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

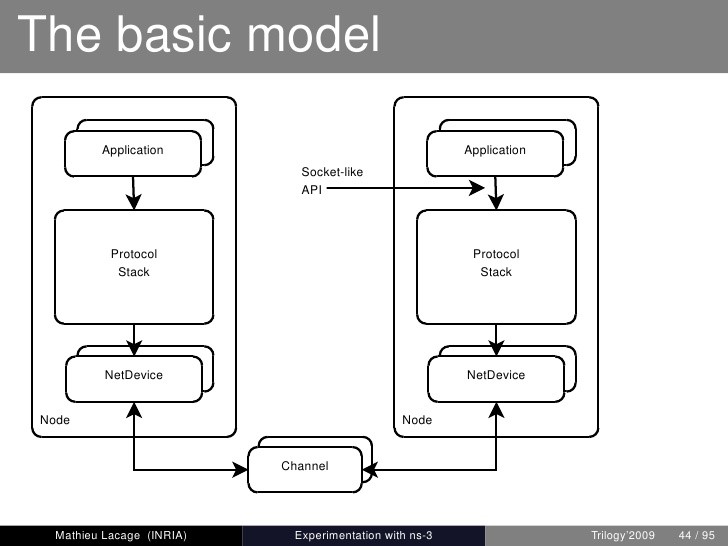


FIG 3: NS3 basic model.

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 IEEE standards and ETSI standards. For our project ETSI standards are used. Vanetza is an open library and written in c++ language. This library partially implemented ETSI-ITS protocol stack. 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.

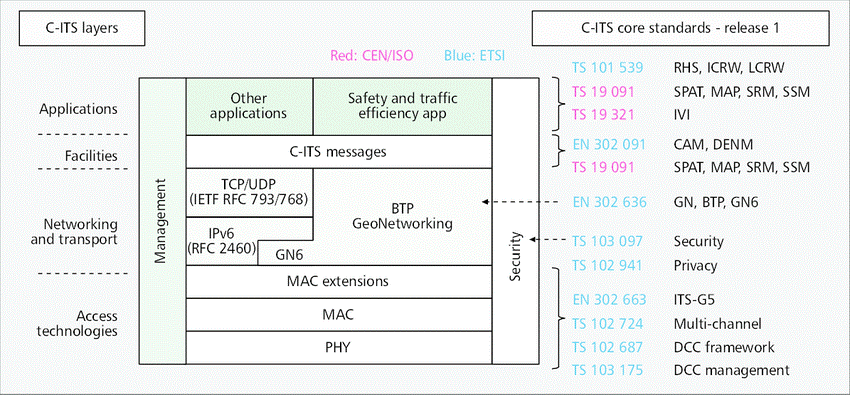


FIG 1: ETSI ITS STANDARDS LAYERS

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

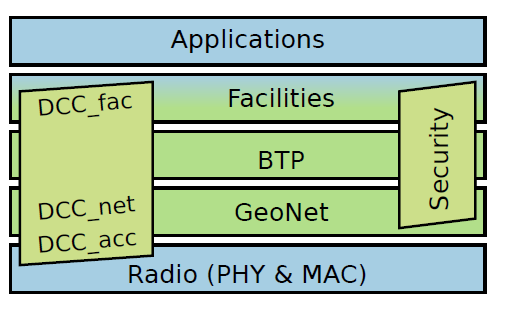


Fig 2: layers supported by vanetza (green) in its g5 architecture

From above diagram, vanetza only implemented geonet, btp, and some part of the facilities layer in ITS g5 architecture along with security and DCC (decongestion control).

