Australasian Transport Research Forum 2011 Proceedings
28 - 30 September 2011, Adelaide, Australia
Publication website: http://www.patrec.org/atrf.aspx

OpenTraffic - An Open Source Platform for Traffic Simulation

Marc Miska[1], Edgar Santos[2], Edward Chung[1], Helmut Prendinger[2]

[1]Smart Transport Research Centre, Faculty of Build Environment and Engineering, Queensland University of Technology, 2 George St GPO Box 2434, Brisbane QLD 4001, Australia

[2]National Institute of Informatics, The Graduate University for Advanced Studies, 2-1-2 Hitosubashi, Chiyoda-ku, Tokyo 101-8430, Japan

marc.miska@qut.edu.au

Abstract

Traffic models are used to support traffic engineers with the optimisation task by predicting the effects of ITS measures before applying them to the real network. Simulation of traffic networks require a whole suite of applications, from data gathering, cleaning and fusion, over origin/destination (OD) estimation and prediction, to the simulation itself, including route choice, driving behaviour and travel behaviour. Thus, simulation is based on several methods and models developed in several disciplines like system control, mathematics, human behaviour, and vehicle dynamics. To improve existing methods or to develop new methods for traffic management, it is essential to be able to test algorithms in a stable environment that allows access to necessary input (e.g. road network description, traffic measurements, incident reports…), and provides tools for assessment and evaluation. This paper introduces OpenTraffic, an open source platform for traffic simulation allowing for tailor made simulation applications that allows researchers to develop new algorithms in such a stable environment.

1. Introduction

The mobility demand of our society increased rapidly in the last decades. To ensure accessibility the traffic infrastructure needs to be used more efficiently. One way to achieve this is the introduction of dynamic traffic management (DTM). DTM monitors the traffic situation and tries to optimize it by applying different control measures like variable speed limits (VSL), route guidance, lane closure, ramp metering, intersection control or others to the traffic network. In its ideal form it is a continuous cycle that gets evaluated by checking the effects of the taken measures to the traffic situation, although in practice the measures are often predefined and not tuned to the detailed actual traffic situation. Traffic models are used to support traffic engineers with the optimisation task by predicting the effect of different measures before applying them to the real network. Online simulations of traffic networks require a whole suite of applications, from data gathering, cleaning and fusion, over origin/destination (OD) estimation and prediction, to the simulation itself, including route choice, driving behaviour and travel behaviour. Thus, online simulation is based on several methods and models developed in several disciplines like control, mathematics, human behaviour, and vehicle dynamics. To improve existing methods or to develop new methods for traffic management, it is essential to be able to test algorithms in a stable environment that allows access to necessary input (e.g. road network description, traffic measurements, incident reports…), and provides tools for assessment and evaluation. Further, with traffic operations becoming more and more complex, improvements are rather in details than in the overall picture. Therefore, it would be desirable for research to be able to focus on improving

specific parts, and to be able to pair those with State of the Art tools to run performance tests. At present however, research and practise are missing standards for network coding and traffic measurement provision, which results in high costs and workload for any project to start with. The differences in simulation platforms prohibit sharing of calibrated and validated models, which makes modelling not a sustainable task. Further, it remains difficult to enhance simulation packages with new developments for testing and evaluation. In sum, these elements slow down research and deployment of cutting edge technology to ensure mobility of our cities.

