# Research Proposal

## *VANETS - Vehicular Ad-hoc Networks*

Submission by Jonathon Davis, Nico Gibson, Samantha Jackson

## 1. References

*A Survey on Network Simulators for Vehicular Ad-hoc Networks (VANETS)*
*Veins - Vehicle in Network Simulation*
*Bidirectionally Coupled Network and Road Traffic Simulation for Improved IVC Analysis*

## 2. Summary (VANETS)

### Overview

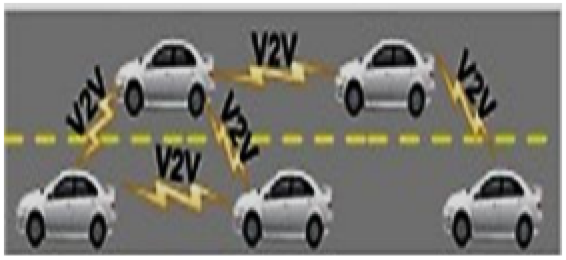
This research discusses technology based on mobile ad hoc wireless network (MANET) infrastructure called vehicular ad hoc networks (VANETs). This technology allows vehicles to communicate via infrastructure to exchange information and warning messages.

### Goals

- Develop VANET performance using network simulators.

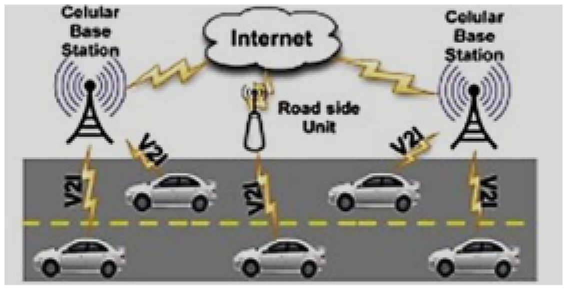
### Background

- VANET communication is classified into two categories:
    - Vehicle-to-Vehicle (V2V) Communication
        - Communication between different vehicles with On-Board Unit (OBU) devices.

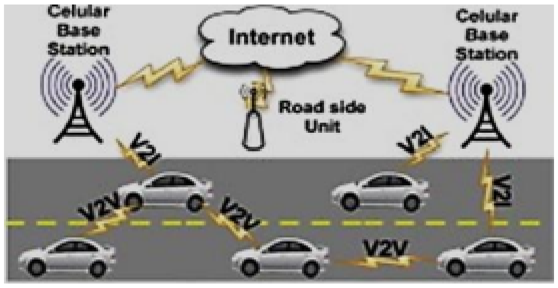


    - Vehicle to Infrastructure (V2I) Communication

        - Communication between vehicles with OBU and Road-Side Unit (RSU) using electromagnetic waves.



    - The combination of these two architectures is considered a Hybrid architecture which can be used for:
        - Avoiding accidents at intersections without traffic signals
        - Altering traffic jamming
        - Accident detection
        - Detecting traffic slowdown



- The modes of operation in VANET can be divided into topology and geographic:
    - Topology-based routing can be unicast or multicast and contain two groups
        - Proactive: Maintains information and updates network topology information in routing tables.
            - Destination-Sequenced Distance Vector (DSDV)
        - Reactive: Do not sustain routing information and locate routes only when source nodes need to transmit data to a destination node.
            - Ad hoc On-demand Distance Vector (AODV)

