# Positioning and Perception in Cooperative ITS Application Simulator

Ye Tao, Manabu Tsukada, Hiroshi Esaki
Graduate School of Information Science and Technology, The University of Tokyo
Email: {tydus, tsukada}@hongo.wide.ad.jp, hiroshi@wide.ad.jp

***Abstract*—Intelligent Transportation System (ITS) is an intelligent system which can make the road traffic safer, more efficient, and more comfortable. The prerequisites of ITS are positioning, perception and networking. Different ITS infrastructures and applications are built by a large society from academia and industry. Nevertheless, the state-of-the-art network application simulators lack the capabilities to access the position and perception related information. In this paper, we proposed one extension to NS-3 network simulator and two to DCE application simulator, to bring the positioning and perception sensors to the simulated application. Both positioning and perception were designed fully experimental reproducible, and the simulated perception sensor can be adjusted to accommodate different types of devices. With the proposed work, much more scenarios of ITS applications can be tested, including Position Based Routing, Cooperative Awareness, etc.**

***Keywords–Intelligent Transportation System; Vehicular Ad-hoc Network; Network Simulator; Network Application Simulator; Perception Aware Simulator***

## I. INTRODUCTION

Intelligent Transportation Systems (ITS) aim at optimization of the road traffic by realizing safe, efficient and comfortable transportation. Cooperative ITS is a branch of ITS features sharing the information between nodes to realize their common objects. Application of Cooperative ITS includes driver assistance in the near future, however the vehicular communication also remains essential in autonomous driving in order to support wider perception of the other vehicles around a vehicle that cannot be detected by the sensors equipped in the vehicle.

In order to connect among vehicles and roadside units, GeoNetworking [1] is employed as one of the network protocols in the ITS station architecture [2], because the geolocation based routing features the strength in the network with dynamic topology compared with topology based routing. GeoNetworking employs position-based routing VANET (PBR-VANET) to adapt to high-speed movement requirements. PBR-VANET is a type of VANET routing protocol that uses the position information of nodes to direct routing. It does not maintain routing tables or exchange link state information. Such a routing protocol shows better performance in a highly dynamic topology, where link states change frequently. However, the vehicle movement scenarios in the simulations are "hand-crafted" and not realistic enough nor scalable enough. We need a network simulator which can provide GPS function, to make the experiments scalable and realistic.

Cooperative Awareness (CA) Service is a service defined in the ITS station architecture [3]. It enables ITS vehicles to report their position through the GeoNetworking, making the automated driving software on the other ITS station be aware of the vehicle. Nevertheless, in the deployment period of ITS, we could not assume every vehicle to be ready for the CA service. Thus, some transition techniques will be required.

Contributions to CA service can be evaluated in the field operational tests (FOT). However, the FOTs can take very expensive devices and human power, which could render the experiments unscalable. If we come to a realistic perception-aware simulator, experiments in large fields and large scale network will become possible.

From the examples above, we know that conducting real field experiments are costly and not scalable. This is why a position and perception enabled simulator is essential for evaluating the ITS researches.

In the different fields, the researchers have invented different sorts of simulators, including:

**Network Simulators** models and simulates different kinds of network with differnt layers, from wireless radio to application protocols.

**Program / Application Simulators** provides reproducible and scalable simulations of real programs for programmer to debug.

**Transportation Simulators** understands real maps. It could simulate vehicle behaviors on the roads simultaneously and output their positions for the other simulators.

Different kinds of the simulators are not aware of the other parts. Network simulators do not know the road and vehicle information which came from the transportation simulators; vehicles in the transportation simulators cannot reflect the feedback from ITS application; application simulators are not capable of realizing the feedback from the vehicle simulator. This issue may be caused by the separation of different research fields: perception simulators are developed by the robotic researchers; transportation simulators are developed by vehicle researchers, etc. On the other hand, ITS applications may have all the portions above involved. Thus, it requires connections among the different simulators, to make an integrated simulated environment of the ITS application.

