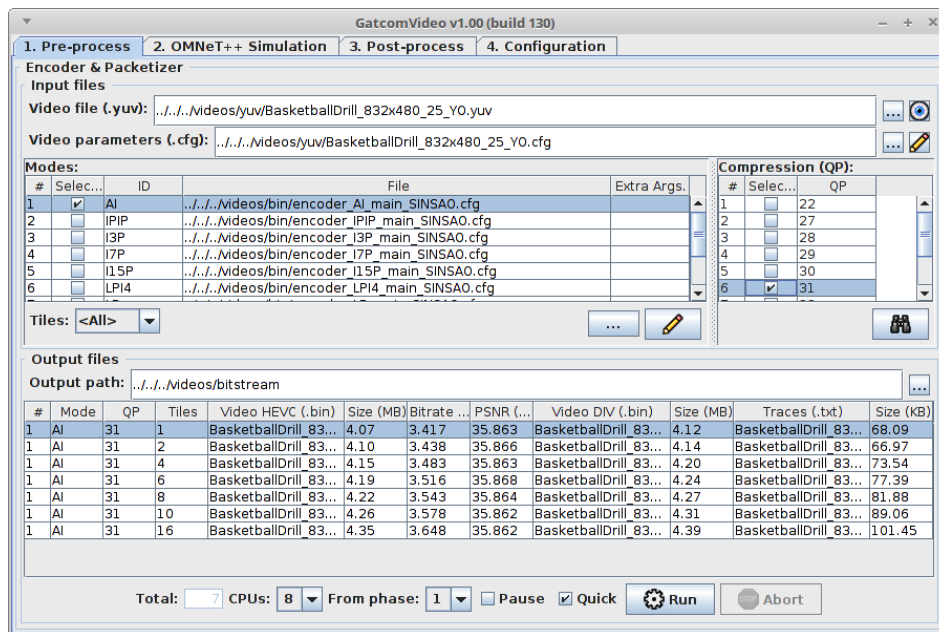


Figure 7. GatcomVideo: Pre-process.

In order to pass the packet trace file to the OMNeT++ simulator, we have extended the Veins framework with a new project ('ppp_qos' in Figure 6), since Veins does not support packet traces. This new simulation model contains a modification of the 'TraCIDemo11p' application and the Medium Access Control (MAC) layer included in Veins. Specifically, this project implements the following functionalities: a) defining input video packet traffic loads based on trace files to simulate the delivery of real video traffic from a particular video server (e.g., an RSU), b) generating output video trace files with the received video packets at each client node, c) defining synthetic background network traffic loads to drive the network to a particular load level, d) collecting statistics about the vehicles mobility (e.g., distance between any pair of nodes, number of neighbors, etc.), as well as many global and node network statistics at different levels (application, MAC and PHY), such as Load, Goodput, Packet Delivery Ratio (PDR), End-to-End Delay (EED), jitter, average number of collisions, etc. Also, when using traffic differentiation at the IEEE 802.11 MAC-level [36], the statistics are also grouped by Access Category (AC).

In the simulation step, GatcomVideo provides two additional valuable features: (a) the launch of the selected simulation runs and, (b) the graphical display of the simulation results. GatcomVideo shows a list of simulation sets extracted from the OMNeT++ configuration file, and then, the user can select which ones to launch, as well as the number of CPUs (cores) that will be used in parallel (see Figure 8). Previously, the user may start the SUMO server within the application without executing commands from the system console. At the end, a set of predefined graphs can be generated with different formats (.eps, .png), and resolutions (dpi), which are visualized within the application (see Figure 9). The graphs are generated by using both the R statistic package [37] and Gnuplot [38], but these programs are executed in background, that is, the user does not need to know anything about them nor to write any script. GatcomVideo allows to define any number of graphs by editing a simple ASCII text file or by using its own graphical wizard. Each graph can show results from a unique simulation, or it can

overlay data from multiple simulations. In addition, the above mentioned statistics are also averaged every second in order to allow an analysis of any specific part of the simulation (i.e., focusing only in a specific area of the vehicular network scenario). Finally, if each simulation run is repeated with more than one pseudo-random seed, the data shown in the graphs can be computed by averaging the results of all iterations, and if required, confidence intervals (CIs) can also be plotted.

