NS3-VANETZA INTEGRATION

There are multiple simulators available for Vehicle Ad-hoc network simulations. Vanet simulators are combination of network and traffic simulators. SUMO is popular traffic simulator for vehicle ad-hoc network but for network simulator multiple simulators are available. Some of them are open source and others are commercial simulators. NS3 is open source network simulator used for wireless network simulations.

**NS3 simulator**:

Ns3 is discrete event simulator for internet systems. Below diagram shows the basic model of simulator

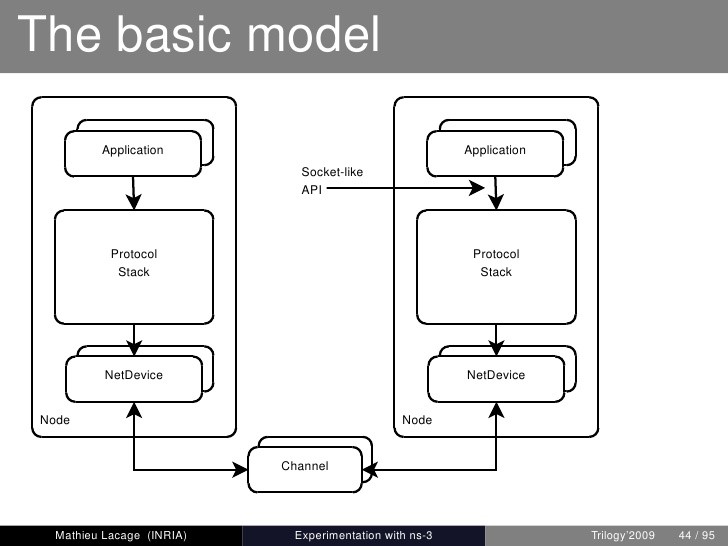


FIG 3: NS3 basic model.

The basic model of ns3 has four main components.

1. Node
2. Application
3. Net device
4. Channel

**Node**:

In ns-3, the basic computing device abstraction is called the node. This abstraction is represented in C++ by the class Node. The Node class provides methods for managing the

Representations of computing devices in simulations. We can think node is a computer where you can add functionality, like application, protocol stack.

**Application**:

Just as software applications run on computers to perform tasks in the “real world,” ns3 applications runs on ns-3 Nodes to drive simulations in the simulated world.

**Channel**:

The media over which data flows in these networks are called channels. In the simulated world of ns-3, one connects a Node to an object representing a communication channel. Here the basic communication subnetwork abstraction is called the channel and is represented in C++ by the class Channel. The Channel class provides methods for managing communication subnetwork objects and connecting nodes to them.

**Net device**:

In ns-3, the net device abstraction covers both the software driver and the simulated hardware. A net device is “installed” in a Node in order to enable the Node to communicate with other Nodes in the simulation via Channels. Just as in a real computer, a Node may be connected to more than one Channel via multiple Net Devices. The net device abstraction is represented in C++ by the class Net Device.

**Protocol Stack:**

In Ns-3 simulator protocol stack is implementation of various network protocols. Ns3 provides flexibility to implement and use various protocol stacks of network. Protocol stack exists as separate container that can be installed inside the node. Application and protocol stack communicate using ns3 socket like API.

**Topology:**

Network topology is arrangement with which computer systems and network devices are connected with each other. Since connecting Net Devices to Nodes, Net Devices to Channels, assigning IP addresses, etc., are such common tasks in ns-3; ns3 provides topology helpers to make process easy.

**Vanetza**: There exist two protocol standards for vehicle ad-hoc network they are IEEE standards and ETSI standards. For our project ETSI standards are used. ITS-G5 components, which are mostly independent of the intended target platform, are in the focus of Vanetza. This comprises the network and transport layers as well as the congestion control and security cross-layers, which are colored green in below Figure. Vanetza is an open library and written in c++ language. Since it is written in c++, it can be built for almost any system where a standard-compliant C++ compiler and the dependencies (Boost, GeographicLib, Crypto++ and optionally OpenSSL) are available.