For our small step towards the integrated simulated environment, we propose a framework of realizing positioning and perception functionalities with the network and application simulators in this paper. It will enable the simulated applications to get the position of the vehicles they are deployed into, which is essential for PBR-VANET. Also, the application simulator will be equipped with virtual perception sensors, which made it capable of simulating the Cooperative Awareness applications.

The remainder of the paper is organized as follows. Section II describes some related works includes *Intelligent Transportation Systems*, *network and application simulators* and an novel work of *obstacle radio propagation extension* for the Network Simulator 3 (NS-3). Section III analyzes the functionalities should be fulfilled by the VANET application

Copyright (c) IARIA, 2017. ISBN: 978-1-61208-573-9 

simulator and additional requirements to the improvements. Section IV gives an overview of the new framework with three parts, to provide the capabilities mentioned above. Section V and VI details the design and implementation of positioning and perception respectively. Section VII shows two use cases with the extensions, one per function. Finally, Section VIII summarizes our contributions and briefly explores directions for future work.

## II. Related work

This section presents various related work in two parts: Intelligent Transportation System and Network Simulators.

### A. Intelligent Transportation System

The road network is inter-connected among countries and there are few barriers and using the network, the vehicles easily cross country border. For the interoperability among the countries, cooperative ITS needs to be developed based on the same architecture, protocols and technologies. Europe has huge necessity of standardization of Cooperative ITS and European Commission (EC) published the action plan [4] followed by ITS standardization mandate [5], to promote the deployment of these systems in Europe. In US, the Institute of Electrical and Electronics Engineers (IEEE) is standardizing Wireless Access in Vehicular Environments (WAVE) architecture in IEEE 1609 family of standards [6] as well as IEEE802.11 variant for vehicular communication as IEEE802.11p [7]. Cooperative ITS and vehicular communications became essential for the cooperation of multiple entities in the road traffic (*i.e.*, vehicles, roadside infrastructure, traffic control centers) in order to achieve shared objectives (safety, efficiency, and comfort).

GeoNetworking [1] has been standardized by ETSI as a network layer protocol. It integrates several Position-Based Routing (PBR) strategies, including Greedy Forwarding (GF) [8] (also known as *GPSR*), which chooses a directly reachable node that is closest to the destination according to the GPS location obtained by the *Location Service(LS)* request action, to route packets more effectively in vehicular networks.

In the literature, the evaluation of GeoNetworking can be performed in flexible and large-scale simulated the network with low cost. However, mere simulations cannot provide realistic evaluation results for a specific implementation of GeoNetworking. In contrast, the experimental evaluation using the implementation in a field operational testbed gives real results in the deployment phase of GeoNetworking. Though in practice, it requires a heavy cost to conduct the experiments regarding time, manpower, space, and expense. To take the benefits of real field test and simulation, a realistic network simulator with geo-positioning features is necessary.

In Vehicle-to-Vehicle (V2V) architectures, the "I-am-here" messages are widely employed to notify the surrounding about the position of the vehicles. In the ITS station architecture, this type of message is named as *Cooperative Awareness Message (CAM)* [3].

CAMs are messages exchanged in the ITS network between stations to create and maintain awareness of each other and to support the cooperative performance of vehicles using the road network. A CAM contains status and attributes information of the originating ITS-S, e.g., time, position, motion state, activated systems. On reception of a CAM, the receiving ITS-S becomes aware of the presence, type, and status of the originating ITS-S.

Due to realistic reasons, such as market penetration ratio, CAM could not be required to be deployed on every vehicle. This may cause a serious problem because the ITS stations cannot be aware of vehicles do not equip with ITS infrastructures. To solve this problem, [9] have proposed an infrastructure-bases Proxy CAM under some environments. It employs roadside units (RSUs) to detect non-ITS objects and generate and broadcast CAMs in the behavior of them.

In the proposal, RSUs are assumed to be equipped with sensors in charge of object detection. On the contrary, in the experiment setup, there is only one stereo camera, because of the limitation of cost and manpower. In other words, the work needs to be evaluated in large scenarios. The only promising way to achieve this is the simulation.

