DRIVE: A Digital Network Oracle for Cooperative Intelligent Transportation Systems

Ioannis Mavromatis*†, Robert J. Piechocki†‡, Mahesh Sooriyabandara*, and Arjun Parekh§
* Bristol Research and Innovation Laboratory (BRIL), Toshiba Europe Ltd., Bristol, UK
† Department of Electrical and Electronic Engineering, University of Bristol, Bristol, UK
‡The Alan Turing Institute, London, UK
§ British Telecom (BT) Group, Bristol, UK
Email: Ioannis.Mavromatis@toshiba-trel.com, R.J.Piechocki@bristol.ac.uk, Mahesh@toshiba-trel.com, Arjun.Parekh@bt.com

***Abstract*—In a world where Artificial Intelligence revolutionizes inference, prediction and decision-making tasks, Digital Twins emerge as game-changing tools. A case in point is the development and optimization of Cooperative Intelligent Transportation Systems (C-ITSs): a confluence of cyber-physical digital infrastructure and (semi)automated mobility. Herein we introduce Digital Twin for self-dRiving Intelligent VEhicles (DRIVE). The developed framework tackles shortcomings of traditional vehicular and network simulators. It provides a flexible, modular, and scalable implementation to ensure large-scale, city-wide experimentation with a moderate computational cost. The defining feature of our Digital Twin is a unique architecture allowing for submission of sequential queries, to which the Digital Twin provides instantaneous responses with the "state of the world", and hence is an Oracle. With such bidirectional interaction with external intelligent agents and realistic mobility traces, DRIVE provides the environment for development, training and optimization of Machine Learning based C-ITS solutions.**

***Index Terms*—C-ITS, CAV, Digital Twin, Simulation Framework, SUMO, Urban Mobility.**

I. Introduction

Next-generation Cooperative Intelligent Transportation Systems (C-ITSs) will bring the paradigm of Mobility-as-a-Service (MaaS) to a whole new level [1]. The C-ITS infrastructure network and the Connected and Autonomous Vehicles (CAVs) will share sensor data and maneuvering intentions in a heterogeneous Vehicle-to-Everything (V2X) fashion [2]. These interactions can enhance the environmental perception, enable co-operative driving and the whole host of additional mobility services [3].

The connectivity performance within C-ITSs and CAVs has been extensively researched [4], [5]. The lack of real CAVs and C-ITSs for experimentation is tackled through complex simulation frameworks [6]. Such an approach can reduce the immense cost of using large fleets of vehicles, and provides an easy and repeatable way to obtain near-perfect results. Based on the above, the concept of "Digital Twins" has been proposed in the past [7]. A Digital Twin attempts to replicate salient features and data flows of the real world for the problem at hand.

978-1-7281-8086-1/20/$31.00 © 2020 IEEE

Traditional wireless networking problems are recently tackled with Machine Learning (ML) techniques and advanced simulation models [8]. Traditional ML usually relies on existing datasets to train and evaluate a model. Later, the trained model is deployed on the end-system. However, C-ITSs can evolve over time, thus requiring more sophisticated decision agents that interact with the environment in real-time (e.g., Reinforcement Learning (RL) agents). Based on that, in this paper, we introduce the concept of a "Digital Network Oracle", an augmented "Digital Twin". A "Digital Network Oracle" introduces a particular architecture that allows the Twin to be queried sequentially, returning a snapshot of the state of the "virtual world". With such capabilities, the Twin can be used to train RL controllers for a swath of inferential and prediction tasks. A Digital Network Oracle, moves beyond static data and assets, and accesses interactions between entire systems covering vehicles, people, processes, and behaviors. Based on the above concepts, we developed our own Digital Network Oracle, called *Digital twin for self-dRiving Intelligent VEhicles (DRIVE)*, and we introduce it in the rest of the paper. Our framework is publicly available under `github.com/ioannismavromatis/DRIVE_Simulator`.

C-ITSs, are a great example of a System-of-Systems (SoS), and are information-centric, i.e., transport and processing of data and information in integral to their performance and dependable operations. With that in mind, we see that traditional simulation frameworks lack in several areas. For example, we observe a lack of supporting technologies (e.g., Veins framework [9] supports only IEEE 802.11p). Other frameworks significantly increase the computational complexity (e.g., Veins, iTetris [10], etc). Frameworks that address the complexity (e.g., Vienna 5G System Level Simulator [11]), lack in the bidirectional interactions with real-world mobility traces, or modularity and extensibility.

DRIVE is designed with the above in mind and can address several shortcomings of the existing frameworks. More specifically, we introduce a flexible and modular implementation, so the required level of realism is chosen, without wasting essential computational resources. Flexible evaluation mechanisms are developed that report systemic

changes and behaviors after a set of interactions. What is more, during the execution of the scenarios, bidirectional interaction with realistic mobility traces is provided [12]. Finally, uncertainty modeling and the unpredictability found in the real-world scenarios are introduced within the framework. Overall, DRIVE, addressing the above, can be used for holistically tackling C-ITSs problems.

The rest of the paper is organized as follows. In Sec. II we describe the key requirements for a Digital Network Oracle, its advantage, and potential use-cases. Sec. III describes the design and the implementation of the main DRIVE components and the benefits introduced. A reference scenario is described and evaluated in Sec. IV. Finally, the paper is concluded in Sec. V with some suggestions for the future.