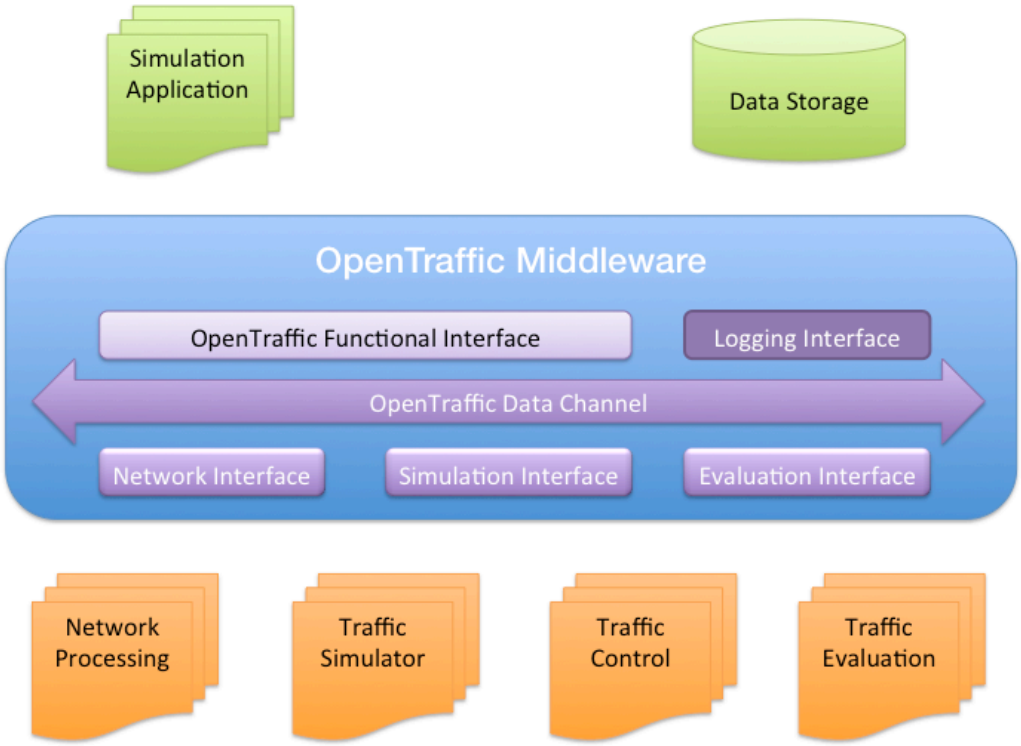
This paper introduces OpenTraffic, an open source platform for traffic simulation allowing for tailor made simulation applications that allows researchers to develop new algorithms in a stable simulation environment without the need to reinvent the wheel. In the following we introduce the architecture of OpenTraffic and before showing a prototype application of a OpenTraffic simulation application, highlighting its benefits.

2. OpenTraffic System Architecture

2.1 Overall Architecture

To avoid the drawbacks of current practice, our approach was to create a common simulation platform that uses toolboxes to perform a traffic simulation. The toolboxes are independently developed, implementing the corresponding OpenTraffic interface, and they communicate with each other by messaging. The OpenTraffic data channel controls the messaging and reports to the simulation application (see Figure 1).

Figure 1: Architectural Diagram of an OpenTraffic Simulation



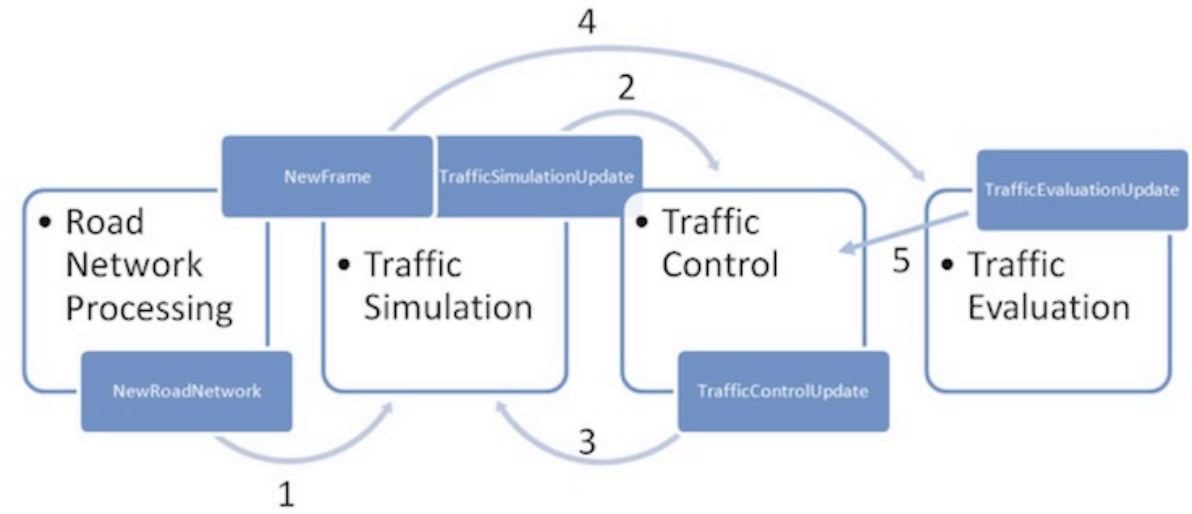
Additional data storage can be used to store the simulation results for off-line post processing. One should note that users could exchange toolboxes to create a tailor made simulation application, without the need of specific knowledge of the remaining toolboxes.