### B. NS-3 Direct Code Execution

NS-3 is a discrete-event network simulator, which is widely used in networking researches. NS-3 Direct Code Execution (NS3-DCE)[10] is an application simulator based on NS3. It mainly features reproducible experiments and easy debugging.

In the paper, the authors emphasized experiments reproducibility which was defined by [11]: experimentation realism, topology flexibility, easy and low-cost replication.

NS3-DCE takes a Library Operating System (LibOS) approach to making a slightly modified Linux kernel network stack running in the simulated environment. It implements a standard-compatible POSIX layer for user applications to be built onto, with no or minor modification. It makes developing and testing new protocols easier and more predictable. Also, it achieved a very good performance compared to the other works such as Mininet.

NS3-DCE have a modified Linux kernel layer and its own implementation of POSIX layer. Subsequential researchers are made easy to extend it to realize new functions or improve the performance.

### C. Obstacle and Radio Fading Model in NS-3

Carpenter et al. [12] proposed an obstacle model implementation for the NS-3, in order to simulate radio shadowing in the NS-3 environment.

The radio shadowing model is separated into three layers. First, it uses a *polygon* to hold the outline of an obstacle (e.g., outer walls of a building). Then, it defines the *Topology* to handle the set of the *polygons*, which including the handling the data imported from OpenStreetMap. Finally, it uses the Computational Geometric Algorithms Library (CGAL) to calculate the radio shadowing according to the *Topology*.

In the implementation, it extended the *ns-3* in two parts: the `core` part and the `radio propagation` part. Three main classes are defined as follows:

**`Obstacle`** implements the `polygon` related data structures. This class is an extension of the `core` layer.

**`Topology`** operates on the set of `Obstacle`s, which includs handling of the data imported from the OpenStreetMap. This class is also an extension of the `core` layer.

**`ObstacleShadowingPropagationLossModel`** extends the standard interface

`PropagationLossModel`, which calculate the path propagation loss by the obstacle information. It could be plugged into any user program with the `PropagationLossModel` compatible interface.

The paper presented a concrete and promising implementation of the radio shadowing model. However, work only provides the model for radio shadowing. Thus, we need a different model for perception calculation. Fortunately, we may re-use the infrastructures including polygons and topology to get rid of re-inventing the wheel.

## III. Problem Statement and Requirements

In the last section, we have detailed the missing functionalities for network and application simulators and shown some related works about obstacles. In this section, we will analyze the issues we've found in the last section and detail the desired functionalities of the system, then define some additional requirements to our proposal.

### A. Position-awareness in application

Obviously, every ITS applications requires knowing the position of the vehicle, either directly or indirectly.

In order to bring the positioning to the applications, we need to fill the gap between Network simulator and ITS applications, by extending the application simulator. Two additional requirements should be fulfilled.

**Keep standardized protocol**
Standardized protocols are important because user applications usually builds according to the standard. If we can stick with the standard protocols, user applications can be simulated directly without any changes. This benefits our user (researcher) on both programming simplicity and does not require extra testing / debugging regarding the protocol changes.
In our case, Global Navigation Satellite System (GNSS) could be selected. Among the GNSSes, the most widely known and used system is the Global Positioning System (GPS). Most of the GPS devices follows the "NMEA 0183" standard. NMEA 0183 is an ASCII character and sentence based protocol running on the serial RS-232 port. It also allows simultaneous listens to a same device, a.k.a broadcasting. These features made it relatively easy to be implemented.

**Experimental reproducibility**
As we described above, experimental reproducibility is essential to application simulator. Experimental reproducibility requires the simulation to the positioning device fully determinative. The output of the device should be completely same between different simulations (and unrelated simulation configurations). This requirement may bring extra challenges to the design of the simulated device.