II. Requirements for a Digital Network Oracle

Overall, a Digital Network Oracle provides a framework where intelligent agents can coexist with a realistic environment and interact in real time. Some potential use-cases when merging ML/RL and a Digital Network Oracle are:

- **Anomaly detection**: A sequence of observations is examined to identify points in the time series where unusual events may have occurred.
- **Performance analysis**: The accumulated Quality-of-Service (QoS) is of prime concern for C-ITSs. The accumulation horizon can vary, and individual rewards can describe the behavior. A typical use case might be a systematic analysis of different wireless environments.
- **Stress test analysis**: Manufacturing of a non-stationary distribution, designed to place an increasing demand on the system. In such a case, one would attempt to observe a QoS metric for possible failure points.

The above use-cases can be tackled by ML/RL techniques [8]. When combined, though, with the highly diverse vehicular environments, the required interactions introduce new obstacles in the existing simulation approaches. For a successful C-ITS evaluation, all the subsystems in this SoS (human users, vehicles, communication planes, etc.) should interact flawlessly and in real-time. Based on the above, some key requirements for a Digital Network Oracle are:

1) ***Repeatability***: It must serve as a real-time environment, generating information for an intelligent agent, and applying new policies. Randomness can potentially be introduced for further validation and increased realism, however, it must be controlled by the user (e.g., with seeds).

2) ***Flexibility and Modularity***: New systems or subsystems should be easily composed (e.g., a new communication plane could be introduced with a change in the configuration parameters). Different parameters must be exposed to adjust between the realism and the execution speed. For example, adjusting the map resolution could benefit the rapid prototyping and debugging (when decreased) or enhance the realism (when increased).

3) ***Bidirectional Interaction with Intelligent Agents***: The framework should handle the training data as well as receive and apply the policies generated by intelligent agents. The framework should provide corresponding data interfaces for interaction with existing libraries.

4) ***Bidirectional Interaction with Traffic Management Tools and Maps***: The framework must provide bidirectional interaction with mobility models (both vehicles and pedestrians) and manipulation of real-world maps. Common formats should be agreed between the traces and the maps.

5) ***Robust evaluation mechanisms***: All intelligent agents should be linked with existing evaluation mechanisms. The status of assigned tasks (e.g., finished, pending, errors) should be exposed as well as the change in the performance introduced by a new policy.

6) ***Parallel sampling ability***: The agent should query the environment (exploration) and adjust itself by the information and feedback received. Parallel sampling could be provided to increase the training efficiency.

7) ***Minimal overhead and faster execution***: The framework must mitigate the computational complexity by introducing minimal overhead and ways of exploiting all the system resources (e.g., parallelization of tasks).

A. Limitations of Existing Vehicular Simulation Frameworks

There are several well-known simulation frameworks used for vehicular communications and C-ITS scenarios. The features of these frameworks vary in terms of the coupling (unidirectional or bidirectional) between mobility traces (realistic or artificially generated) and vehicular communication planes. With regards to the communication links, they are either described in a fine-grained fashion (packet-level) or focus more on the global behavior of large cyber-physical systems (system-level). Table I gives an overview of the existing popular frameworks.

Starting with the packet-level network simulators, and as shown in Table I, all frameworks except Netsim [13], provide integration with SUMO traffic generator [12], whereas, Netsim artificially generates the mobility traces used. Furthermore, Veins [9], iTetris [10] and VSimRTI [15] allow online re-configuration and re-routing of vehicles in reaction to network packets. Finally, all packet-level simulators implement all or a number of the Open Systems Interconnection (OSI) layers. That allows for fine-grained experimentation and results in excellent agreement with the real world. However, as shown in [6], due to the assessment of the propagation characteristics on a per-packet-basis, city-scale scenarios and just a few minutes of real-world time, could easily result in days of simulation time.

The computational complexity is being addressed by Vienna [11] and GEMV^2 [16] frameworks. Both reduce the complexity introducing a simplified propagation assessment scheme. The Vienna simulator implements a concise "map" feature. whereas, GEMV^2, can parse real-world maps and traffic traces, but lacks the bidirectional interaction with them, a feature found in Veins or iTetris for example.

As a general observation, and based on the requirements introduced before, it is evident that the existing frameworks lack essential features required for a systemic C-ITS ex-