Figure 1 shows the ETSI standards layer including the ISO layers.

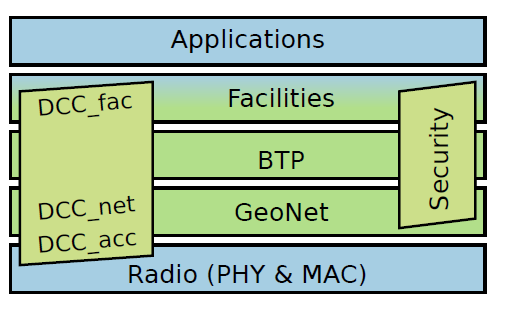


Fig 2: layers supported by vanetza (green) in ETSI-ITS g5 architecture

There are several components in ITS-G5 depending on time progress, e.g. expiry of location table entries, packet routing or repetitions. Vanetza, however, is a purely reactive system on purpose, i.e. no Vanetza code runs without stimuli from outside. Those stimuli can be incoming and outgoing packets but also time triggers. Such time triggers are supported by Vanetza through its Runtime instance, which can be used by other components for scheduling timed callbacks. Progress of Runtime in turn is controlled from outside, e.g. a system clock read by the enclosing program.

At initial stage of the project, veins simulator is considered for simulations. it is a combination of omnetpp network simulator and SUMO traffic simulator. However, it is found that for corporate use, omnetpp is commercial simulator and cost of procurement is very high. We searched for an alternate network simulator and evaluated multiple simulators and found Ns3 simulator, which is an open source simulator also for corporate firms. As discussed previously ETSI protocol standards were used in project but Ns3 only implemented two layers of IEEE protocols physical, mac layers which are almost same for both standards. Implementing ETSI protocols stack in Ns3 is very difficult and it will take more time. We found **ezcar2x** framework, which is developed by **Fraunhofer** **institute.** It developed on top of Ns3 network simulator and uses SUMO as traffic simulator. It implemented ETSI protocol stack. However, acquiring ezcar2x framework takes some time and it depends on ns3 simulator. Vanetza an open library implemented rest of ETSI protocols layers. Therefore, we thought integrating vanetza with ns3 would complete the ETSI stack in ns3 and enable us to simulate vehicle-to-vehicle simulations. It also helps to understand Ns3 simulator function and ETSI protocols.

**METHOD**:

Our main objective is to integrate the Ns3 simulator and vanetza library to create ETSI protocol stack and simulate the vehicle-to-vehicle network.

Below diagram shows workflow.