### B. Object-detection functionality in NS-3 and application

In robotics, perception (or machine perception) means the ability a computer receives the real world data and interpret them similar humans beings. In the ITS, perception is also a fundamental requirement, since the autonomous vehicles should always aware of what happens the around them. In the most basic period, we narrow the term "machine perception" into "machine vision", which itself is based on object-detection functionality.

Nevertheless, the object-detection function is missing on both NS-3 and DCE. Thus, we need to implement it from scratch in NS-3, then provide a simulated sensor interface to the user program through DCE. Two requirements are defined in the design, as well as the most basic one: experimental reproducibility.

**Scalable to large-scale scenario**
Performance and scalability should be an important consideration. If we come to a factorial complexity ($O(n!)$) approach, even fastest computers cannot make the simulator scale into hundreds of vehicles. In order to conduct large-scale experiments, we should optimize the computational complexity by workaround the time-consuming part, and make sure it is well scalable.

**Able to accommodate different types of sensors**
In the market, various types of perception sensors based on different technologies are developed, including vision-based, lazer-based, IR-based, doppler-based, etc. Different types of sensors greatly differ on functionalities and performances. The simulator should be able to accommodate these types of sensors, to fulfill the requirements of experimented applications.

## IV. Overview of Position and Perception Simulation

In this section, we will discuss how to realize the position and perception in NS3-DCE and give out our proposals.

Hereby, we define *full-simulation* and *para-simulation* which are similar to the widely-used term *full-virtualization* and *para-virtualization* [13]:

**Full simulation**
Fully reflect the behavior of the simulatee, which means the interface to the device and the protocol are completely identical as the real device. In our case, a full-simulated device does not require any modification to the user program.

**Para-simulation**
A technique to provide an interface to the device, but not identical to the real underlying hardware. This approach may boost the performance by bypassing unnecessary work from the protocol and data representation. In our case, a para-simulated device requires a user-mode daemon or plugin to the user program to cooperate with.

Figure 1 is a simplified module graph of the NS-3 and DCE, with the extended modules. Three new modules are required: **GPS** and **Perception Sensors** in the DCE, and **Object Detection** in the NS-3.

## V. Design and Implementation of Positioning

In the last section, we proposed a architecture considering the extension to NS-3 and DCE. This section details the design and implementation of the positioning module.

### A. Position augmentation, stablize and coordinate conversion

In order to generate GPS signal, we need to convert the cartesian coordinate in the NS3 to the GPS coordinate (i.e., latitude and longitude). Several parameters are defined to make the simulation scalable to different GPS devices.