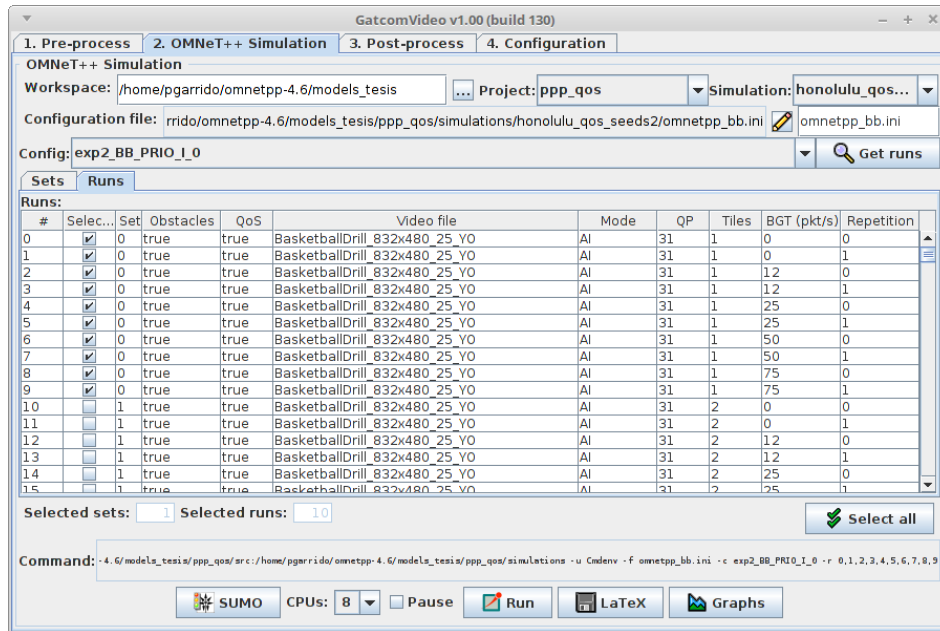


Figure 8. GatcomVideo: Simulation runs.

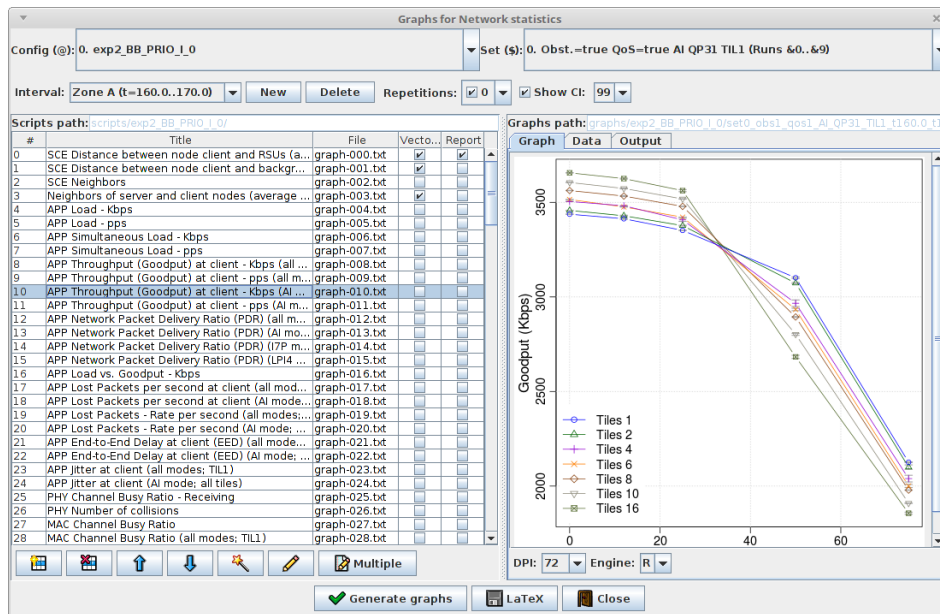


Figure 9. GatcomVideo: Graphs.