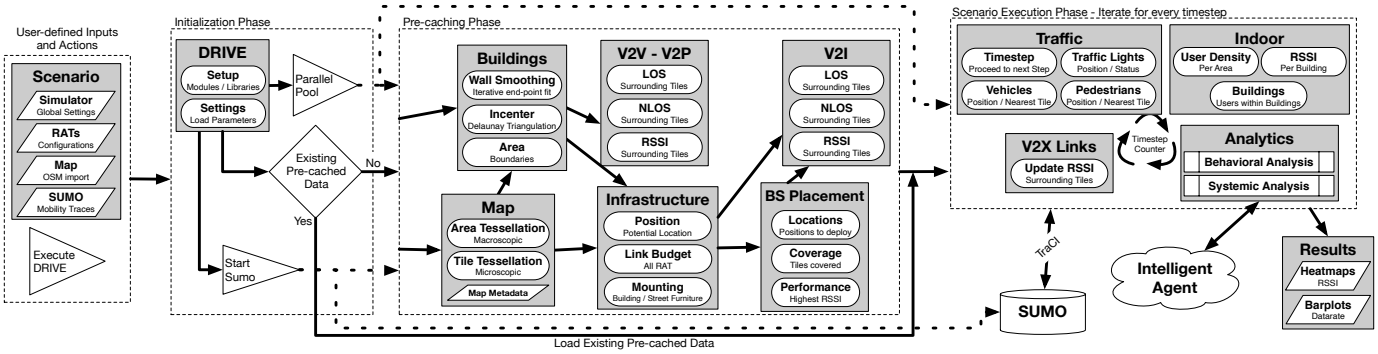


Figure 1. A high-level framework design of DRIVE, with the steps followed during the initialization, pre-caching, and scenario execution phases. The user inputs several parameters via configuration files, and the framework handles the different interactions later. A pre-caching phase can speed up the consecutive executions, while interactions with mobility traces and intelligent agents can introduce new avenues of realistic C-ITS evaluation.

Table I
STATE-OF-THE-ART FRAMEWORKS FOR SIMULATION OF VEHICULAR COMMUNICATIONS AND COMPARISON WITH DRIVE.

Simulation Framework	Open Source	Mobility Interaction	Traces – Maps	Level	Supported Wireless Technologies	Core Framework	Intel. Agent Interaction	Scalability
Veins [9]	Yes	Bidirectional	Real-world	Packet	IEEE 802.11p	OMNeT++	No	Poor
INET	Yes	Bidirectional	Real-world	Packet	IEEE 802.11x, Cellular	OMNeT++	No	Poor
iTetris [10]	Yes	Bidirectional	Real-world	Packet	IEEE802.11p, WiMAX	NS3	No	Poor
Netsim [13]	No	Unidirectional	Artificial	Packet	IEEE 802.11p	Standalone	No	Poor
TraNS [14]	Yes	Bidirectional	Real-world	Packet	Generic Wireless Plane	NS2	No	Poor
VSimRTI [15]	No	Bidirectional	Real-world	Packet	IEEE 802.11p, Cellular	OMNeT++, NS3	No	Poor
Vienna 5G [11]	Yes	Unidirectional	Artificial	System	Cellular/5G	Standalone	No	Fair
GEMV^2 [16]	Yes	Unidirectional	Real-world	System	IEEE 802.11p	Standalone	No	Good
DRIVE	Yes	Bidirectional	Real-world	System	Technology-agnostic	Standalone	Yes	Great

perimentation. For example, their functionality is tied to specific communication technologies (e.g., GEMV^2 is based on just IEEE 802.11p), or the integration of new communication planes becomes convoluted (e.g., integration of LTE within Veins). Additionally, several frameworks do not provide interactions with real-world maps and mobility traces (e.g., Vienna Simulator and Netsim), limiting the synergy of mobility and communication links. All frameworks but Veins, limit the mobility traces to just one type of vehicle and do not consider pedestrians or indoor users. Finally, many introduce increased complexity, thus running scenarios in conjunction with intelligent agents becomes a very time-consuming activity.

III. DESIGN AND IMPLEMENTATION OF DRIVE FRAMEWORK

The above limitations show the obstacles introduced when evaluating realistic and large-scale C-ITS scenarios with the existing simulation frameworks. This is the gap that DRIVE intents to fill. DRIVE is a framework where all decision processes, routes, and interactions are determined at runtime. It can enhance the simulation of large-scale C-ITS city scenarios, where all traffic participants, and the communication infrastructure interact through various means of communication links. The framework is written in MATLAB and is open-sourced to enable the design of extensions by other users. In addition, MATLAB provides integration with all common programming languages (Python, C++, etc.), so libraries written in them can be easily utilized within our framework. A high-level visualization diagram of the developed framework can be found in Fig. 1.

DRIVE introduces a pre-caching feature not found in other simulation frameworks that can significantly enhance the user experience (Fig. 1). More specifically, static information such as building positions, potential BS placement, and potential communication interactions are calculated on the first run and stored for future usage. This ensures that when a user re-runs a scenario, the execution time will be minimized. In the next sections, we describe the information that is pre-cached or calculated on every run and how DRIVE operates under different scenarios.

DRIVE operates in two modes. The first one does not consider the mobility of vehicles and pedestrians, and the scenarios are based on just a given OSM map. This mode is ideal for evaluating different communication solutions with simplistic user densities per city block, as in our work [17], where a city-scale infrastructure placement scheme was proposed. The second mode provides a more realistic C-ITS implementation. It introduces a bidirectional interaction with SUMO traffic generator [12] for both vehicle and pedestrian traffic. SUMO interacts with our framework via the TraCI4Matlab framework [18]. As before, different agents can be designed that will interact with the environment in various ways and devise optimal policies (e.g., a cell switch-on switch-off mechanism can be designed based on the traffic density on the roads). In the next sections, we describe in more detail the different components of DRIVE.