Copyright (c) IARIA, 2017. ISBN: 978-1-61208-573-9

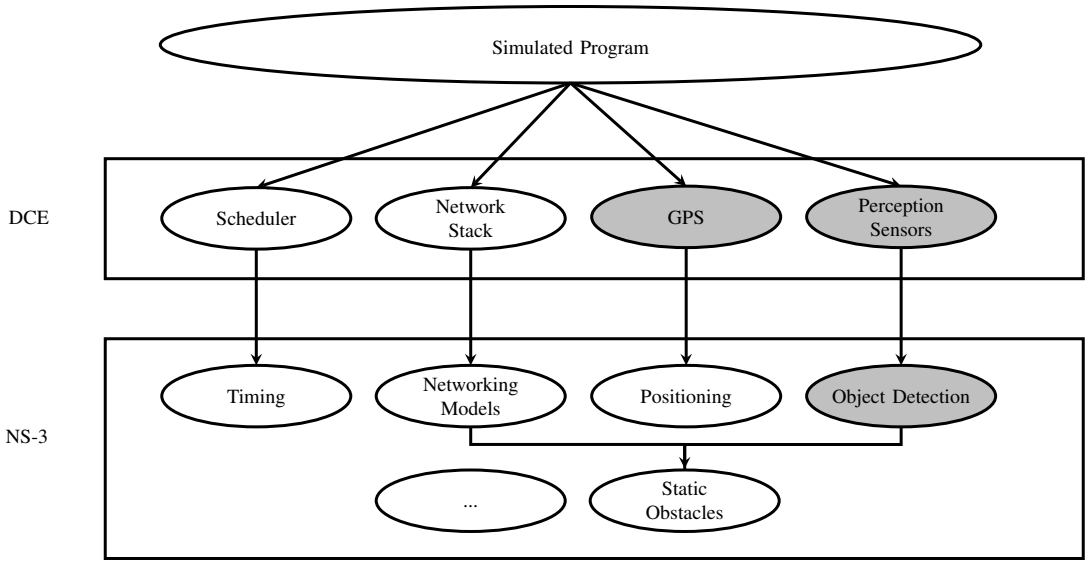


Figure 1. NS3-DCE architecture and extended modules

**center** The center coordinate of the experiment, in (latitude, longitude).
**accuracy** The accuracy of the simulated GPS device, in m.

To make the node position more realistic, we did some data augmentation to add randomness to it. For the experimental reproducibility, the randomization approach should be fully deterministic. NS-3 provides deterministic Random Number Generators (RNGs) including different distributions. In the case, we use a `NormalRandomVariable` with $\sigma = accuracy \div 3$ for the radius and a `UniformRandomVariable` with range $[-\pi, \pi)$ for the angle.

Then we did a sensor fusion to stabilize the randomized position using the Kalman Filter Algorithm. The internal variable of the filter is stored with the state of RNGs in the `DceNodeContext`, which is used for keeping the per-node information, in order to keep the experimental reproducibility among different experimental setups. Finally, the NS-3 coordinate is converted into GPS coordinate.

### B. NMEA full-simulation

We take the fully-simulation approach because of the requirement of *keeping on standard format*, i.e., the NMEA format.

The data flow is shown in Figure 2. A unix character device `/dev/ttyGPS` is exposed to the user program. The device driver is defined in the POSIX layer and named `UnixGPSttyFd`. When the device is read by a program, `DCENodeContext::GPSttyRead()` is called. Then the function call another function `DCENodeContext::GetGPSPosition()` to get the "GPS position" from the mobility model in NS-3.

After the `DCENodeContext::GetGPSPosition()` got the node's real position according to the mobility model, it calculates the GPS position as defined in Section V-A. Then the function returns to the `DCENodeContext::GPSttyRead()`. Here, the function

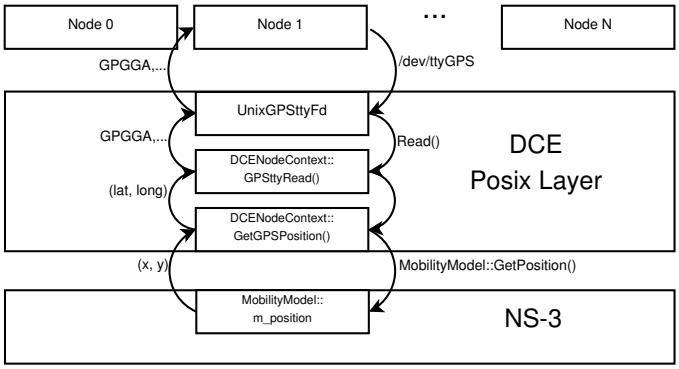


Figure 2. GPS device data flow

generate the necessary GPS flavor NMEA clauses (specifically `GPGGA` and `GPRMC`) and finally return to the caller program.

## VI. Design and Implementation of Perception

This section describes an on-going work towards the perception functionality of NS-3 and DCE. According to the architecture defined in Figure 1, vehicle perception sensor simulation on the *DCE* is based on the object detection function on *NS-3*.

### A. Architecture overview

In NS-3, polygons and some of its operations are supported through the `CGAL` by the obstacle extension module contributed by [12]. Additionally, the module comes with an approach to import obstacle information from the OpenStreetMap (OSM). These functions can be reused as a common infrastructure in NS-3 to represent and handle obstacles, with several new functions.