This is the most powerful aspect of OpenTraffic, it allows users to focus on their area of expertise while taking full advantage of State of the Art modules from others. In the following we will describe the components of OpenTraffic in more detail.

2.2 Middleware

To support a modular structure for the platform we have chosen a middleware to coordinate the workflow of a simulation. The middleware supports two types of modules: a logical and a functional.

Figure 2: Messaging between the logical modules of OpenTraffic



The workflow for a simulation application (see Figure 2) starts with the road network processing, which results in a network description that is understood by the simulator. This network description is passed to the traffic simulation to perform the demand generation and vehicle movements. Once all vehicle are moved, and their movements have been registered with the placed sensors, the traffic control and traffic evaluation are triggered to update traffic control states and to evaluate traffic states. Feedback loops are placed between the traffic control and simulation, as well as between the traffic evaluation and traffic control. The first feedback loop represents adaptive or intelligent control measures, while the latter on represents high-level decisions made on a policy level.

2.3 Logical Modules

Logical modules, or toolboxes (e.g. traffic simulation), need to implement the *IOpenTrafficLogicalModule interface* to receive messages through the middleware. The interface consists of the following functions:

```
public bool Load(IOpenTrafficLogicApi api); //called when
module is loaded, argument is the Api the module can call

public bool Unload(); //called when module is being destroyed

public bool Initialize(); //called when simulation logic is
asked to be initialized
```

```
public bool HandleMessage(MessageType type, IMessageData
data); //called when there is a message for this module

public string GetModuleName(); //called when the middleware
needs to know the name of the module
```

The *Load(…)* and *Unload()* functions are used for memory management, so to ensure that the application has no memory leaks and does not require more resources than necessary. At the begin of each workflow, or simulation the *Initialize()* function is called, which allows the developers to instantiate the needed objects for the tasks ahead. Main part is the *HandleMessage(…)* function, that allows developers to react on messages from other modules. Finally, the *GetModuleName()* function is used to identify the module. In the following we will describe the main functions for the logical modules in more detail.

2.3.1. Network processing

The network preparation module handles the translation of a coded transport network into a form that is understood by the used simulator. Necessary for the implementation of the module is the definition of the network by the simulator. With this information, the module can generate the needed information and convert it in the right format. The Smart Transport Research Centre is promoting a model free network-coding standard for this purpose, to maximise portability of not only networks, but also simulation modules, using them.

Existing efforts of standardisation are based on exchange formats for transport related application, such as:

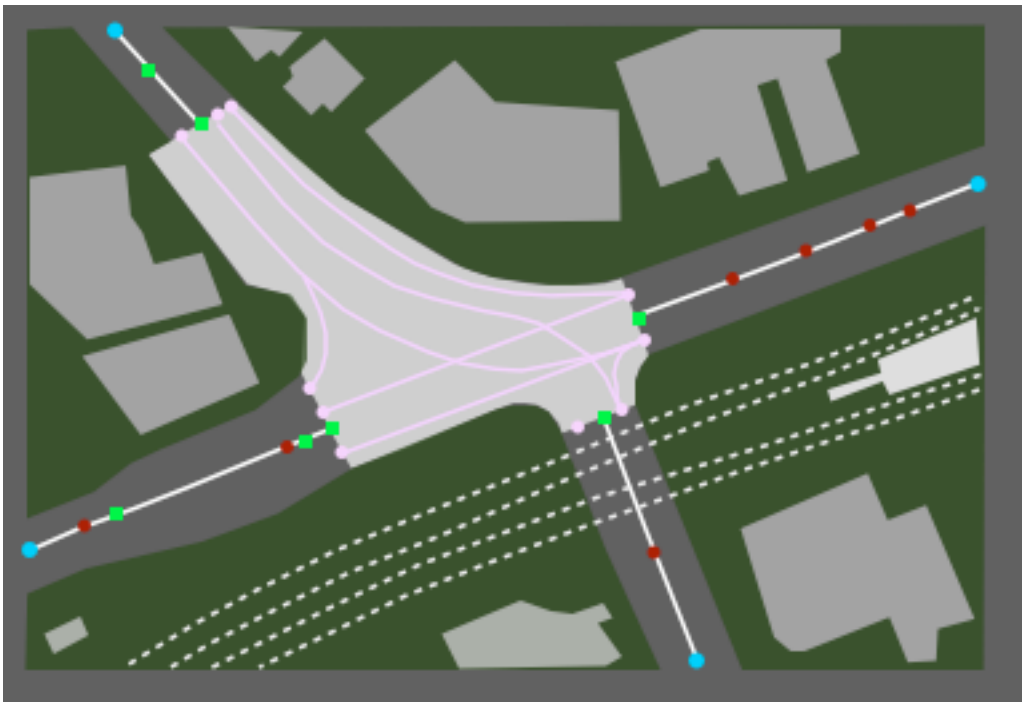
- EuroRoadS (http://www.euroroads.org/)
- RoadXML (http://www.road-xml.org/)
- OpenDrive (http://www.opendrive.org/)
- LandXML (http://www.landxml.org/)
- Abstratct Network Model (ANM)
- OpenMicroSim (http://openmicrosim.org/)