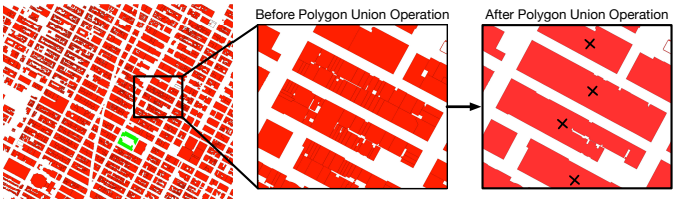


Figure 2. Example of a parsed OpenStreetMap file, the polygon union operation on two example city blocks and their incenters.

A. Building Manipulation

The Line-of-Sight (LOS) and Non-LOS (NLOS) communication links are dictated by the environmental obstacles, these usually being the buildings within cities. The propagated signal severely attenuates when intersecting with an obstacle. Especially for higher frequencies (e.g., Millimeter Wave (mmWave) frequencies of $>28\,\text{GHz}$), usually, a single intersection with a wall almost completely blocks the signal. Usually, the calculation of the signal attenuation requires increased computational resources and introduces bottlenecks for the existing simulation frameworks [6]. Within DRIVE we introduce a pre-caching phase (Fig. 1) and several building manipulation strategies (Fig. 1-*Buildings*) to minimize the execution time, without the loss of accuracy.

Under urban scenarios, building blocks consist of buildings with adjacent tangent sides or small negligible gaps between them (Fig. 2). Therefore, buildings on a map are represented as 2D or 3D *Simple Polygons (SPs)*. An SP is considered a flat-shaped object consisting of straight, non-intersecting line segments. These lines, when joined pairwise, form a closed path. The overall signal attenuation of a ray is the summation of the attenuation introduced by all intersecting building walls. Reducing the number of walls, the number of intersections is reduced as well. For DRIVE, we concatenate the side-by-side buildings using the polygon union operation [19] and remove the holes introduced (e.g., a courtyard) to form solid objects (Fig. 2).

To further reduce the complexity, we introduce a smoothing of the buildings layout. Our approach decimates a curve composed of line segments, to a similar curve with fewer points based on a simplification tolerance parameter. Modifying this parameter, a user can decide the level of simplification required. For our implementation, we utilized the Douglas–Peucker iterative end-point fit algorithm. Reducing the number of polygon edges reduces the LOS and NLOS calculations for the entire city map. What is more, DRIVE introduces a 3D city environment. All building polygons inherit their height from OpenStreetMap (OSM) metadata [20]. If the building height is missing, a value is assigned at random from a range controlled by the end-user. In our framework, there is also a provision to control the randomness with seeds to ensure the reproducibility of the results. The above simplifications align with Reqs. 1, 2 and 7 introduced before. Based on the above, DRIVE can still achieve realistic performance from the system-level perspective, minimizing the computation time.

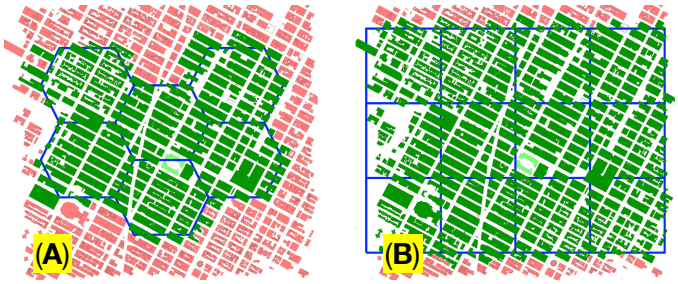


Figure 3. Examples of the tessellation approach. (A) shows the hexagonal tessellation while (B) shows the squared approach.

B. Interaction between Maps and Intelligent Agents

As discussed in Sec. II, one of the key requirements for a Digital Network Oracle is the bidirectional interaction with Intelligent Agents. Agents are responsible for generating policies that will be deployed on given scenarios. Lately, "self-organized" Multi-Agent Systems (MASs) are widely investigated. MAS is a network of agents who generate localized policies, having though as a goal to maximize the global reward.

A MAS-compliant architecture is established in DRIVE. We introduce a map tessellation in areas of equal size, in a two-way fashion (Fig. 1-*Map*). Individual policies can be applied on each area. The areas can have either a squared or a hexagonal shape (Fig. 3). The first is optimized for speed, while the latter provides a more realistic representation of the real world (e.g., the LTE BS coverage region is usually represented as a hexagonal pattern in the literature). All buildings with at least one edge within an area are considered part of this area. The benefits of such an approach are two-fold. Initially, all the interactions-of-interest can be focused on specific areas. That can reduce the execution time, without the need for reconfiguring the entire scenario. Most importantly, having a MAS intelligent scheme, we can apply different policies in different areas. Each individual agent can run locally and make decisions for specific areas. The above align with the Reqs. 2, 3, 6 and 7. An example of that can be found in our previous work [17].

A similar tessellation is followed in the microscopic world as well, with the map being discretized in tiles (Figs. 4 and 1-*Map*). A tile is considered a small uniform area with the same properties on its entire surface. For DRIVE, all the calculations take place using the incenters of the given tiles to speed up the performance. Using a relatively small tile size (e.g., $\leq 4\,\text{m}$), a very realistic result can be achieved, significantly reducing the computational complexity (Req. 7). More information about that can be found in [17].