On the other side, the vehicles themselves are also polygons. We can define the outline of each type of vehicles as 2D polygons, and then they can be awared in the calculation. Other types of objects (e.g., non-its vehicles, non-motor vehicles,

Copyright (c) IARIA, 2017. ISBN: 978-1-61208-573-9 

pedestrians) can be defined as "stub nodes", i.e., nodes do not run any protocol stacks or programs on it.

Based on the static obstacle and vehicle outline representation, the object detection simulation could be realized. When an object-detection is requested on a sensor, it counts all the vehicles (including stub ones) which are within or intersects with its sensing range. Then a line-of-sight check is performed to check each vehicle in the range is visible, or fully blocked by either static obstacles or another vehicle at the moment. Finally, for each visible vehicle, an `ID` is assigned to it, for both NS-3 representation and simulated sensor representation.

### B. Modeling the sensors

Currently, different types of perception sensors are available on the market. According to [14], vehicle perception sensors can be classified into different types, including vision-based, lazer-based, IR-based, radar, doppler, ultrasonic, induction loop, magnet field, etc. To accommodate different types of perception sensors, several sensor parameters are defined:

**position** Coordinate of the sensors, in (x, y). Follows the position of the node, if not specified.
**range** Shape of the sensible region of the device, in one of the Circle_2, Circular_arc_2, Polygon_2.
**min_size** Minimum object size (in solid angle) which could be detected.
**accuracy** Accuracy of the simulated sensor, in m.
**capability** Mask of capabilities the sensor has (identify vehicles, classify vehicles, or just count)

For instance, several sensor products could be modeled with parameters in Table I.

### C. The Para-simulation

Different from the position part, we decided to follow the para-simulation approach for the perception. The reasons are as follows:

Firstly, the vehicle perception sensors do not have a widely-used standard protocol: different products require user applications to communicate in different protocols, and also they need different routines to achieve the object-detection from the device data. In this case, extra developing will always be required for a new type of device, including our simulated ones.

Secondly, object-detection from the images or point-clouds are very time-consuming, as well as image generation, if we use a full-simulation approach. To simplify and speed-up the simulation, we bypass the *Object* $\rightarrow$ *Image* $\rightarrow$ *Object Detection* routine and directly offer the result of object detection to the user program. We model different types of the perception sensors and define several arguments to imitate the behavior and performance of each sensor.

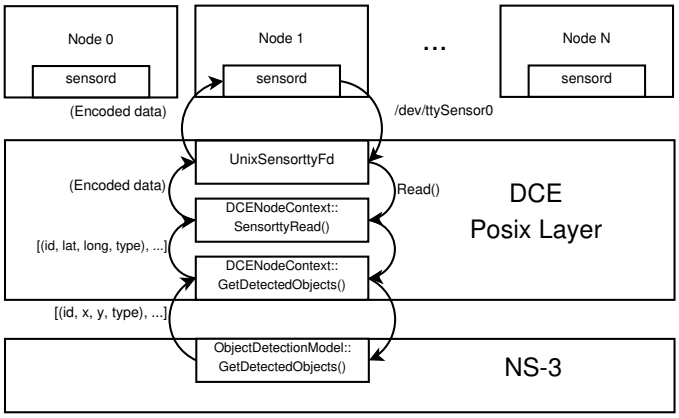


Figure 3. Perception sensor device data flow

The data flow is shown in Figure 3. Unlike the GPS approach described above, we inject a program `sensord` to the user environment, which is in charge of communication and object-detection. `Sensord` comes with an SDK with libraries to access the data, like other real sensor hardware does.

The `sensord` program communicates with the POSIX layer with the character device `/dev/ttySensor0`. The posix layer processes in a similar way as the GPS device, and finally the request comes to the NS-3 Object detection model.

The model returned the raw data which is presented as a list of *id*, *x*, *y*, *type*, and is subject to extend. Then the `DCENodeContext::GetDetectedObjects()` function converts the *(x,y)* coordinate into the GPS coordinate, with the help of the utility from GPS model. The function `DCENodeContext::SensorttyRead()` encode the data in json (to keep flexibility to extensions) and return to the user space daemon `sensord`. `Sensord` decodes the data and return to its subscriber (i.e., the user programs in the same node).