Making vanetza shared library

Exploring existing VANET standards in ns3

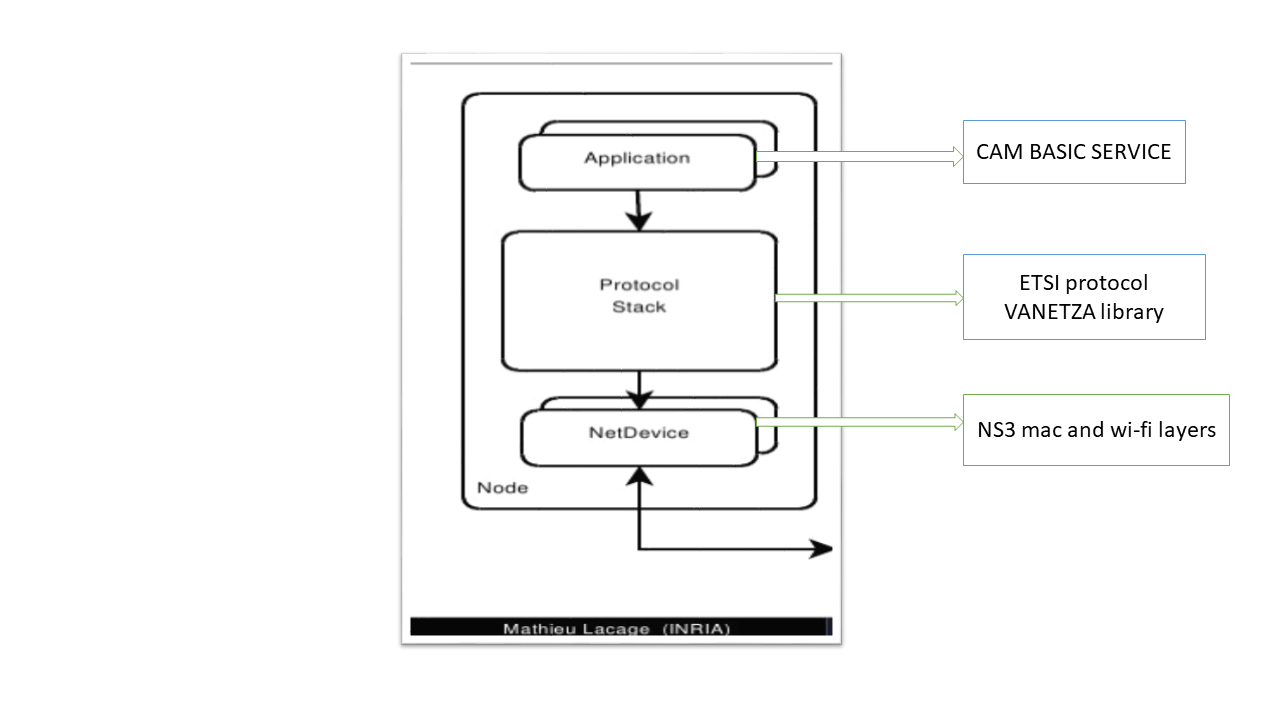
Implementation of Cam basic service in ns3 application

Implementing ETSI protocol stack in ns3

Simulation of vehicle-to-vehicle scenario

FIG: Workflow

Therefore, the idea is to with the help of vanetza complete the ETSI protocol stack and run a basic vanet application (CAM application) to test it in vehicle-to-vehicle communication.



**Making vanetza a sharing library:**

**Exploring existed VANET standards in NS3:**

**Implementation and execution of Ca-basic service application in ns3:**

**Approach:**

**Cam service** **Application:** Cam is short form for “**co-operative awareness message**”. The main purpose of this service is broadcasting the vehicle status to neighboring vehicles for every 10 seconds example speed, heading, acceleration etc.

ETSI standard **EN 302-637** defines the Cam-Service application, conditions for broad casting and encoding method. Cam message uses ASN.1 encoding.

However, we found out that artery has already implemented CA service application but problem is it is strongly dependent on omnetpp simulator, so we cannot use that. Therefore, we decided to implement our own application in ns3 based on artery application.

**Work flow:**

1. Reading EN 302-637 protocol to understand CA basic service.
2. Replicating CA service application in ns3 same as in artery.
3. Running simulation of vehicle-to-vehicle communication with Caservice application using ISO network protocols.

**Work done**:

1. Reading cam basic service application protocol in ETSI standard. (ETSI EN 302-637).
2. Reading protocols of ETSI layers. Focus on geonet, btp part.
3. Understanding artery code of cam basic service.
4. By example of CA Service in artery code defining the functions of CA-service in ns3.
5. Partially implemented some of the functions in CA service in ns3.

Challenges at each stage:

Artery code:

Artery module documentation is not available so understanding flow of CA-service application became hard. Solved problem by extensively studying the protocols existed and comparing it with artery code and digging deep into artery source code.

Replication of CA service in ns3:

1. How to write an application in ns3 is a major problem, as we have no knowledge on ns3 simulator. Therefore, to understand the ns3 simulator we went through tutorials and already given examples.
2. We are not considering traffic simulator at this stage but we need movement of the nodes in simulation for this we took help of mobility model in ns3.
3. After creating the cam message, it is an object but we need to convert it into a buffer using ns3 buffer class. However, understanding buffer class in ns3 is taking some time because it is complicated structure.