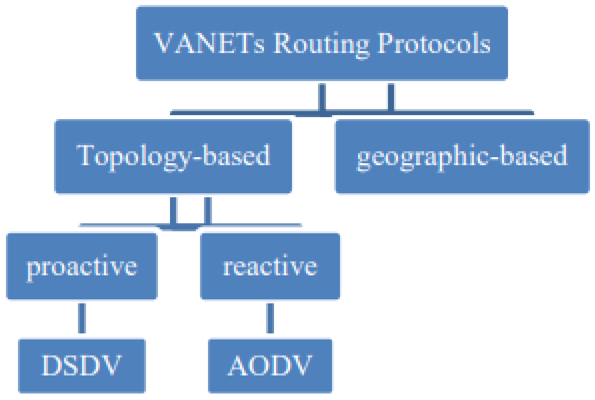
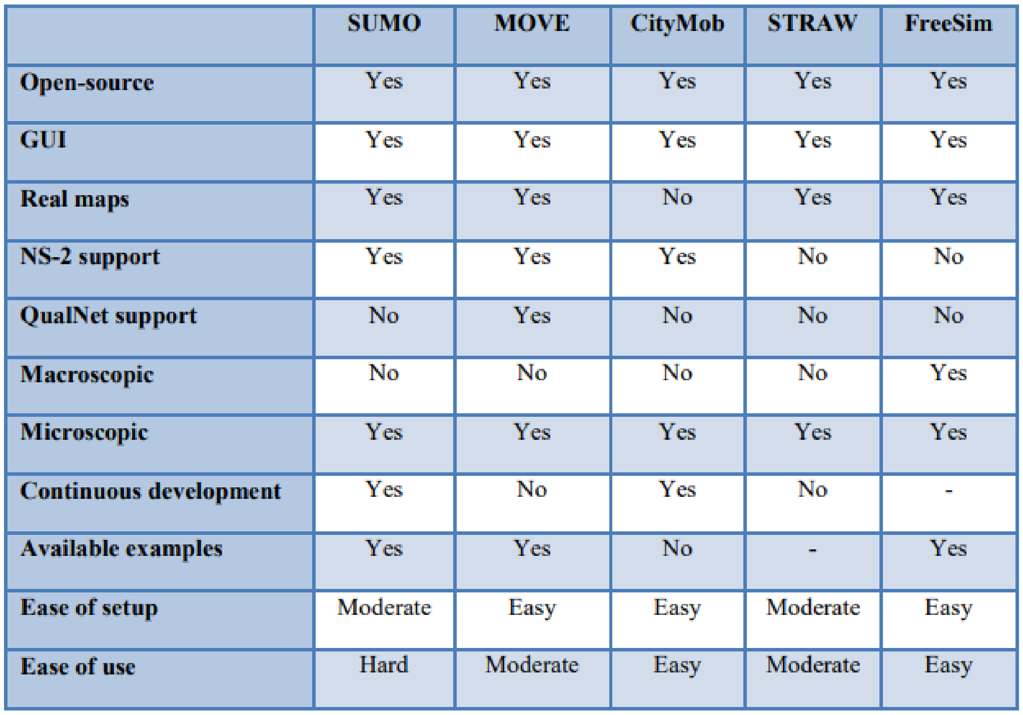


**Fig 2: Routing protocols in VANETs**

## Simulators

- There are many open-source VANET simulators available, these mobility simulators add a level of realism through road models and scenario parameters. To be accurate these simulators should contain: Accurate and realistic topological maps, vehicle acceleration and deceleration, obstacles, attraction points (final destinations), weather and traffic conditions, and driving patterns:
    - **Simulation of Urban Mobility (SUMO):** Free, downloadable, microscopic, and continuous traffic simulation package for large road networks implemented in C++ and Python. It enables the modeling of road vehicles, public transport, and pedestrians. It can also handle tasks like finding routes, visualizing, importing networks, and emission calculations.
    - **Mobility Model Generator for Vehicle Networks (MOVE):** Created on top of the SUMO simulator implemented in Java. It includes a map editor and a vehicle movement editor.
    - **City Mobility Generator (CityMob):** Open-source simulator written in C. Enables users to create urban mobility scenarios like floods. It has three mobility models: the Simple Model, the Manhattan Model, and the Downtown Model.
    - **STreet RAndom Waypoint (STRAW):** Offers precise simulation results using mobility in a vehicle in real vehicle traffic in US cities. Limited by map knowledge which causes restricted mobility.

- **FreeSim:** Macroscopic customizable and microscopic free-flow simulator allowing for multiple freeway networks for quick visualization. Can build and execute traffic and graph algorithms. The source code is freely available for download.