This being a step in the right direction, the problem remains, that all those formats are designed to serve existing models, and do not take into account advances in traffic modelling. The solution is to separate the network description from the model. This means that the network coding shifts from defining the parameters for a model to describing the real world and its physical objects, such as roads, traffic lights, intersections, and so on. The definition is done in three levels:

- Geometry and Terrain modelling
- Physical Network Object description
- Process description

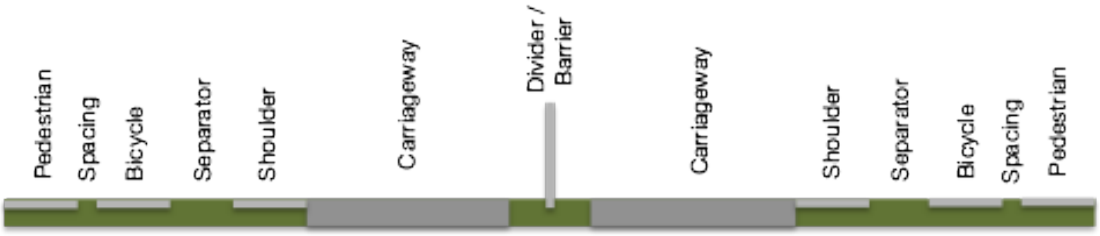
In the geometry and terrain definition, the network is described positions, centrelines, polygons and markers that indicate the position of objects, such as roads, intersections, buildings, train stations and so on. Figure 3 shows an example of an intersection model in the terrain and geometry stage.

Figure 3: Example network representation of an intersection in the model free network standard promoted by the Smart Transport Research Centre



Now that the positions of objects are defined, a further description is needed to utilise them. In the context of this paper, we just illustrate the description of links, while further information can be obtained from the STRC network-coding manual. Links, described by their centreline already are described by their cross-section. The cross-section includes not only the carriageways, but also the whole traffic area, as shown in Figure 4.

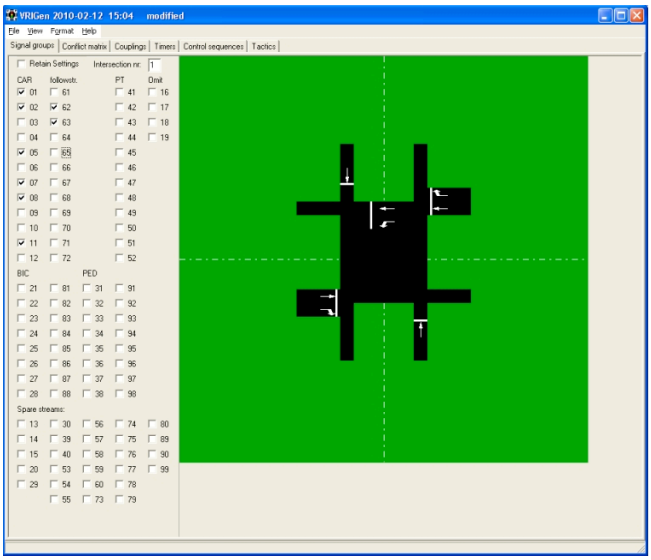
Figure 4: ***Generalised cross-section of a roadway***



The benefit, of such a detailed description is that it contains enough information for various evaluations. The cross-section can change over time to allow for hard shoulder usage, or to integrate parking regulations into the simulation. Further, with more advances in driving psychology, this description contains sufficient information to feed perception modules of driver models. Next to the physical objects, abstracts, such as travel demand are described as well.