**Implementation of ETSI protocol in ns3 using vanetza**

**Simulation of vehicle-vehicle communication scenario in ns3**

**Future work:**

Reason for choosing ETSI standards above IEEE is

1. IEEE standards are not openly available. You needed a membership to get access to the standards. When we started, we did not have access to them and so we figured that we would focus on ETSI because they were freely accessible.
2. Even after we received the IEEE standards access, there was heavy dependency on another set of standards developed by SAE (Society of Automobile Engineers) which is also behind paywall. Therefore, it was almost impossible to get access to all these standards.
3. Due to poor access to standards, there was no easy way to verify the completeness of IEEE standards implementations by various frameworks/libraries. This may also be a reason why very little literature is available to assess or compare the quality of IEEE implementation.
4. On the other hand, ETSI standards were completely open and accessible, there were implementations of these standards that were better documented, and literature was there that compared these implementations and highlighted drawbacks.
5. In omnetpp simulator veins is developed in IEEE standards but one of the key components for our project “security” is missing. Artery that implemented security part in omnetpp is developed using ETSI standards.
6. For ns3 and vanetza integration, ns3 did not implemented vanet standards at all. Therefore, we need to implement total protocol stack starting from zero which is time consuming. However, vanetza an open library that can be integrated with the ns3 simulator is providing ETSI protocol stack.

**SIMULATORS:**

Vanet technology is still in development and to carry out real world experiments it requires large infrastructure and space. Cost is very high to set the infrastructure. However, there are multiple simulators were available to create and test the vanet network. For selecting a simulator for our project, multiple simulators were explored and pros and cons of every simulator were studied.

Let us understand what vehicle-to-vehicle simulators are. Vehicle-to-vehicle simulators are combination of two independent simulators.

1. Network simulator.
2. Traffic simulator.

**Network simulator**: network simulation is a technique whereby a software program models the behavior of a network by calculating the interaction between the different network entities. For example ns1, ns2, omnetpp etc.

**Traffic simulator**: Traffic simulation or the simulation of transportation systems is the mathematical modeling of transportation systems (e.g., freeway junctions, arterial routes, roundabouts, downtown grid systems, etc.) through the application of computer software to better help plan, design, and operate transportation systems.

In vehicle-to-vehicle simulator, both traffic and network simulators are integrated to simulate the scenarios. A two-way communication between both simulators is required as strong coupling.

For deciding the simulator, below features are considered as priority.

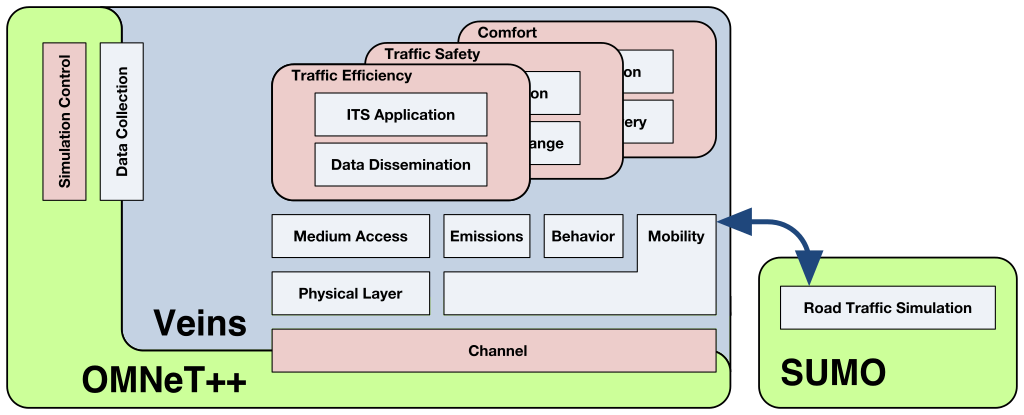
1. Protocols of the vehicle-to-vehicle network should completely implemented either IEEE 1609.x or ETSI-ITS G5.
2. Security standard (IEEE 1609.2) proposed for the vehicle ad-hoc network (VANET) should implemented including PKI architecture.
3. Documentation of the simulator should be clear to understand simulator.
4. Strong integration between traffic and network simulator.