The main objective of the post-process step is to perform a quality evaluation of the received video sequences. This is achieved by reconstructing and decoding the received video sequences, and then by computing certain metrics to measure the objective quality of the video. In our experiments, since the video servers send each video sequence in a cyclic way, it was necessary to split the ordered list of received packets into individual video sequences. This could be done in the post-process tab (see Figure 10), where a data table can be built showing all individual video sequences received by

a particular client node; for each one, this indicates the simulated time at which it was received, its duration and the number of received and lost packets. In this way, the user can select the area(s) of interest, i.e., the received video sequences to be analysed. As mentioned previously, the user can also analyse the network statistics for any defined zone when generating the graphs for the simulation results. After this, the selected video sequences need to be reconstructed and decoded. This is done with a modified version of the HEVC (HM software) decoder, in which we have included a depacketiser and an EC functionality in order to minimise the impact of packet loss on the perceived quality. A technique known as frame copy concealment [39] was used, which is based on temporal prediction and replaces the missing parts of a single frame (or even the whole frame) with those found in the last correctly received frame. Once the selected video sequences have been decoded, an evaluation of the video quality is carried out based on certain quality of experience (QoE) metrics such as the frame loss ratio (FLR), the tile loss ratio (TLR), the mean squared error (MSE), and the PSNR.

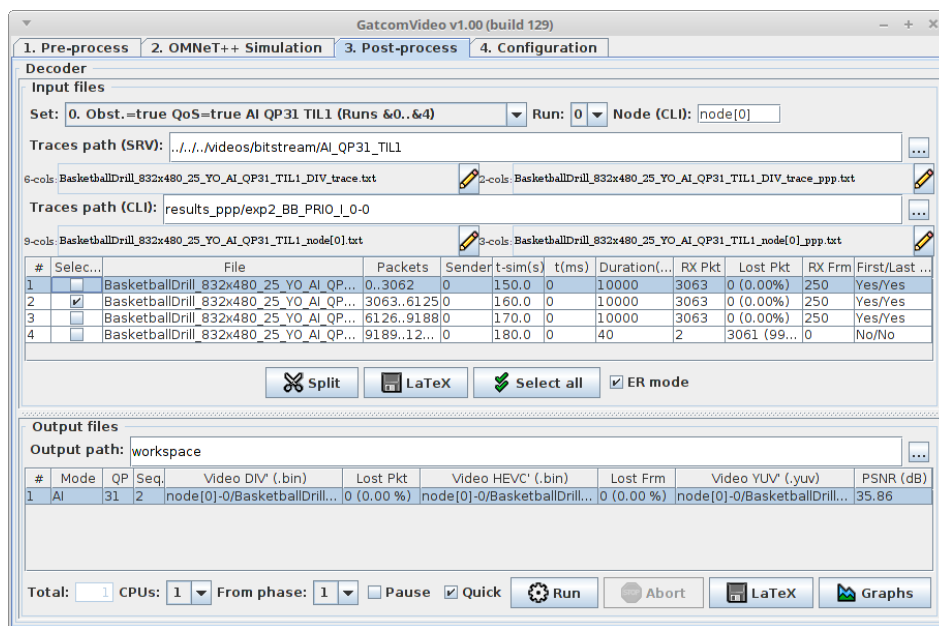


Figure 10. GatcomVideo: Post-process.