Last, but not least is the integration of traffic control equipment to the network description. Since OpenTraffic separates the tasks of vehicle movements, traffic control and evaluation, it is the ideal platform to integrate external traffic control software (e.g., VriGen/Trafcod) as shown in Figure 5.

Figure 5: Example of an *external traffic controller that handles vehicle actuated* signal installations



Additionally, it allows for testing new traffic control schemes or even new detection technology. Detectors can be freely defined and hence allow the creation and testing of new technology before prototyping. Using this three level framework for network coding, OpenTraffic contains a filter to import such networks and to make them available to the simulation components through a simple API, which is included in the online help of OpenTraffic.

2.3.2. Traffic Simulation

After the initial step of the network processing, the traffic simulation module starts the simulation process. Key of this module is the *Frame* object, which stores all dynamic information of the simulation, such as vehicle positions, vehicle dynamics, and detector and control states. While this object, as the network, can be redefined to the user's needs, the Smart Transport Research Centre is working towards a common definition that suits various applications from different fields. In the current version the *Frame* object contains:

- **Vehicle**
  - Position
  - Velocity
  - Acceleration
  - Vehicle Id
  - Link
  - Section
  - Lane
  - Emission
  - Lead Vehicle Id
  - Following Vehicle Id
  - Head Light Status
  - Break Light Status

- **Traffic Light**

    - Position
    - Status
    - Link/ Node
    - Section
    - Lane
- **Display**
    - Position
    - Message
    - Link
    - Section
    - Lane
- **Sensor**
    - Position
    - Collection info
    - Current Reading
    - Link/Node
    - Section
    - Lane
- **Emission Cloud**
    - Position
    - Connected sensors
    - Current Reading

This information will be passed between the modules and together with the network information is the sole basis for visualisation.

During the initialisation of the traffic simulation, the *Frame* object is created and memory is allocated. Then the simulation can be triggered by the application through a *StartSimulation* message. Once the message has been received, the traffic simulation follows in its initial version the simulation steps of:

- Demand generation
- Vehicle update
- Sensor update

During the demand generation, the module determines how many vehicles are to be generated during the time step, calculates the route choice, and places the vehicles in the frame. Then the vehicle movement takes place, on a vehicle by vehicle basis, followed by a sensor update, during which each sensor updates its reading for the predefined measures. The workflow of the simulation is not fixed, and can be freely edited, as long as the Frame object is updated, which is the one common interface between the modules. After those steps have been performed, the *Frame* gets send to the Traffic Control module.

2.3.3. Traffic Control

Once the Traffic Control module receives a *TrafficSimulationUpdate* message, it will read all sensor information, feed it to the traffic control logic or external controllers, and determines the state of displays in the network. Those can be traffic signs, routing panels, park guidance system, ramp metering, and others. One should keep in mind that the traffic control is fully independent from the simulation. Hence, the traffic control can be implemented without knowledge of how the car movement is done. The input extracted from the *Frame* is the detector information, such as in real-world applications.

Recognising that traffic control in today's complex environment is not always so straightforward, additional messages can be send to the traffic control module (i.e., from the

traffic evaluation). This allows more flexibility and a wider decision base for intelligent traffic control systems.

2.3.4. Traffic Evaluation

Once the simulation module has created a new Frame, the traffic evaluation module gets notified. This module aims to serve decision-making processes on a network level, rather than local level as simple traffic control. One can either use this module for assessment studies, or as an additional feedback loop to the traffic control module to implement high-level policies. In our example, we use the evaluation to limit vehicle access to the city, depending on the level of CO2 in the area. If CO2 levels reach a threshold, the traffic controllers of the traffic lights in the vicinity reduce the throughput of cars into the area.

One should keep in mind that this is just an example application for demonstration purposes, and not meant as an attempt to lower CO2 emissions inside a city, by just moving it to another area.