C. Communication Planes and Potential BS Positions

DRIVE supports two "cell-like" types of communication planes. i.e., macrocell and femtocell, with the main difference being the mounting point of the BSs (Fig. 1-*Infrastructure*). For the macrocell plane, the BSs are positioned on top of the buildings, placed at the incenter of the concatenated building blocks (Fig. 2). All buildings are considered potential BS positions during a simulation

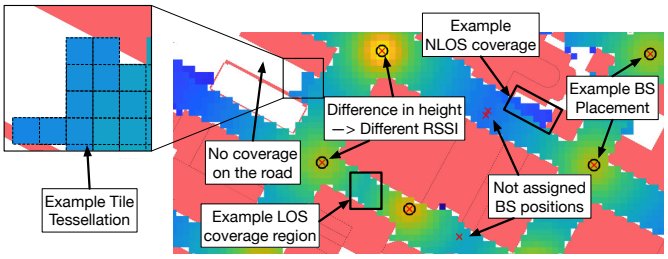


Figure 4. Example operational output. The heatmap shows the calculated RSSI per tile, the chosen femtocell BSs and the unassigned positions.

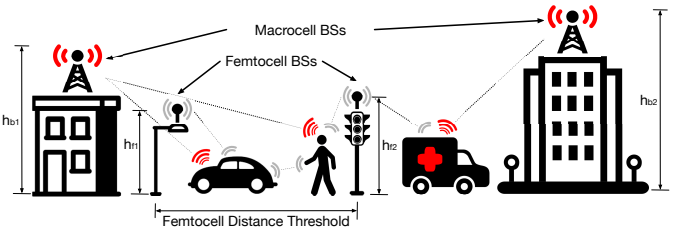


Figure 5. High-level system representation. The different types of vehicles and the pedestrians, equipped with a number of communication devices, choose the links that will be formulated based on an intelligent decisions.

scenario (Fig. 1-*BS Placement*). The macrocell plane is intended to be used for long-range high-power communication technologies (e.g., LTE, etc.).

The femtocell BSs are mounted on street furniture (Figs. 4 and 5). The traffic light locations are imported from the OSM map metadata. However, OSM does not provide information about lampposts positions (Fig. 4). As a solution, for a given road, we artificially generate the lamppost positions by finding the road junctions and discretizing the road to equal segments, based on a user-defined value. The segments intersection point created are the potential lamppost positions (on both sides of the road). The above two sets of locations form the potential femtocell BS positions (Figs. 4 and 1-*BS Placement*)). The femtocell plane is intended to be used for short-range communication technologies (e.g., IEEE 802.11p, mmWaves, etc.).

The end-user can create different communication planes providing a configuration for each one with various wireless communication parameters for these cell types, i.e., carrier frequency, antenna gains, noise floor and figure, channel bandwidth, etc. DRIVE already implements some well known pathloss models (freespace, shadow fading described in [21], and the urban macrocell model described in 3GPP TR 36.873) but users can define their own models as well (Fig. 1-*Infrastructure*). The above provide a robust evaluation framework as the different performance indicators can be thoroughly evaluated.

Similarly, as in Sec. III-A, the traffic lights, and the lampposts are assigned a height from a uniform random distribution within a given range of values (Fig. 5). At the moment, the vehicles are not considered as blocking objects. In the future, the 3D environment will be extended, with the blockage of moving vehicles as their height could affect the LOS or NLOS V2I links (Fig. 1-*V2I*). All the above align with the Reqs. 1, 2, and 5 mentioned before.

DRIVE provides a unique feature compared to many other frameworks. The objects, systems and interactions described in Secs. III-A, III-B and III-C are pre-cached before the interaction with a mobility model. Information such as the potential V2X interactions and their LOS/NLOS nature are pre-calculated using the existing information (Fig. 1-*V2I/V2V-V2P*). Pre-calculating these information, and recalling them during the interaction with mobility traces can significantly reduces the execution time. To enhance the repeatability and the user experience, tools that store and load these information are provided. Utilizing such an approach, the scenario can run significantly faster after the first time as all the required information are already pre-calculated. These features align with the Reqs. 1 and 2.

D. Interaction with Mobility Traces and Indoor Users

DRIVE can handle two sets of "end-users". Initially, DRIVE approximates the traffic density as a spatial Weibull distribution. The calculated distribution can be updated periodically generating realistic spatial traffic patterns for cellular networks can be achieved [22]. The use of that is two-fold. When the vehicles and pedestrians are not considered, the generated distributions represent both indoor and outdoor users. On the other hand, when mobility traces are generated via SUMO, only the indoor user are based on this distribution.

For the latter, DRIVE provides a bidirectional interaction with the SUMO framework via TraCI4Matlab. The fuzziness in the SUMO route generation is controlled using seeds. During the execution time (Fig. 1-*Traffic*), the position of all vehicles is acquired from SUMO, and the nearest tile incenter is calculated. Assigning vehicles to tile incenters, we can leverage from the pre-cached information for the V2X links and almost instantaneously find their expected V2X link performance. When updates happen in the network configuration (e.g., the TX power of a BS is increased), DRIVE updates the required potential links in real-time (Fig. 1-*V2X Links*). Using the above architecture, DRIVE provides a flexible framework for interacting with intelligent agents and apply policies based on systemic behaviors. An example use-case can be the redirection of an emergency CAV from a particular route providing high-quality 5G connectivity to enable tactile robotic telesurgery. The above mentioned features align with the Reqs. 1, and 4.