The MSE is computed for each frame n by averaging the differences in the squared intensity between the pixels (i, j) of the original (Y_S) and distorted (Y_D) frames, as shown in Equation (1), where $i \in 1 \dots N_{\text{col}}$ and $j \in 1 \dots N_{\text{row}}$, where N_{col} and N_{row} are the width and height of the video frames (in pixels). The PSNR is the metric most commonly used to measure the objective quality of a video. It is based on the MSE, and is computed frame by frame for the luminance (Y) component, as shown in Equation (2), and is averaged for all frames.

$$MSE(n) = \frac{1}{N_{\text{col}} \cdot N_{\text{row}}} \sum_{i=1}^{N_{\text{col}}} \sum_{j=1}^{N_{\text{row}}} [Y_S(n, i, j) - Y_D(n, i, j)]^2 \quad (1)$$

$$PSNR(n)_{dB} = 20 \cdot \log_{10} \left(\frac{V_{\text{peak}}}{\sqrt{MSE(n)}} \right) \quad (2)$$

Since this step is time-consuming, it was implemented using multithread programming, so that it can be executed in parallel depending on the available hardware resources, thus reducing the required processing time.

Finally, the user can define any number of graphs in a similar way to those generated for the network simulation. In addition to graphs, GatcomVideo can also generate text files in $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$ format at several places, so that these can be included in reports or manuscripts. These include tables with

relevant data about coded video sequences (in the pre-process tab), tables with a brief summary of the different simulation sets and parameters for each simulation run (in the simulation tab), or a report on the selected graphs along with their corresponding numerical data, to make it easier to analyse the results.

Since both GatcomVideo and GatcomSUMO were developed in Java, they are available for the main computing platforms (Windows, Linux, and Mac OS X). As described above, third-party software must be used with both GatcomSUMO and GatcomVideo, i.e., simulators (OMNeT++, SUMO, Veins), graphing packages (R, Gnuplot), and several binaries related to video processing (e.g., an encoder, packetiser, depacketiser, decoder, and others). Although some of these binaries are executables compiled for the Microsoft Windows operating system, the application can be configured for the proper execution of those binaries on any other platform, thanks to the use of WINE (Wine Is Not an Emulator) [40] (see Figure 11). This turns VDSF-VN into a multiplatform simulation framework based on open-source packages that allow for the evaluation of video transmission over VANETs by means of an efficient and ease-to-use GUI.

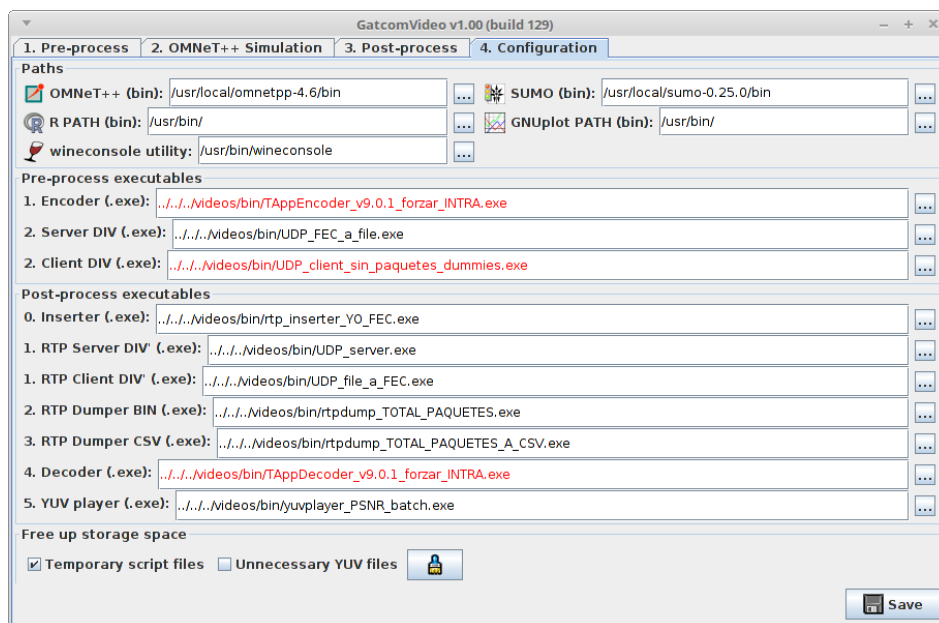


Figure 11. GatcomVideo: Configuration.