Below table points the various simulators were explored and the features of them

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Name** | **Open source** | **Focus** (standards) | **Missing standard features** | **Traffic simulator** | **Network simulator** |
| Veins | Yes | WAVE | Security, data encoding, time position coupling | SUMO | OMNETPP |
| ARTERY | Yes | ETSI-ITS | Outdated security, time-position coupling , its-g5 , DCC | SUMO | OMNETPP |
| iTETRIS | Yes | ETSI-ITS | Security, ASN.1 , time position coupling | SUMO | Ns-3 |
| VSIMRTI | No | ETSI-ITS | DCC | SUMO, ViSSim | Ns-3 , OMNETPP |
| Ezcar2x | No | ETSI-ITS | - | SUMO | Ns3 |

Table 1: COMPARISON OF FEATURES OF DIFFERENT SIMULATION FRAMEWORKS FOR VANETS

**Veins simulator**: Veins (vehicle in network simulator) is open source simulation framework, suitable for vehicle-to-vehicle simulations. Veins is a framework built top on **Omnetpp** network simulator and it uses **SUMO** as traffic simulator. Other components takes care of setting up, running and tracking simulators. It is framework because it is meant to serve as basis for writing application specific code for simulations.  Typically, this user written code will be an application that is to be evaluated by means of a simulation. The framework takes care of the rest: modeling lower protocol layers and node mobility, taking care of setting up the simulation, ensuring its proper execution, and collecting results during and after the simulation.



VEINS FRAMEWORK

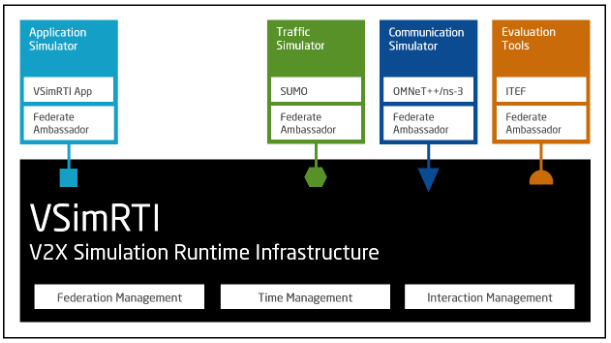
At every instance of simulation in veins, it is combination of two other simulations network simulator (omnetpp) and traffic simulator (SUMO). Both are connected bi-directional by a well-defined standard called traffic control Interface (TraCi).

IEEE 1609. X except 1609.2(security) implemented in this simulator. Documentation is available and for security, it is taking help of external modules built top of it ex. PREXT. However, omnetpp is not an open source network simulator for corporate and the license cost is very high so it is not ideal for project.

**Artery**: Artery is an open source module built top on **omnetpp** to simulate ETSI standards instead of IEEE standards. It is taking help of open library **vanetza** for implementation of ETSI standards. However, the module documentation is not available and it only support omnetpp, which is not open source simulator.

**ITETRIS**: this project is not actively maintained and very outdated.

**VSimRti**: The V2X Simulation Runtime Infrastructure (VSimRTI) is a comprehensive framework for the assessment of new solutions for Cooperative Intelligent Transportation Systems. Vehicle movements and sophisticated communication technologies like Vehicle-2-X communication and cellular networks can be modeled in detail. VSimRTI couples different simulators to allow the simulation of the various aspects of future vehicle-to-vehicle network Systems. The easy integration and exchange of simulators enables the substitution of the most relevant simulators for a realistic presentation of vehicular traffic, emissions, wireless communication (cellular and ad-hoc), user behavior, and the modelling of mobility applications.



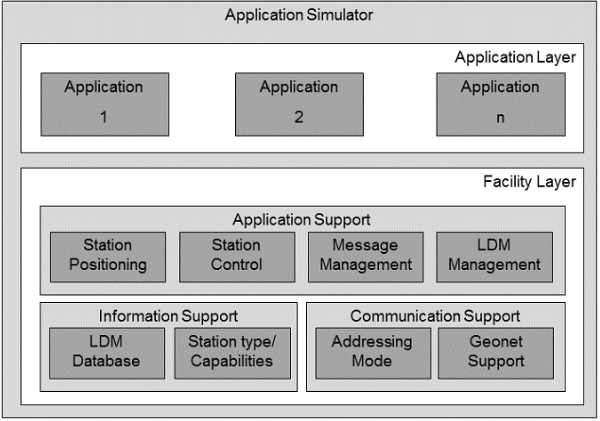
**An example of a VSimRTI simulator coupling**