Once the evaluation has been done, messages are sent to the traffic control module for updating the logic of the traffic controllers, and the workflow starts again. Having all modules separated, also allows running the modules in different time steps. One could chose to run the traffic control updates on a second by second basis, while vehicle movements are updated every 1/10s.

2.4 Functional Modules

Functional modules (e.g. simulation application) need to implement the *IOpenTrafficFunctionalModule interface* to receive messages through the middleware. A module that implements this interface will receive every generated frame object through:

```
public void NewFrame(Frame frame);
```

This frame is the result of the workflow, described in Section 2.2 and allows the application to access all information of the ongoing simulation, to be used for visualisation. The basic OpenTraffic Simulation Suite consists of a network editor in the Smart Transport Centre network-coding standard, and a visualisation of the simulated network. For users to create their own tailor made simulation application it is only necessary to create customised versions of the simulation, control, and evaluation module. Components of these modules, such as various car following model implementations can be downloaded from our website.

Additional modules are in development, such as a 3D environment (Miska, 2010), that creates a virtual world based on the network description, and allows for human driving in it (Jiang, 2010). For this, an interface is created to capture input from a driving gear connected to the computer and to feed the information into a virtual world based on Unity. This feature allows for virtual driving experiments (Prendinger, 2011), such as performed with driving simulators, but with the advantage of much lower costs and the possibility of multi-user driving (Nakasone,2011).

3. Prototype Implementation

As a demonstration for this paper, we have implemented a prototype that includes the main features of OpenTraffic. Goal is to limit the CO2 emission of traffic in front of QUT's Gardens Point campus in Brisbane. If the value rises above 150 units, the traffic lights in the area will

reduce the throughput of vehicles into the area. Once the level of CO2 is lowered below 80 units, the controllers allow again more traffic into the area.

3.1 Area

The area of interest is in Brisbane's CBD. The campus lies on Alice St., a one-way road feeding the motorway and is connected to a grid network of downtown Brisbane.

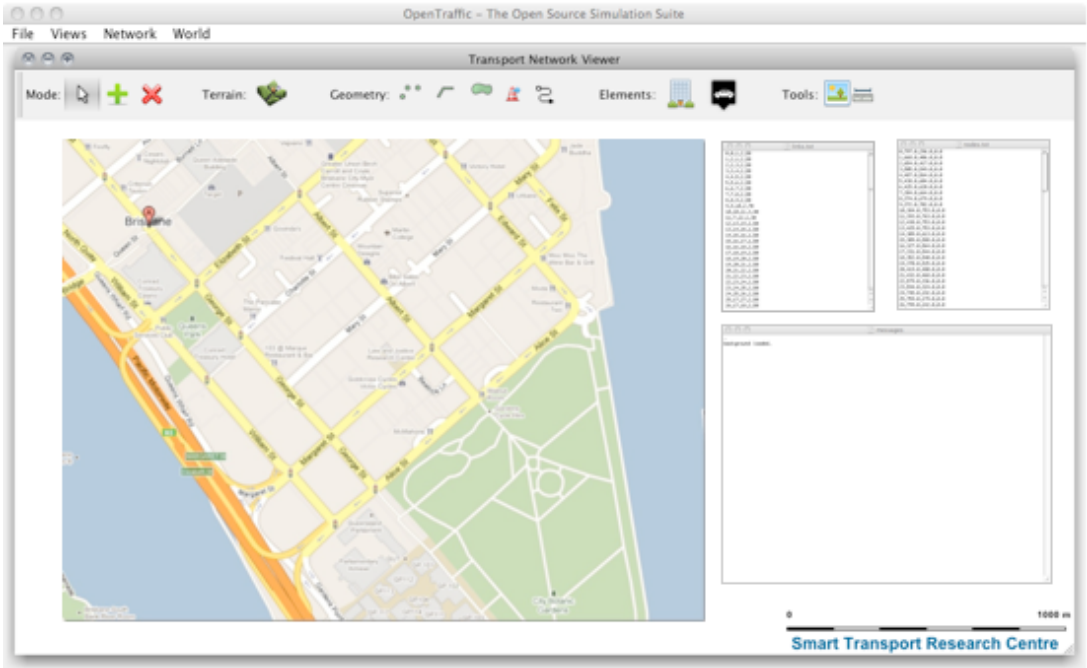
Figure 6: Intersection at QUT's Gardens Point Campus and surrounding street network

Figure 6 shows the area on a map and in Google Street view. Due to the high number of one-way streets in downtown Brisbane, the simulated area has been chosen a bit bigger, to allow a more effective control of vehicle access to the area. The network coding is described in the following.