## VII. USE CASES

Here, we show some example that our extensions could be employed. Both positioning and perception sensing are most common tasks in ITS. Our work will be useful for various kinds of scenarios and not limited to the following ones.

### A. Position Based Routing

The Position-based routing algorithms are not perfect enough and needs improvements on different issues. GeoNetworking is not an exception. We presented a proposal named "Duplicated Unicast Packet Encapsulation" [15] to improve the reliability of the GeoNetworking protocol for important multi-hop messages by employing overlay networking and

TABLE I. PARAMETERS FOR DIFFERENT TYPES OF SENSORS

| Sensor Type | ZMP RoboVision II | Velodyne LiDAR | Diamond Phoenix II |
|---|---|---|---|
| Type | Stereo camera | Laser Radar | Induction Loop |
| Position | In-vehicle / Fixed | In-vehicle / Fixed | Fixed |
| Range | Sector, 80m | Circle, 100m | Circle / Rectangle, 1m |
| Min Size | 50cm, 10cm | ... | 10cm, 10cm |
| Accuracy | 30cm | ... | ... |
| Capability | Identify | Identify | Classify |

Copyright (c) IARIA, 2017. ISBN: 978-1-61208-573-9 

multipath routing. It presents an implementation based on an open-source ITS network stack, and several (real and simulated) experiments were conducted on the "Combined Realistic Evaluation Workflow" [16]. However, the vehicle movement scenarios in the simulations are "hand-crafted" and not realistic enough nor scalable enough. With the proposed work, we can conduct experiments with real street maps, to make the experiments scalable and realistic.

### B. Cooperative Awareness Messages

In order to keep compatibilities with the plain old manual-driving vehicles, some transition techniques of Cooperative Awareness Service is required. Kitazato et al. [9] proposed a preliminary work call "Proxy CAM". By using perception techniques provided by sensors, road-side units (RSUs) could aware of the non-CA-ready vehicles and send CA messages (CAMs) on their behalf. In the paper, they conducted several simple and small-scale experiments using three to four nodes. Apparently, those are far not enough to evaluate the proposal in a large scenario.

A realistic simulator with sensors enabled could greatly help the evaluation. We can build flexible scenarios based on real maps of different cities, and deploy various types of position sensors on various type of objects, e.g., LiDARs and stereo visions on the ITS enabled vehicles, dopplers on traffic lights, induction detection loops under the road.

## VIII. Conclusion

In this paper, we proposed methods to realize positioning and perception devices in the application simulator NS3-DCE. Both work we proposed satisfy the "experimental reproducibility" request, i.e., producing deterministic result among different simulation runs. Additionally, we attempted to model different types of perception sensor techniques and bringing adjustable parameters for flexibility to the users of the simulator.

The design and implementation of perception are not finished yet. In the near future, we will finish the following targets:

**Update Location Module**
: We have built a prototype location module with basic function (i.e., NMEA simulation). The data augmentation is not implemented yet. We will rewrite the location module with the data augmentation and other features.

**Implement 2D Perception**
: Currently, 2D perception is in the design period. In the future, design and implementation challenges may occur. The challenges should be overcome or workaround.

**Evaluation**
: After the location module is updated and 2D perception module is implemented, we are able to conduct experiments on it. With the help of the modules, large-scale experiments using the real world map become possible. We will evaluate using different applications, maps of different cities, and different types of the perception sensors.

## Acknowledgment

We would like to thank Xin Li of our lab, for his groundbreaking ideas of the prototype GPS mocking implementation, e.g., intercepting device handler for the GPS device. We also thank Masahiro Kitazawa, for his knowledges and ideas on the characteristics of various types of perception sensors.

This work was supported by JSPS KAKENHI Grant Number JP17H04678 and JP26730045.