Above diagram shows an example of combination of different simulators to simulate the vehicle ad-hoc network. In contrast to existing fixed simulator couplings, the VSimRTI simulation infrastructure allows the easy integration and exchange of simulators. Thus, the high flexibility of VSimRTI enables the coupling of the most appropriate simulators. VSimRTI uses an ambassador concept inspired by some fundamental concepts of the High Level Architecture (HLA).  Attaching an additional simulator only requires that the ambassador interface is implemented and, then, the specified commands are executed.

For below simulators ambassador is already developed.

1. Traffic simulator SUMO
2. Network simulator Ns3 and omnetpp
3. Application simulator VsimRti\_AppNt
4. Several visualization and analysis tools

**The VSimRTI Application Simulator:** VsimRti application is used to simulation of smart mobility application. Application runs in sandbox that offers vehicle like interface. It implements the ETSI-ITS standards but, decongestion control (DCC) one of the main component of the ETSI protocol is not implemented.

****

**Features of the VSimRTI Application Simulator**

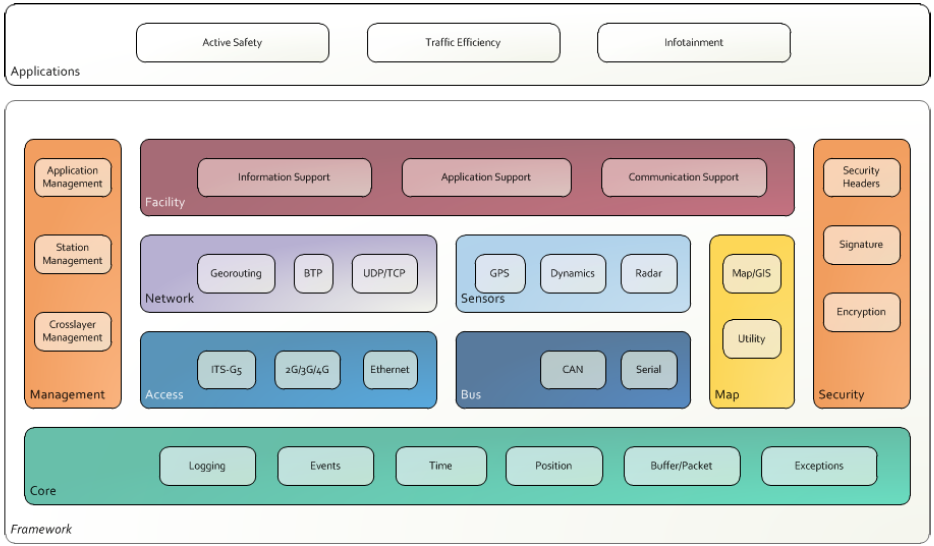
The individual aspects should be explained in the following:

* **Station Positioning:** The realistic movement information from the traffic simulator is provided for the application of the EGO-vehicle. This information is already formatted in WGS-84 GPS coordinates to simulate real navigation system. Furthermore, Map and own Route Information are available.
* **Station Control:** The Vehicle is able to sensor events, which can be simulated by the Environment Simulator. The Vehicle can moreover directly influence its driving maneuvers by slowing down, accelerating or changing the route to another road.
* **Message Management:** From the V2X domain, several message types are supported. Most prominent is the periodical CAM-generation following specified rules (by ETSI standard, by User-timing, etc.). Event based DENMs are supported as well. Besides, further message types as SPAT (signal phases from intelligent traffic lights) are also included.
* **LDM-Management and LDM-Database:** Currently the features of a dynamic knowledge base a.k.a. local dynamic map are implemented following the concepts of the LDM++. These concepts respect the information dynamics from static to highly dynamic without discretization loss due to the LDM layer structure from the SOTIS approach. Rules in the LDM-Management then control which information should be aggregated and which information should be used directly, according to the lifetime of the information.
* **Station type/ Capabilities:** Simulated stations in VSimRTI can be specified with different capabilities. The configuration is part of the VSimRTI Mapping Component, which supports different kinds of vehicles and certainly also Roadside Units as well as intelligent Traffic Lights.
* **Addressing and GeoNetworking:** VSimRTI Applications have the necessary APIs to control the according addressing mode. While the actual routing is implemented in the Communication Simulator, the Application Simulator supports different addressing schemes for ad-hoc communication (Unicast, Broadcast, Geocast) as well as for communication over the infrastructure-based cellular networks.