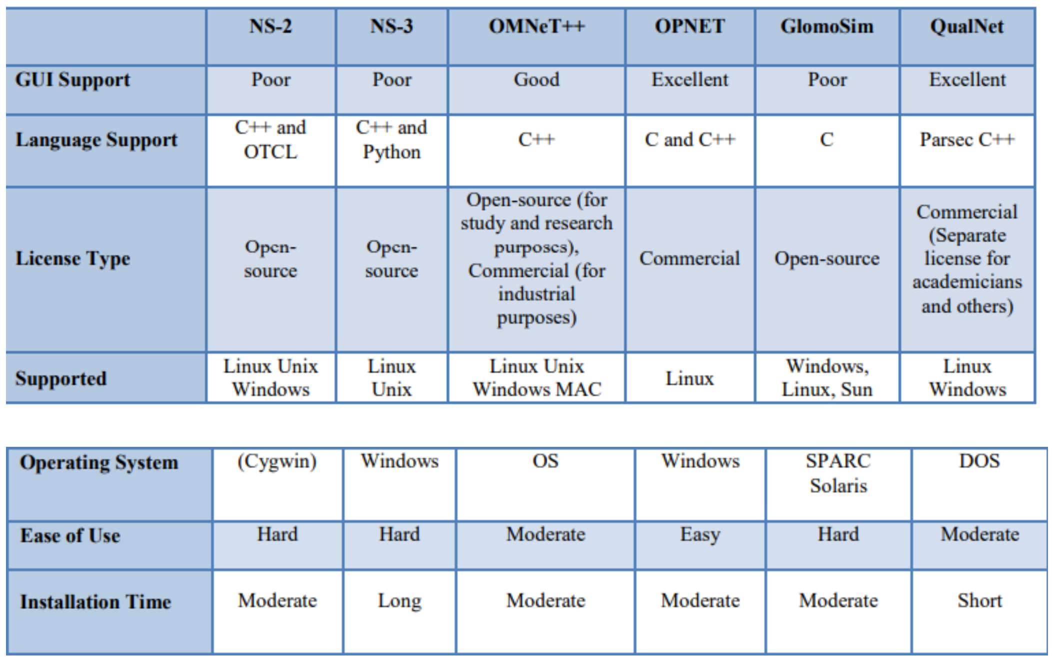


| | SUMO | MOVE | CityMob | STRAW | FreeSim |
|---|---|---|---|---|---|
| **Open-source** | Yes | Yes | Yes | Yes | Yes |
| **GUI** | Yes | Yes | Yes | Yes | Yes |
| **Real maps** | Yes | Yes | No | Yes | Yes |
| **NS-2 support** | Yes | Yes | Yes | No | No |
| **QualNet support** | No | Yes | No | No | No |
| **Macroscopic** | No | No | No | No | Yes |
| **Microscopic** | Yes | Yes | Yes | Yes | Yes |
| **Continuous development** | Yes | No | Yes | No | - |
| **Available examples** | Yes | Yes | No | - | Yes |
| **Ease of setup** | Moderate | Easy | Easy | Moderate | Easy |
| **Ease of use** | Hard | Moderate | Easy | Moderate | Easy |

- There are various types of commercial and open-source simulators. Commercial simulators are often more maintained, with good documentation and bug fixes. Open-source simulators are adaptive and reflect new and current technology and topologies but offer little documentation and support.
  - **NS-2:** Discrete simulation in Python and networks in pure C++ and OTCL. Includes: mobility of nodes, functional physical layer with a radio propagation device model, interacts with the radio network, IEEE 802.11 MAC (Medium Access Control) protocol with Distributed Coordinating Feature (DCF). Has defects in design and modeling.
  - **NS-3:** Discrete event network simulator for Internet systems. Mainly for academic and educational purposes. Not an extension of NS-2. Written in C++ and Python. Networks can be implemented in pure C++ and part of simulation in Python.

- **OMNeT++:** Extensible, modular, component-based C++ simulation library and framework. INET architecture is maintained by the OMNeT++ team using patches and new models submitted by community members.
- **OPNET:** Licensed network simulator for wireless and wired networks in C and C++. Allow users to design and study network communication, protocols, devices, applications, and routing protocols. Supports IEEE 802.11, 802.15.1, 802.20, and satellite networks.
- **Glomosim:** Global Mobile Information System Simulator targeted towards wireless network simulation. Coded in Parsec, contains a front end based on Java. Can run on an SMP (shared-memory symmetric processor) which reduces the load on the CPU allowing it to simulate tens of thousands of nodes in a single simulation.
- **Qualnet:** Quality Networking is a program for network evaluation and is written entirely in C++. Built for testing purposes on a layered architecture.