IV. Proof of Concept Evaluation

In this section, we present the setup and the results of our proof of concept evaluation. Our reference map is a city area of Manhattan, USA. During the run time, the indoor users are picked from a random distribution, as discussed in Sec. III-D, while the outdoor users (vehicles and pedestrians) are generated using SUMO. We utilized two different communication planes, i.e., a macrocell plane based on 3GPP LTE, and a femtocell plane based on mmWaves. All the simulation parameters (chosen map, mobility traces, communication links) are summarized in

Table II
EXAMPLE MAP AREA AND LIST OF SIMULATION PARAMETERS.

Urban Area	**Manhattan, NY, USA**
Center	$-73.9841°W, 40.75545°N$
Map Size	2.274 km × 2.607 km
Indoor Users	≤ 100 per building
Outdoor Users	200 vehicles / 200 pedestrians

Communication Plane	**Macrocell**	**Femtocell**
Example Technology	LTE	MmWaves
Transmission power	20 dBm to 43 dBm	15 dBm to 25 dBm
TX / RX Antenna Gain	18 dBi / 0 dBi	22.6 dBi / 22.6 dBi
Carrier Frequency	2.6 GHz	60 GHz
Bandwidth	20 MHz	2.16 GHz
Propag. Model LOS / NLOS	3GPP TR 36.873 / COST Hata Model	Shadow Fading
Path-Loss Exp. LOS / NLOS	–	2.66 / 7.17
Antenna Beamwidth	120°	15°
Distance Separation	–	100 m
BS Height	15 m to 50 m	5 m to 15 m

Tab. II. As a scenario, we consider a simplistic intelligent agent that, based on the user density per timestep, adapts the TX power of the deployed BSs to enhance the datarate.

A. Evaluation of the Pre-caching Execution Time

As discussed in Sec. II-A, one of the biggest drawbacks of the existing frameworks is their execution time. DRIVE, being highly optimized, manages to achieve rapid simulation time without loss in realism. In Fig. 6, we present the time required for pre-caching all the V2X LOS and NLOS interactions for our example scenario. We evaluated our framework for five different grid-like tessellation sizes, i.e., $\{4, 8, 12, 16, 20\}$ m. The simulated time is 200 s, and the mobility traces consist of 200 vehicles and 200 pedestrians controlled by SUMO. The chosen SUMO timestep length was 1 s. For our evaluation, we used a system with macOS 10.15, an 8-core Intel i9 2.3 GHz CPU, and 16 GB of RAM.

As expected, when the tessellation size increases, the pre-caching time increases as well. For fine-grained simulations (tile size ≤ 4 m), a big increase in the execution time is observed. This is due to the small RAM size of the used machine. For large-scale scenarios, like this one, and small tile sizes, at least 32 GB of RAM is recommended. Overall, the execution time is always kept at reasonable levels. As shown, after the initial execution, the simulation length is significantly decreased. For example, the 4 m-tile requires 25 s compared to the initial 878.1 s. The pre-caching feature introduced can enhance the repeatability of the experimentation (e.g., when a bug is found in an under-development intelligent agent). The time needed during the scenario execution is significantly decreased as well. As shown 200 s of real-world time required roughly 54 s of simulation time within DRIVE. These results, compared to the hours of simulation time required by other frameworks [6], can significantly enhance the end-user experience and reduce the overhead introduced.

B. Example Use-case Scenario

In this section, we describe the actual execution of the scenario and how DRIVE handles the different interactions.

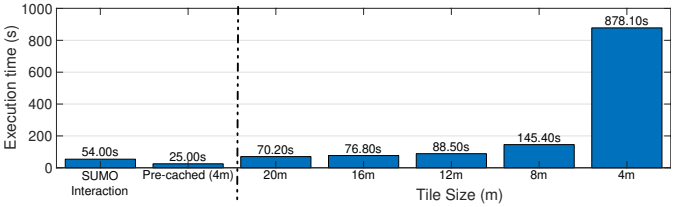


Figure 6. Execution time for different pre-caching scenarios. The first barplot shows the time required for the SUMO-DRIVE interaction (no other functions executed), the second shows the time needed for loading the pre-cached scenario information, and the rest show the overall time required for the "pre-caching phase" and for different tile sizes.