4. Conclusions and Future Work

In this paper, a simulation framework called Video Delivery Simulation Framework over Vehicular Networks (VDSF-VN) was introduced. This framework is intended to allow research to be conducted on video streaming over VANETs through simulation. VDSF-VN includes several command-line programs, and makes use of other utilities provided by the SUMO and OMNeT++ simulators. However, the user only needs to use two graphical applications: GatcomSUMO and GatcomVideo. Both applications invoke the other tools and programs in a transparent way.

VDSF-VN provides the following benefits to the user: (i) it facilitates the setup of the configuration files needed to conduct the simulations with OMNeT++ and SUMO, and generates them automatically from the graphical environment rather than using different command-line utilities or manually writing text files with strict syntax; (ii) it automates many tasks, thus avoiding the introduction of errors, particularly by inexperienced users; (iii) it substantially reduces the time needed to encode video sequences and run the simulations, which are highly time-consuming tasks, by launching the necessary jobs in parallel; and (iv) it reduces the effort involved in visualising the obtained results, as it allows the user to define and generate graphs within its own interface and to combine other statistics collected from any simulation run, thus avoiding the use of complex spreadsheets or other scripting languages

like R or Gnuplot (although both are used transparently by the applications). The level of expertise required is therefore lowered, while the time spent on each experiment is drastically reduced.

A preliminary core version of the VDSF-VN framework was used in previous studies, in which various experimental setups were used to test different aspects of the HEVC video encoder. For example, in [41–43], we applied techniques such as (i) intra-refresh video coding modes; (ii) frame partitioning (tiles/slices); and (iii) quality of service at the medium access control (MAC) level, in order to reduce the degradation in video quality produced by impairments arising from vehicular transmission. These works present the simulation results in the form of plots and tables provided by the VDSF-VN tool, giving consistent results that are coherent with those provided by other authors in the literature, and as a consequence from different simulation tools.

The development of VDSF-VN is a work in progress, and in future work, the following improvements are planned: (i) A new module for error protection techniques such as FEC; (ii) an adaptive video protection scheme based on the availability of network resources during the video delivery sessions; (iii) the optimisation of several tasks on specific supported platforms; and (iv) the development of support for next-generation video coding standards such as VVC.

Author Contributions: Funding acquisition, O.L.G.; Investigation, P.P.G.A.; Software, P.P.G.A. and P.P.; Supervision, M.P.M. and P.P.; Validation, M.P.M. and P.P.; Writing—original draft, P.P.G.A., M.P.M., P.P. and O.L.G.; Writing—review & editing, P.P.G.A., M.P.M., P.P. and O.L.G. All authors have read and agreed to the published version of the manuscript.

Funding: This work has been supported by Spanish Ministry of Science, Innovation and Universities under Grant RTI2018-098156-B-C54, co-financed by FEDER funds (MINECO/FEDER/UE).