3.2. Network coding

The network coding is done with the OpenTraffic network-coding tool that is included in the base application (see Figure 7).

Figure 7: Network modelling tool of OpenTraffic for network representation in the Smart Transport Research Centre network-coding standard

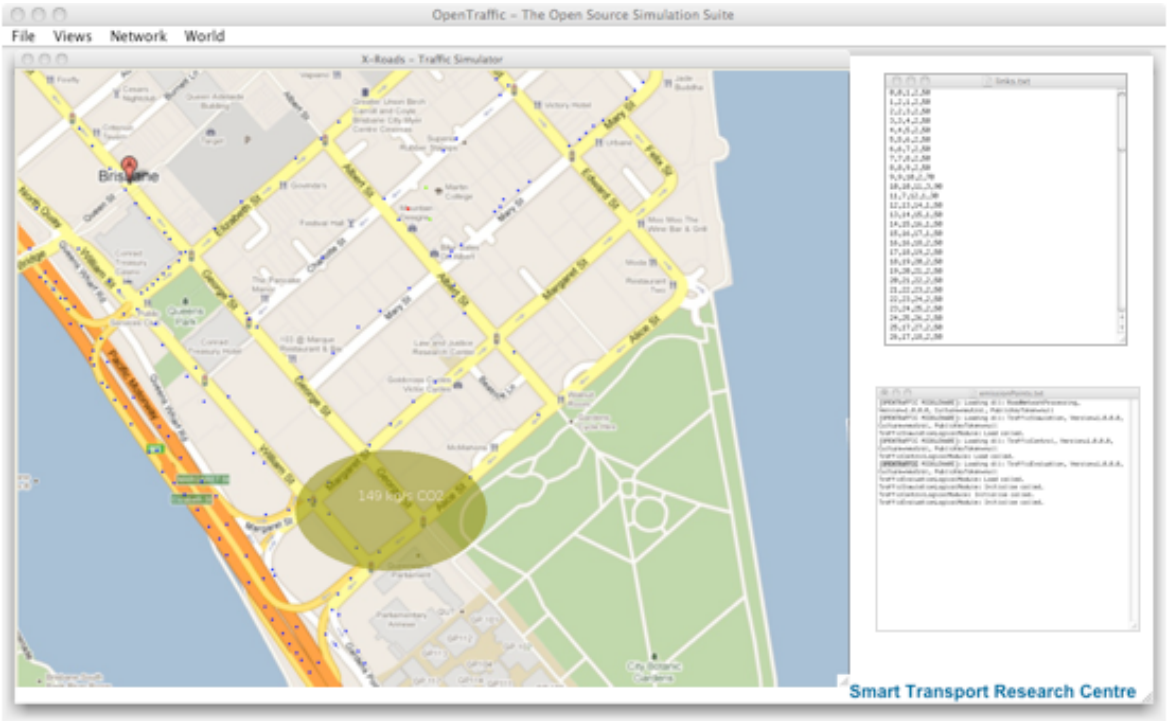


On a map background, we have generated the network based on 136 nodes, 181 links and 21 traffic areas. Traffic lights have been placed at the intersections, connected to a fixed-time controller for simplicity. Further, CO2 detector has been placed at the intersection in front of Gardens Point Campus.