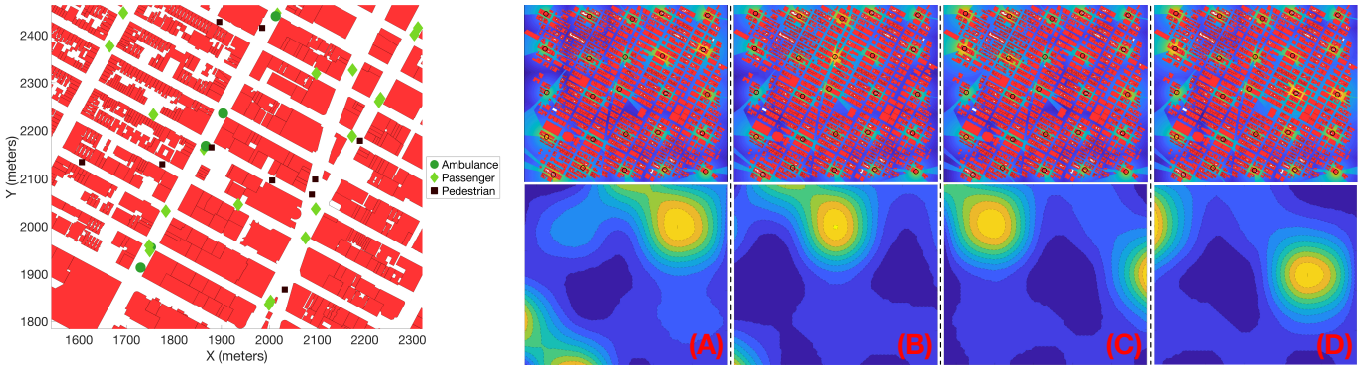
We choose the BS on the map as follows (Fig. 4). The macrocell BSs are equally spread with ~500 m distance separation. Their exact location is the incenter of the nearest building block (Fig. 2). The femtocell BSs are placed using a greedy addition algorithm. Starting from the BS that covers the most tiles, we add BSs until 90% of the city tiles are in LOS or NLOS coverage (as in [17]).

For our use-case, we discretized the map into 200 m × 200 m areas (as in Fig. 3). Later, we find the highest and lowest user densities for all areas, calculated as the number of vehicles, pedestrians, and indoor users within this area. A snapshot of the different vehicles and pedestrians positions is shown in Fig. 7a. Having these densities, we later adapt the TX power of our BSs to increase the perceived datarate. More specifically, we assume that the macrocell BSs in close proximity to the area with the lowest density is configured with a TX power of 20 dBm. On the other hand, the BS in areas with the highest density is tuned at 43 dBm of TX power. Proportionally, we configure the rest of the BSs. Similarly, we configure the femtocell BSs as well using a TX power range between 15 dBm to 25 dBm. For the ground truth, we set all the BSs to the median value, i.e., 31.5 dBm for the macrocell BSs and 20 dBm for the femtocell ones.

In Fig. 7, we see four instances of the executed scenario. For each one, the user density was calculated and visualized for the entire city (Fig. 7b-bottom). As shown, DRIVE provides progressively calculates changes the user density on the map. Fig. 7b-top shows the corresponding RSSI heatmaps for the macrocell plane. As discussed before, the TX power is adapted based on the density. In Fig. 8, we see the datarate for the static and the adaptive TX power scenarios. As expected, when the TX power is increased in areas with more users, the datarate is being increased as well. Even though this is a very simplistic use-case, it demonstrates very well the capabilities of DRIVE. As discussed before, our framework was designed to provide a flexible and fast way of interacting with intelligent agents, deploying policies, and evaluating performance improvements. Utilizing more sophisticated agents in the future, several C-ITS use-cases can be evaluated in an SoS-fashion.

V. CONCLUSIONS AND FUTURE WORK

In this paper, we described DRIVE. DRIVE is a city-scale Digital Network Oracle for C-ITS experimentation. Our solution is developed with an SoS ecosystem in mind



(a) Snapshot of vehicles and pedestrians positions.

(b) Example of the adaptive TX power for macrocell BSs and four different time instances.

Figure 7. An intelligent agent adapts the TX power based on the user density. Fig. 7a shows an example mobility trace of the two generated vehicle types and pedestrians. Fig. 7b-top shows the heatmap with the perceived RSSI and Fig. 7b-bottom shows the user densities.

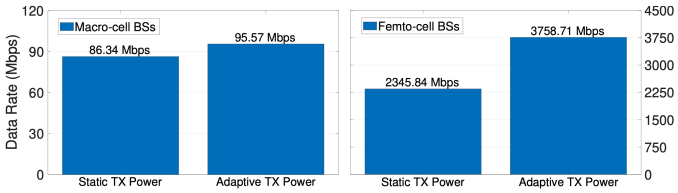


Figure 8. Comparison between the static and adaptive TX power scenarios.

and provides a robust environment for large-scale C-ITS-related scenario evaluation. Allowing an easy and adaptive bidirectional interaction with real-world mobility traces and intelligent agents, DRIVE can be used for several use-cases spanning from cybersecurity within C-ITSs, to behavioral system analysis based on various conditions. Furthermore, being highly flexible, modular, and scalable can simulate city-scale realistic scenarios within minutes. Overall, DRIVE was designed to tackle the problems of the existing simulation frameworks and enhance the prototyping of intelligent solutions for pressing concerns of the next-generation C-ITSs. In the future, DRIVE will be integrated with ML intelligent agents, and several C-ITS use-cases will be evaluated.

Acknowledgment

This work is funded in part by Toshiba Europe Ltd., in part by the Next-Generation Converged Digital Infrastructures (NG-CDI) project, supported by BT Group and EPSRC (EP/R004935/1), and in part by the CAVShield project (Innovate UK, no. 133898).

References

[1] R. Giesecke, T. Surakka, and M. Hakonen, "Conceptualising Mobility as a Service," in *Proc. of International Conf. EVER 2016*, Apr. 2016, pp. 1–11.