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

References

1. Joshi, J.; Jain, K.; Agarwal, Y.; Deka, M.J.; Tuteja, P. VWS: Video surveillance on wheels using cloud in VANETs. In Proceedings of the 2015 IEEE 12th Malaysia International Conference on Communications (MICC), Kuching, Malaysia, 23–25 November 2015; pp. 129–134. [[CrossRef](#)]
2. Agarwal, Y.; Jain, K.; Karabasoglu, O. Smart vehicle monitoring and assistance using cloud computing in vehicular Ad Hoc networks. *Int. J. Transp. Sci. Technol.* **2018**, *7*, 60–73. [[CrossRef](#)]
3. Kumar, N.; Kaur, K.; Jindal, A.; Rodrigues, J.J. Providing healthcare services on-the-fly using multi-player cooperation game theory in Internet of Vehicles (IoV) environment. *Digit. Commun. Netw.* **2015**, *1*, 191–203. [[CrossRef](#)]
4. Liu, X.; Quan, H.; Zhang, Y.; Zhao, Q.; Liu, L. SVC: Secure VANET-Assisted Remote Healthcare Monitoring System in Disaster Area. *KSII Trans. Internet Inf. Syst.* **2016**, *10*, 1229–1248. [[CrossRef](#)]
5. Olaverri-Monreal, C.; Gomes, P.; Fernandes, R.; Vieira, F.; Ferreira, M. The See-Through System: A VANET-enabled assistant for overtaking maneuvers. In Proceedings of the 2010 IEEE Intelligent Vehicles Symposium, San Diego, CA, USA, 21–24 June 2010; pp. 123–128. [[CrossRef](#)]
6. Uhrmacher, A.M.; Brailsford, S.; Liu, J.; Rabe, M.; Tolk, A. Reproducible Research in Discrete Event Simulation: A Must or Rather a Maybe? In Proceedings of the 2016 Winter Simulation Conference WSC '16, Arlington, VA, USA, 11–14 December 2016; pp. 1301–1315.
7. Krajzewicz, D.; Erdmann, J.; Behrisch, M.; Bieker, L. Recent Development and Applications of SUMO—Simulation of Urban MObility. *Int. J. Adv. Syst. Meas.* **2012**, *5*, 128–138.
8. Kurczveil, T.; López, P.A. eNetEditor: Rapid prototyping urban traffic scenarios for SUMO and evaluating their energy consumption. In *SUMO 2015—Intermodal Simulation for Intermodal Transport*; Deutsches Zentrum für Luft und Raumfahrt e.V.: Berlin-Adlershof, Germany, 2015; pp. 137–160. [[CrossRef](#)]
9. Papaleondiou, L.G.; Dikaiakos, M.D. TrafficModeler: A Graphical Tool for Programming Microscopic Traffic Simulators through High-Level Abstractions. In Proceedings of the VTC Spring 2009—IEEE 69th Vehicular Technology Conference, Barcelona, Spain, 26–29 April 2009; pp. 1–5. [[CrossRef](#)]
10. Arellano, W.; Mahgoub, I. TrafficModeler extensions: A case for rapid VANET simulation using, OMNET++, SUMO, and VEINS. In Proceedings of the 2013 High Capacity Optical Networks and Emerging/Enabling Technologies, Magosa, Cyprus, 11–13 December 2013; pp. 109–115. [[CrossRef](#)]