3.3. Simulation

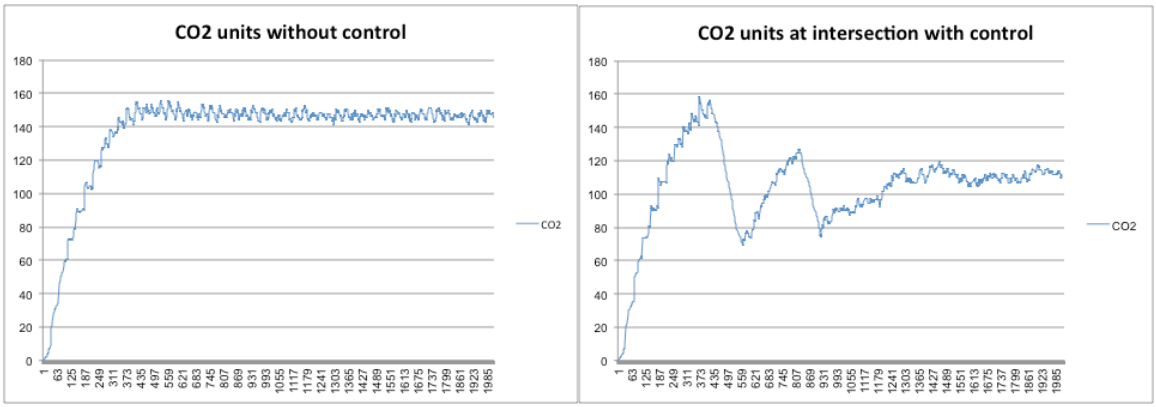
For the simulation prototype we have coded a microscopic simulation module based on the Intelligent Driver Model for car following and the MOBIL lane changing model. The CO2 emission is analysed in a coded evaluation module, based on a statistical model developed at MIT and the control module is implemented as a standard fixed time traffic controller in which we can adjust the cycle time and green split. During the simulation, the emission cloud is visualised on top of the simulated traffic as shown in Figure 8. We have run the simulation with and without the traffic evaluation module to check the difference.

Figure 8: Screenshot of the simulation, showing the network, traffic and emission cloud



As the simulation shows (Figure 9), the feedback loop from the evaluation module manages to limit the CO2 units measured at Gardens Point Campus below the required 120.

Figure 9: CO2 units measured at the intersection with and without applied evaluation feedback



Again, the importance of the result is not the reduction in CO2 emissions, but that four independent modules are performing a traffic simulation.

4. Conclusion

In this paper we have presented OpenTraffic, an Open Source traffic simulation Suite, that allows to develop the tasks of simulation, traffic control, traffic evaluation independently. This concept allows researchers to focus on the field of their expertise without deeper knowledge of the other modules. Traffic control algorithms can be tested in a stable and fixed environment that reliably reproduces the same conditions, and without the limitation from APIs to commercial simulation packages. For practitioners, this opens the world to tailor made simulation models that can be adjusted for a specific purpose, and allows utilising cutting edge technology, without the waiting time until it is incorporated into a commercial package.

Future work will focus on further standardisation of the *Frame* as central component of the simulation, and to promote the usage of OpenTraffic as a framework. This would lead to a community effort that could generate better traffic models faster and offers timely solutions for policy makers to ensure mobility and liveability of our cities.

References

Nakasone, A., Prendinger, H., Miska, M., Lindner, M., Horiguchi, R., Ibarra, J. C., Gajananan, K. Mendes, R. Madruga, M. and Kuwahara, M., OpenEnergySim: A 3D Internet based experimental framework for integrating traffic simulation and multi-user immersive driving. Proc 4th Int'l Conference on Simulation Tools and Techniques (SIMUTools'11), Industry Track, Barcelona, Spain, 2011.3.

Prendinger, H., Nakasone, A., Miska, M. and Kuwahara, M.. OpenEnergySim: Conducting behavioral studies in virtual worlds for sustainable transportation. Proc IEEE Forum on Integrated and Sustainable Transportation System (FISTS'11), Vienna, Austria, 2011.6.

Miska, M., Prendinger, H., Nakasone, A. and Kuwahara, M., Driving and traveller behavior studies using 3D Internet. Proc 13th Intl IEEE Conf on Intelligent Transportation Systems (ITSC'10), Madeira Island, Portugal, 2010.9.

Jiang, T., Miska, M., Kuwahara, M. Nakasone, A. and Prendinger, H., Microscopic simulation for virtual worlds with self-driving avatars. Proc 13th Intl IEEE Conf on Intelligent Transportation Systems (ITSC'10), Madeira Island, Portugal, 2010.9.

Wikström, L., EuroRoadS Deliverable D6.3, Road Network Information Model, http://www.euroroads.org/, 2006

Ducloux, P., RoadXML Specification 2.2, http://www.road-xml.org/, 2009

Dupuis, M., et.al., *OpenDRIVE* Format Specification, Rev. 1.3, http://www.opendrive.org/, 2010

http://www.landxml.org/schema/LandXML-1.2/documentation/LandXML-1.2Doc.html

http://openmicrosim.org/