**METHOD**:

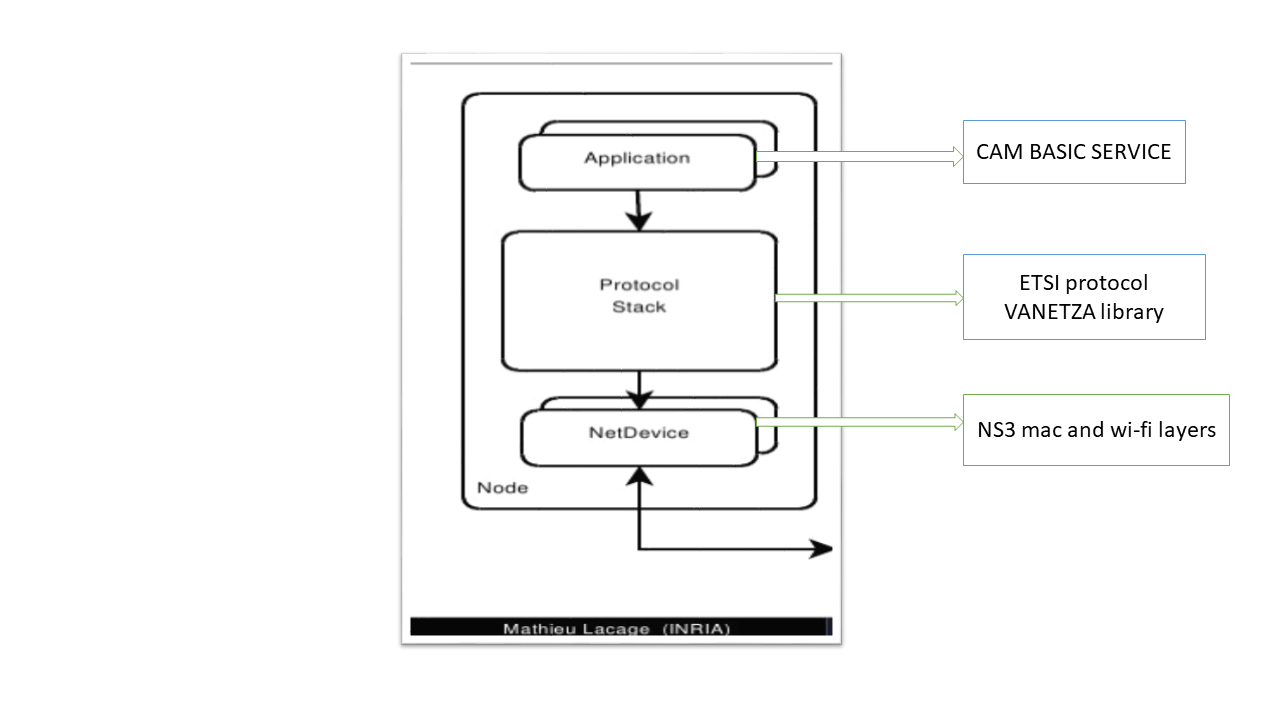
Our main objective is to integrate the Ns3 simulator and vanetza library to create ETSI protocol stack and simulate the vehicle-to-vehicle network. To do this let us understand ns3 and vanetza.

Ns3 has already implemented some part of the IEEE standards, mainly the lower layers (physical and mac) of ETSI ITS.

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.

STEPS:

1. Integration of the vanetza library with ns3 library.
2. Understanding the physical and mac layer already existed in ns3.
3. Implementation and execution of basic cam application and testing it with already existed internet protocol.
4. After the step 3 with help of vanetza, creating ETSI protocol stack in ns3.
5. After step four testing the basic cam application with the complete protocol stack.



**PROGRESS**:

Step 1: Completed.

Step 2: Partially completed.

**Step 3: partially completed**.

**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.

**Steps:**

1. Reading EN 302-637 protocol to understand CA basic service.
2. Replicating CA service application in ns3 same as in artery.

**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.

Step 4: Yet to be started.

Future work:

In Future, our aim is to complete step 3 as soon as possible and run it in simulator. If there is any error and bugs are present, we need to debug it. Then proceed to step 4 and complete it as soon as possible.

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. This itself is a major project. However, vanetza an open library that can be integrated with the ns3 simulator is providing ETSI protocol stack.

**SIMULATORS**

For selecting a simulator for our project, multiple simulators were explored and pros and cons of every simulator were studied.

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.

Traffic simulator: Traffic simulator is a mathematical modeling of transportation system.

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 is a module built top on **Omnetpp** network simulator and it uses **SUMO** as traffic simulator. 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. It uses **ns-3** or **omnetpp** as network simulator and **SUMO** or **vissim** as traffic simulator. It implements the ETSI-ITS standards but, decongestion control (DCC) one of the main component of the ETSI protocol is not implemented. 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. They claim that it has completely implemented **ETSI-ITS** standards along with security. It is ideal simulator for project.

It uses **ns3** as network simulator and **sumo** as traffic simulator. However, to use ezcar2x simulator we must have a clear understanding of the **Ns3** network simulator and **sumo** traffic simulator. Sumo simulates traffic scenario but Ns3 is the main simulator in which implementation the vehicle-to-vehicle network scenario.

Cognizant is planning to procure the ezcar2x simulator license from this institute but it will take some time. So to prepare for the ezcar2x simulator and if we are unable to get hands on ezcar2x for plan B we initiated our own project of integrating Ns3 and vanetza to simulate vehicle-to-vehicle network with ETSI protocols.

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.