11. Schweizer, J. *SUMOPy: An Advanced Simulation Suite for SUMO*; Lecture Notes in Computer Science (LNCS); Springer: Berlin/Heidelberg, Germany, 2014; pp. 71–82. [[CrossRef](#)]
12. Fogue, M.; Garrido, P.; Martinez, F.J.; Cano, J.; Calafate, C.T.; Manzoni, P. Using roadmap profiling to enhance the warning message dissemination in vehicular environments. In Proceedings of the 2011 IEEE 36th Conference on Local Computer Networks, Bonn, Germany, 4–7 October 2011; pp. 18–20. [[CrossRef](#)]
13. Varga, A.; Hornig, R. An Overview of the OMNeT++ Simulation Environment. In Proceedings of the 1st International Conference on Simulation Tools and Techniques for Communications, Networks and Systems & Workshops Simutools '08, Marseille, France, 3–7 March 2008; pp. 60:1–60:10.
14. Sommer, C.; German, R.; Dressler, F. Bidirectionally Coupled Network and Road Traffic Simulation for Improved IVC Analysis. *IEEE Trans. Mob. Comput.* **2011**, *10*, 3–15. [[CrossRef](#)]
15. Gocmenoglu, C.; Acarman, T.; Levrat, B. WGL-VANET: A web-based visualization tool for VANET simulations. In Proceedings of the 2015 IEEE International Conference on Vehicular Electronics and Safety (ICVES), Yokohama, Japan, 5–7 November 2015; pp. 62–63.
16. Riley, G.F.; Henderson, T.R. The ns-3 Network Simulator. In *Modeling and Tools for Network Simulation*; Springer: Berlin/Heidelberg, Germany, 2010; pp. 15–34. [[CrossRef](#)]
17. Barberis, C.; Gueli, E.; Le, M.T.; Malnati, G.; Nassisi, A. A customizable visualization framework for VANET application design and development. In Proceedings of the 2011 IEEE International Conference on Consumer Electronics (ICCE), Las Vegas, NV, USA, 9–12 January 2011; pp. 569–570. [[CrossRef](#)]
18. Barr, R.; Hass, Z.J.; van Renesse, R. JiST/SWANS: Java in Simulation Time/Scalable Wireless Ad hoc Network Simulator. Available online: <http://jist.ece.cornell.edu> (accessed on 21 November 2020).
19. Finnson, J.; Zhang, J.; Tran, T.; Minhas, U.F.; Cohen, R. A Framework for Modeling Trustworthiness of Users in Mobile Vehicular Ad-Hoc Networks and Its Validation through Simulated Traffic Flow. In Proceedings of the 20th International Conference on User Modeling, Adaptation, and Personalization UMAP'12, Montreal, QC, Canada, 16–20 July 2012; Springer: Berlin/Heidelberg, Germany, 2012; pp. 76–87. [[CrossRef](#)]
20. Klaue, J.; Rathke, B.; Wolisz, A. *EvalVid—A Framework for Video Transmission and Quality Evaluation*; Springer: Berlin/Heidelberg, Germany, 2003; pp. 255–272. [[CrossRef](#)]
21. ISO/IEC JTC1. ISO/IEC 14496-2. Coding of Audio-Visual Objects. 2001. Available online: <https://www.iso.org/standard/36081.html> (accessed on 21 November 2020).
22. ns-2. The Network Simulator. Available online: <http://www.isi.edu/nsnam/ns/> (accessed on 21 November 2020).
23. Peditakis, D.; Tselishchev, Y.; Boulis, A. Performance and scalability evaluation of the Castalia wireless sensor network simulator. In Proceedings of the 3rd International ICST Conference on Simulation Tools and Techniques, Malaga, Spain, 15–19 March 2010; p. 53. [[CrossRef](#)]
24. Rosário, D.; Zhao, Z.; Silva, C.; Cerqueira, E.; Braun, T. An OMNeT++ Framework to Evaluate Video Transmission in Mobile Wireless Multimedia Sensor Networks. In Proceedings of the 6th International ICST Conference on Simulation Tools and Techniques, Cannes, France, 5–7 March 2013; pp. 277–284. [[CrossRef](#)]
25. Saladino, D.; Paganelli, A.; Casoni, M. A tool for multimedia quality assessment in NS3: QoE Monitor. *Simul. Model. Pract. Theory* **2013**, *32*, 30–41. [[CrossRef](#)]
26. High Efficiency Video Coding (HEVC). ITU-T Recommendation H.265. 2013. Available online: <https://www.itu.int/rec/T-REC-H.265> (accessed on 21 November 2020).
27. Haklay, M.M.; Weber, P. OpenStreetMap: User-Generated Street Maps. *IEEE Pervasive Comput.* **2008**, *7*, 12–18. [[CrossRef](#)]
28. Wegener, A.; Piórkowski, M.; Raya, M.; Hellbrück, H.; Fischer, S.; Hubaux, J.P. TraCI: An Interface for Coupling Road Traffic and Network Simulators. In Proceedings of the 11th Communications and Networking Simulation Symposium CNS '08, Ottawa, ON, Canada, 14–17 April 2008; ACM: New York, NY, USA, 2008; pp. 155–163. [[CrossRef](#)]
29. Garrido Abenza, P.P.; Malumbres, M.P.; Piñol Peral, P. GatcomSUMO: A Graphical Tool for VANET Simulations Using SUMO and OMNeT++. In Proceedings of the SUMO User Conference 2017 (SUMO2017), Berlin-Adlershof, Germany, 8–10 May 2017; Volume 31, pp. 113–133.
30. Advanced Video Coding (AVC) for Generic Audiovisual Services. ITU-T Recommendation H.264. 2003. Available online: <https://www.itu.int/rec/T-REC-H.264> (accessed on 21 November 2020).
31. Joint Collaborative Team on Video Coding (JCT-VC). HEVC Reference Software HM (HEVC Test Model) and Common Test Conditions. Available online: <https://hevc.hhi.fraunhofer.de> (accessed on 21 November 2020).