It is not an open software. As for evaluation, development team provided a limited time license. However, to run simulations every time license has to be authenticated by them, it is failing most of the time due to cognizant proxy and firewall and we have very little support from the vsimrti team.

**Ezcar2x**: ezcaer2x is a framework developed by **Fraunhofer** **institute**. It is not an open source simulator but they are planning to make it open source by end of 2019.

Ezcar2x is a collection of reusable software components for protocols and application around connected vehicle network. It is event driven simulator. It has completely implemented ETSI communication stack. A flexible software framework for prototyping cooperative driving assistance and intelligent transport system simulations. it Is developed in c++ language.

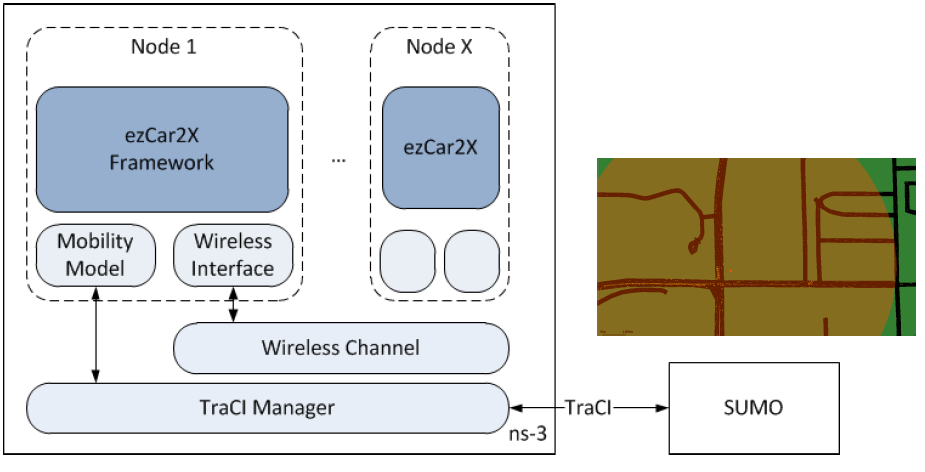


**Architecture of ezcar2x**

Above picture shows architecture of ezcar2x. //write something about figure//

The advantages of simulator are

1. Flexibility
2. Extensibility: Integration with hardware devices and communication technologies is possible.
3. Portability: developed algorithms can be easily integrated to different platforms. Example ARM, Android... Etc.



**Ezcar2x components**

As shown in figure It uses **ns3** as network simulator and **sumo** as traffic simulator. Main concepts of ezcar2x are:

1. Abstraction: abstract classes define functionality of components.
2. Event-driven design.

Cognizant is planning to procure the ezcar2x simulator license from this institute but it will take some time.

Notes:

Overview (introduction):   
 Description for Sybil attack (Problem Statement)

1. Explanation of Vanet (Brief introduction)   
 - Subheading:   
 - Anatomy of Vanets   
 - PKI Infrastructure

2. Section for standards

3. Section for Simulators

4. Current work in Sybil attack

5. Challenges Faced and why we opt for that particular Simulator

6. Delivery outcome from 4Pi

7. State of

8. Flow of work items which are doing (State of work) (timeline of work Items)

9. Current State of work

10.Future work

11.Business Ideas how we can link for project

12.Challenges we faced and approach taken to solve them

13.Reference   
 - Thesis   
 -

14.