## References

[1] Intelligent Transport Systems (ITS); Vehicular Communications; GeoNetworking; Part 4; Sub-part 1, Std., Jul. 2014. [Online]. Available: http://www.etsi.org/deliver/etsi_en/302600_302699/3026360401/01.02.01_60/en_3026360401v010201p.pdf

[2] ISO 21217:2010 Intelligent transport systems – Communications access for land mobiles (CALM) – Architecture, ISO CALM TC204 Std., April 2010.

[3] Intelligent Transport Systems (ITS); Vehicular Communications; Basic Set of Applications; Part 2, Std., Sep. 2014. [Online]. Available: http://www.etsi.org/deliver/etsi_en/302600_302699/30263702/01.03.01_30/en_30263702v010301v.pdf

[4] Action plan for the deployment of Intelligent Transport Systems in Europe, European Commission Std., December 2008, cOM(2008) 886 final.

[5] Standardisation mandate addressed to CEN, CENELEC and ETSI in the field of information and communication technologies to support the interoperability of co-operative systems for intelligent transport in the european community, EUROPEAN COMMISSION Std., October 2009.

[6] IEEE 1609.0 Draft Standard for Wireless Access in Vehicular Environments (WAVE) - Architecture, IEEE Std., April 2010.

[7] IEEE Standard for Information technology - Telecommunications and information exchange between systems - Local and metropolitan area networks - Specific requirement, Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications, IEEE Computer Society Std., July 2010, iEEE Std 802.11p-2010.

[8] B. Karp and H. T. Kung, "Gpsr: Greedy perimeter stateless routing for wireless networks," in 6th Annual International Conference on Mobile Computing and Networking, MobiCom 2000, August 6.-11., 2000, Boston, Massachusetts, USA. ACM / IEEE, August 2000, pp. 243–254.

[9] T. Kitazato, M. Tsukada, H. Ochiai, and H. Esaki, "Proxy cooperative awareness message: an infrastructure-assisted v2v messaging," in Mobile Computing and Ubiquitous Networking (ICMU), 2016 Ninth International Conference on. IEEE, 2016, pp. 1–6.

[10] H. Tazaki, F. Uarbani, E. Mancini, M. Lacage, D. Camara, T. Turletti, and W. Dabbous, "Direct code execution: Revisiting library os architecture for reproducible network experiments," in Proceedings of the ninth ACM conference on Emerging networking experiments and technologies. ACM, 2013, pp. 217–228.

[11] N. Handigol, B. Heller, V. Jeyakumar, B. Lantz, and N. McKeown, "Reproducible network experiments using container-based emulation," in Proceedings of the 8th international conference on Emerging networking experiments and technologies. ACM, 2012, pp. 253–264.

[12] S. E. Carpenter and M. L. Sichitiu, "An obstacle model implementation for evaluating radio shadowing with ns-3," in Proceedings of the 2015 Workshop on ns-3. ACM, 2015, pp. 17–24.

[13] A. Whitaker, M. Shaw, S. D. Gribble et al., "Denali: Lightweight virtual machines for distributed and networked applications," Technical Report 02-02-01, University of Washington, Tech. Rep., 2002.

[14] T. V. Mathew, "Automated traffic measurement," Trac Engineering And Management, IIT Bombay April, 2012. [Online]. Available: https://www.civil.iitb.ac.in/tvm/1111_nptel/524_AutoMer/plain.pdf

[15] Y. Tao, X. Li, M. Tsukada, and H. Esaki, "Dupe: Duplicated unicast packet encapsulation in position-based routing vanet," in 2016 9th IFIP Wireless and Mobile Networking Conference (WMNC), July 2016, pp. 123–130.

[16] Y. Tao, M. Tsukada, X. Li, M. Kakiuchi, and H. Esaki, "Reproducing and extending real testbed evaluation of geonetworking implementation in simulated networks," in The 10th International Conference on Future Internet. ACM, 2015, pp. 27–34.

Copyright (c) IARIA, 2017. ISBN: 978-1-61208-573-9 