32. Wang, Y.; Sanchez, Y.; Schierl, T.; Wenger, S.; Hannuksela, M. *RTP Payload Format for High Efficiency Video Coding*; RFC 7798; Internet Engineering Task Force: Fremont, CA, USA, 2016. [CrossRef]
33. Seeling, P.; Reisslein, M. Video Transport Evaluation With H.264 Video Traces. *IEEE Commun. Surv. Tutor.* **2012**, *14*, 1142–1165. [CrossRef]
34. Misra, K.; Segall, A.; Horowitz, M.; Xu, S.; Fuldseth, A.; Zhou, M. An Overview of Tiles in HEVC. *IEEE J. Sel. Top. Signal Process.* **2013**, *7*, 969–977. [CrossRef]
35. Bossen, F. Common test conditions and software reference. In Proceedings of the 11th Meeting of the Joint Collaborative Team on Video Coding (JCT-VC), Shanghai, China, 10–19 October 2012.
36. LAN/MAN Standards Committee of the IEEE Computer Society, *IEEE Standard for Information Technology—Telecommunications and Information Exchange between Systems Local and Metropolitan Area Networks—Specific Requirements—Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications*; IEEE Std 802.11-2016; IEEE: Piscataway, NJ, USA, 2016; pp. 1–3534. [CrossRef]
37. R Core Team. *R: A Language and Environment for Statistical Computing*; R Foundation for Statistical Computing: Vienna, Austria, 2016. Available online: <https://www.r-project.org/> (accessed on 21 November 2020).
38. Williams, T.; Kelley, C. Gnuplot 4.6: An Interactive Plotting Program. 2013. Available online: <http://gnuplot.sourceforge.net/> (accessed on 21 November 2020).
39. Bandyopadhyay, S.K.; Wu, Z.; Pandit, P.; Boyce, J.M. An Error Concealment Scheme for Entire Frame Losses for H.264/AVC. In Proceedings of the 2006 IEEE Sarnoff Symposium, Princeton, NJ, USA, 27–28 March 2006; pp. 1–4. [CrossRef]
40. Julliard, A. Wine. Available online: <https://www.winehq.org/> (accessed on 21 November 2020).
41. Abenza, P.P.G.; Malumbres, M.P.; Piñol, P.; López-Granado, O. Source Coding Options to Improve HEVC Video Streaming in Vehicular Networks. *Sensors* **2018**, *18*, 3107. [CrossRef] [PubMed]
42. Garrido Abenza, P.P.; Malumbres, M.P.; Peral, P.P.; López-Granado, O. Evaluating the Use of QoS for Video Delivery in Vehicular Networks. In Proceedings of the 2020 29th International Conference on Computer Communications and Networks (ICCCN), Honolulu, HI, USA, 3–6 August 2020; pp. 1–9. [CrossRef]
43. Abenza, P.P.G.; Peral, P.P.; Malumbres, M.P.; López-Granado, O. Simulation Framework for Evaluating Video Delivery Services over Vehicular Networks. In Proceedings of the 2018 IEEE 88th Vehicular Technology Conference (VTC-Fall), Chicago, IL, USA, 27–30 August 2018; pp. 1–5. [CrossRef]

Publisher’s Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.



© 2020 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<http://creativecommons.org/licenses/by/4.0/>).