[2] K. Zheng, Q. Zheng, H. Yang *et al.*, "Reliable and Efficient Autonomous Driving: The Need for Heterogeneous Vehicular Networks," *IEEE Commun. Mag.*, vol. 53, no. 12, pp. 72–79, Dec. 2015.

[3] N. Lu, N. Cheng, N. Zhang *et al.*, "Connected Vehicles: Solutions and Challenges," *IEEE Internet Things J.*, vol. 1, no. 4, pp. 289–299, Aug. 2014.

[4] J. Wang, J. Liu, and N. Kato, "Networking and Communications in Autonomous Driving: A Survey," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 2, pp. 1243–1274, Dec. 2019.

[5] J. Zhang, F. Wang, K. Wang *et al.*, "Data-Driven Intelligent Transportation Systems: A Survey," *IEEE Trans. Intell. Transp. Syst.*, vol. 12, no. 4, pp. 1624–1639, Dec. 2011.

[6] I. Mavromatis, A. Tassi, R. J. Piechocki *et al.*, "Poster: Parallel Implementation of the OMNeT++ INET Framework for V2X Communications," in *Proc. of IEEE VNC 2018*, Dec. 2018, pp. 1–2.

[7] C. Steinmetz, A. Rettberg, F. G. C. Ribeiro *et al.*, "Internet of Things Ontology for Digital Twin in Cyber Physical Systems," in *Proc. of SBESC 2018*, Nov. 2018, pp. 154–159.

[8] D. Gunduz, P. de Kerret, N. D. Sidiropoulos *et al.*, "Machine Learning in the Air," *ArXiv preprint*, Apr. 2019.

[9] C. Sommer, R. German, and F. Dressler, "Bidirectionally Coupled Network and Road Traffic Simulation for Improved IVC Analysis," *IEEE Trans. Mobile Comput.*, vol. 10, no. 1, pp. 3–15, Jan. 2011.

[10] M. Rondinone, J. Maneros, D. Krajzewicz *et al.*, "iTETRIS: A modular simulation platform for the large scale evaluation of cooperative ITS applications," *Simul. Model. Pract. Th.*, vol. 34, pp. 99–125, 2013.

[11] M. K. Müller, F. Ademaj, T. Dittrich *et al.*, "Flexible multi-node simulation of cellular mobile communications: the Vienna 5G System Level Simulator," *EURASIP J. Wirel. Comm.*, vol. 2018, no. 1, pp. 17–26, Sep. 2018.

[12] P. A. Lopez, M. Behrisch, L. Bieker-Walz *et al.*, "Microscopic Traffic Simulation using SUMO," in *Proc. of ITSC 2018*. IEEE, Nov. 2018.

[13] M. Lord and D. Memmi, "NetSim: A Simulation and Visualization Software for Information Network Modeling," in *Proc. of International MCETECH 2008*. IEEE Computer Society, Jan. 2008, pp. 167–177.

[14] M. Piórkowski, M. Raya, A. L. Lugo *et al.*, "Trans: Realistic joint traffic and network simulator for vanets," *SIGMOBILE Mob. Comput. Commun. Rev.*, vol. 12, no. 1, pp. 31–33, Jan. 2008.

[15] B. Schünemann, "V2X simulation runtime infrastructure VSimRTI: An assessment tool to design smart traffic management systems," *Computer Networks*, vol. 55, no. 14, pp. 3189–3198, Oct. 2011.

[16] M. Boban, J. Barros, and O. K. Tonguz, "Geometry-Based Vehicle-to-Vehicle Channel Modeling for Large-Scale Simulation," *IEEE Trans. Veh. Technol.*, vol. 63, no. 9, pp. 4146–4164, Nov. 2014.

[17] I. Mavromatis, A. Tassi, R. J. Piechocki *et al.*, "Efficient Millimeter-Wave Infrastructure Placement for City-Scale ITS," in *Proc. of IEEE VTC-Spring 2019*, Apr. 2019, pp. 1–5.

[18] A. F. Acosta, J. E. Espinosa, and J. Espinosa, "TraCI4Matlab: Enabling the Integration of the SUMO Road Traffic Simulator and Matlab® Through a Software Re-engineering Process," *Modeling Mobility with Open Data*, pp. 155–170, Jun. 2015.

[19] F. Martínez, A. J. Rueda, and F. R. Feito, "A New Algorithm for Computing Boolean Operations on Polygons," *Computers & Geosciences*, vol. 35, no. 6, pp. 1177–1185, Aug. 2009.

[20] OpenStreetMap contributors, "Planet dump retrieved from https://planet.osm.org ," https://www.openstreetmap.org, 2017.

[21] I. A. Hemadeh, K. Satyanarayana, M. El-Hajjar *et al.*, "Millimeter-Wave Communications: Physical Channel Models, Design Considerations, Antenna Constructions, and Link-Budget," *IEEE Commun. Surveys Tuts.*, vol. 20, no. 2, pp. 870–913, Dec. 2018.

[22] D. Lee, S. Zhou, X. Zhong *et al.*, "Spatial Modeling of the Traffic Density in Cellular Networks," *IEEE Wirel. Commun.*, vol. 21, no. 1, pp. 80–88, Feb. 2014.