**Table.3 A Features Comparison of the Various Network Simulators**

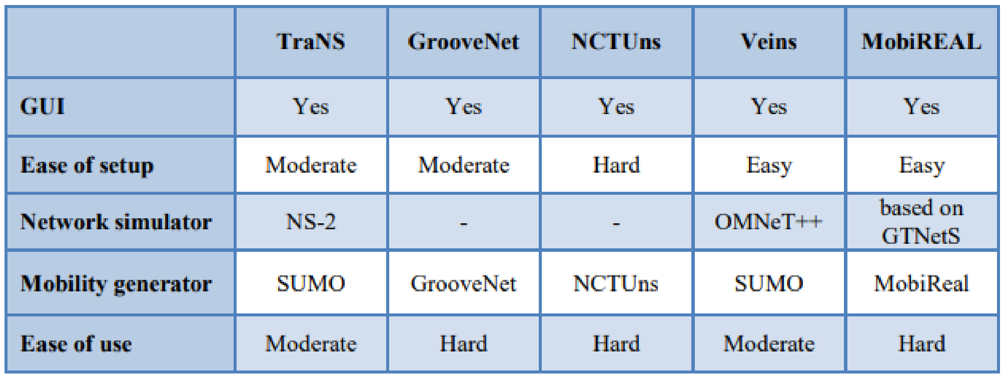


| | NS-2 | NS-3 | OMNeT++ | OPNET | GlomoSim | QualNet |
|---|---|---|---|---|---|---|
| **GUI Support** | Poor | Poor | Good | Excellent | Poor | Excellent |
| **Language Support** | C++ and OTCL | C++ and Python | C++ | C and C++ | C | Parsec C++ |
| **License Type** | Open-source | Open-source | Open-source (for study and research purposes), Commercial (for industrial purposes) | Commercial | Open-source | Commercial (Separate license for academicians and others) |
| **Supported Operating System** | Linux Unix Windows (Cygwin) | Linux Unix Windows | Linux Unix Windows MAC OS | Linux Windows | Windows, Linux, Sun SPARC Solaris | Linux Windows DOS |
| **Ease of Use** | Hard | Hard | Moderate | Easy | Hard | Moderate |
| **Installation Time** | Moderate | Long | Moderate | Moderate | Moderate | Short |

- Drivers' reactions in different situations can impact traffic flow. These programs provide a simulation environment for vehicle actions based on context; simply known as VANET simulators:

- **TRaNS**
- **GrooveNet**
- **NCTUns**
- **Veins**
- **MobiREAL**

**Table.4 A Features Comparison of the Various VANET Simulators**



| | TraNS | GrooveNet | NCTUns | Veins | MobiREAL |
|---|---|---|---|---|---|
| **GUI** | Yes | Yes | Yes | Yes | Yes |
| **Ease of setup** | Moderate | Moderate | Hard | Easy | Easy |
| **Network simulator** | NS-2 | - | - | OMNeT++ | based on GTNetS |
| **Mobility generator** | SUMO | GrooveNet | NCTUns | SUMO | MobiReal |
| **Ease of use** | Moderate | Hard | Hard | Moderate | Hard |

## Proposed Network Model and Results

- To demonstrate performance this paper used the following environment:
  - Basra City in Urban streets using real-time integration of OMNeT++ and SUMO
  - Vehicle speed is 40Km/h and 60 Km/h without considering the obstacles or disorder in signal transmission
  - Vehicle density is represented by nodes between a range of 10-40 ignoring changes in vehicle direction
  - Veins will use a TCP link and Python scripts to allow SUMO to act as a mobility model.

Table 5. Simulation Parameter

| Parameter | Value or Protocol |
|---|---|
| OMNeT++ Version | OMNeT++ V 5.5.1 |
| SUMO Version | SUMO 1.6.0 |
| INET Version | INET 4.2.1 |
| Veins Version | Veins 5.0 |
| Vehicles Number | 10,20,30,40 |
| Speed | 40Km/h, 60Km/h |
| Simulation Time | 600 Seconds |
| MAC Protocol | IEEE802.11p |

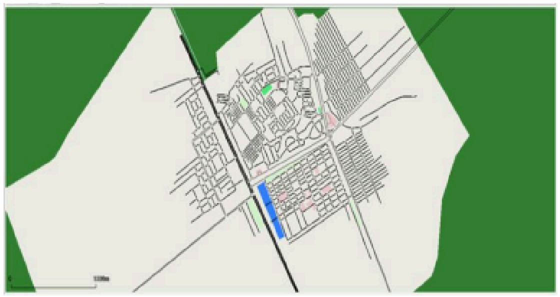
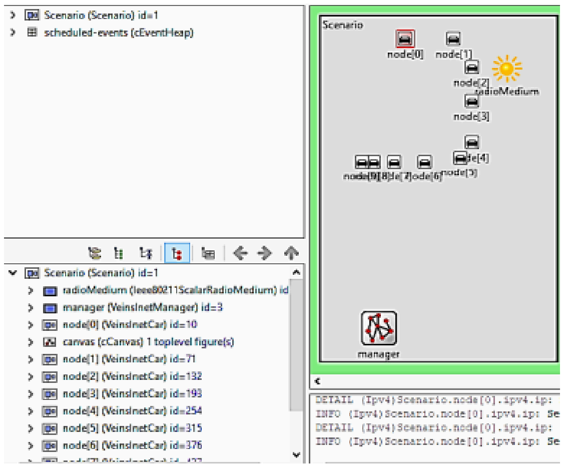


Fig 4: Basrah streets in SUMO 1.6.0


```
> Scenario (Scenario) id=1
> scheduled-events (cEventHeap)
```

```
v Scenario (Scenario) id=1
    > radioMedium (Ieee80211ScalarRadioMedium) id
    > manager (VeinsInetManager) id=3
    > node[0] (VeinsInetCar) id=10
    > canvas (cCanvas) 1 toplevel figure(s)
    > node[1] (VeinsInetCar) id=71
    > node[2] (VeinsInetCar) id=132
    > node[3] (VeinsInetCar) id=193
    > node[4] (VeinsInetCar) id=254
    > node[5] (VeinsInetCar) id=315
    > node[6] (VeinsInetCar) id=376
```

```
DETAIL (Ipv4)Scenario.node[0].ipv4.ip:
INFO (Ipv4)Scenario.node[0].ipv4.ip: Se
DETAIL (Ipv4)Scenario.node[0].ipv4.ip:
INFO (Ipv4)Scenario.node[0].ipv4.ip: Se
```

Fig 5: Simulation using OMNeT++

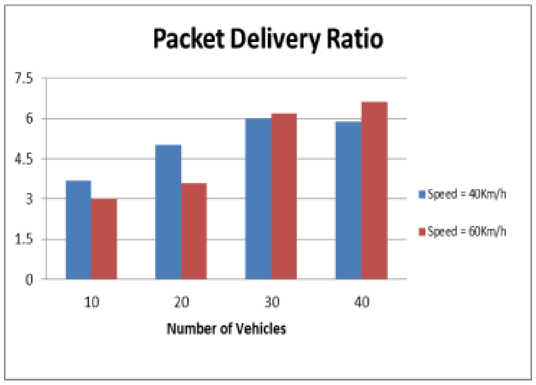


Fig 6: Packet delivery Ratio vs. Number of Vehicles

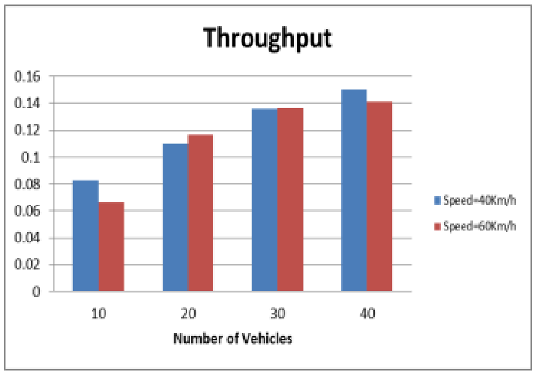


Fig 7: Throughput vs. Number of Vehicles

- Network performance in terms of packet delivery ratio and network throughput were examined using the number of nodes.

$$\text{Packet Delivery Ratio} = \frac{\text{Total No.of Packet Recived}}{\text{Total No.of Packet Sent}}$$

    - Network throughput increases as nodes increase.

$$\text{Throughput} = \frac{\text{Total Data Sent}}{\text{Total Time}}$$

## 3. Development Stack

- Simulation Platform: OMNeT++ 5.6.2
- Simulation Frameworks:
    - Open-Source Communication Networks Simulation Package: INET-4.2.2
    - Urban Mobility Simulator: SUMO 1.8.0
    - Vehicular Network Simulation Framework: VEINS

## 4. Evaluation

- Our project seeks to use the same simulators as the initial research paper but create new simulation parameters. There is also potential to add to existing research through different situations with custom parameters. For example, attacks on VANETs could be potential research opportunities.

## 5. Potential Challenges

- **Simulation Stack:** The amount of simulators in this project could be complicated to set up within OMNeT++. On top of the difficulty of managing multiple simulation frameworks within the project. Thankfully there are a lot of resources on this combination of frameworks.