LEARNING omnet++

#omnet++

Table of Contents

About 1
Chapter 1: Getting started with omnet++ 2
Remarks 2
Examples 2
Installation or Setup 2
Chapter 2: Simulation of VANETs using OMNeT++ and Artery 3
Introduction 3
Examples 3
The Artery Simulation Framework 3
Feature Overview: 3
How does Artery work? 3
Installation 4
Installing OMNeT++ 4
Installing SUMO 4
Installing Artery 5
Running an Example 5
Running an example in debug mode 6
Adding my own ITS-G5 Service: Where should I start? 6
Credits 7

About

Chapter 1: Getting started with omnet++

Remarks

This section provides an overview of what omnet++ is, and why a developer might want to use it.

It should also mention any large subjects within omnet++, and link out to the related topics. Since the Documentation for omnet++ is new, you may need to create initial versions of those related topics.

Examples

Installation or Setup

Detailed instructions on getting omnet++ set up or installed.

Read Getting started with omnet++ online: https://riptutorial.com/omnetplusplus/topic/9564/getting-started-with-omnetplusplus

Chapter 2: Simulation of VANETs using OMNeT++ and Artery

Introduction

This documentation will give you a insight how European VANETs based on the ETSI ITS-G5 and IEEE 802.11p can be simulated by using the discrete event simulator OMNeT++.

Examples

The Artery Simulation Framework

Artery (https://github.com/riebl/artery) is a simulation framework for *OMNeT++*. It is designed to simulate the *ETSI ITS-G5* protocol stack which is used in European *VANET*s. Currently, it is using OMNeT++ in version 5.0. It is completely **open source** and under **active development**.

Feature Overview:

- Day One applications like *Cooperate Awarenes Message* (CAM) and *Decentralized Environmental Notification Message* (DENM) specified by *ETSI ITS-G5*
- It provides common *Facilities* for this applications
- It allows rapid prototyping to create new applications easily
- Geographic routing, which is handled by *Vanetza* (https://github.com/riebl/vanetza)
- *Vanetza* also covers the *Basic Transport Protocol* (BTP)
- It uses *Veins* (http://veins.car2x.org/) or *INET* (https://inet.omnetpp.org/) as IEEE 802.11p MAC layer; these provide also various physical layer wave propagation models
- Realistic vehicle movement provided by the open source traffic simulator SUMO (http://www.dlr.de/)
- An easy way to create different traffic scenarios (like traffic jams or weather influences)

How does Artery work?

Artery is basically composed of four parts. First, there is the network simulation which handles the sending and receiving of packets. This is based on *OMNeT++* networks, where *Veins* or *INET* provide reasonable realistic models of the radio medium. Also, *Veins* and *INET* both provide an implementation of the IEEE 802.11 physical and link layers. You are free to choose which framework you want to use by changing the networks configuration file.

The routing between the network participants is done by *Vanetza*. *Vanetza* includes topological ITS-G5 routing algorithms like the Single Hop Broadcast (SHB) as well as geographic routing like GeoBroadcasts (GBC). Each ITS-G5 station in the network is represented by a *Vanetza* `Router`. Currently, only vehicles are used as ITS-G5 stations but Road Side Units (RSU) are on the roadmap already. Based on the chosen routing algorithm, the router determines the next hop in the network and sends the packet down to the physical layer, which is provided by either *Veins* or *INET*.

Applications are the reason why packets are created at first hand. The application layer is provided by *Artery*. The *Day One* applications mentioned by *ETSI* are already implemented and generate Cooperative Awareness (CA) and Decentralized Environmental Notification (DEN) Messages according to the standard's definition. For triggering DEN messages, the definition of proper trigger scenarios is needed. For this purpose the `Storyboard` can be used. It allows to define scenario conditions as well as effects to provoke various traffic scenarios like accidents, weather conditions or traffic jams. According to the traffic situation, the application triggers the appropriate DEN message.

Last but not least, there is the traffic simulation provided by *SUMO*. *SUMO* and *Artery* are connected using the *TraCI* interface. *TraCI* allows for reading information about the current state of each vehicle inside the simulation as well as changing vehicle parameters. Changing vehicle parameters is mainly used by the `Storyboard` to achieve above mentioned traffic situations. The maps used by `SUMO` can be derived from real maps with realistic traffic flows (like https://github.com/lcodeca/LuSTScenario/wiki or Open Street Map) or synthetic scenarios.

Installation

It is quite easy to install *Artery* on Linux based systems. You need to have a C++ Compiler with *C++11* support as well as *Boost* and *Vanetza* libraries for building *Artery*. Also, you need one of *Veins* and *INET*. Of course, *OMNeT++* and *SUMO* are also required.

Installing *OMNeT++*

1. Download the *OMNeT++ 5.0* archive (https://omnetpp.org/omnetpp).
2. Extract the archive into a new folder.
3. Type `./configure` and after this type `make`.
4. Add your `path/to/OMNeT/build` directory into your `PATH` environment variable.
5. Make sure your installation works by typing `omnetpp`. It is expected that the *OMNeT++* development environment starts.
6. Further installing instructions can be found at https://omnetpp.org/doc/omnetpp/InstallGuide.pdf.

Installing *SUMO*

1. For installing SUMO, first download the latest release (it can be found at http://www.sumo.dlr.de/userdoc/Downloads.html). It is not recommended to use the *SUMO* version shipped with Ubuntu or Debian system, because these versions are rather old.
2. Extract the downloaded files.
3. After this you have to build your SUMO version. Make sure that you have the packet **`libproj-dev`** (on Debian or Ubuntu) installed.

4. Navigate to the directory, where you have extracted SUMO.
5. Type: `./configure`. After the configuring was done, take a look at the `Optional features summary`, which was printed at the end. It has to include the entry **`PROJ`** in the`Enabled:` list. This feature needs the above mentioned `libproj-dev` and enables SUMO to provide geographic coordinates for each vehicle. **Without this feature, the simulation will stop at run-time because of invalid coordinates.**
6. Type: `make` to build SUMO in the `./build` directory.
7. Add `path/to/your/sumo-version/build` to your `PATH` environment variable.
8. Verify your installation by typing `sumo-gui`.
9. Further information on installing *SUMO* can be found at http://www.sumo.dlr.de/userdoc/Installing.html

---

Installing *Artery*

To install *Artery*, you need to have *Vanetza*, *Veins* and *Inet*. For the ease of installing, the *Artery* repository contains all these frameworks as sub-repository links. This ensures, that only compatible versions of *INET, *Veins*, and *Vanetza* are used with *Artery*.

1. Pull the Artery repository from https://github.com/riebl/artery
2. To build *Vanetza*, make sure you are in the root directory and type `make vanetza`. *Vanetza* will now be built in `extern/vanetza/build`
3. To build *Inet*, make sure you are in the root directory of *Artery* and type `make inet`. *Inet* will now be built.
4. To build *Veins* make sure you are in the root directory of *Artery* and type `make veins`. *Veins* will now be built in `extern/veins/build`
5. To build *Artery*, make sure you are in the root directory again. Type `mkdir build`.
6. Type `cd build`
7. Type `cmake ..`
8. Type `cmake --build`

Further information about building the above mentioned tools can be found at the following sources. This may be useful in case of an error.

1. *Vanetza*: https://github.com/riebl/vanetza
2. *Artery*: https://github.com/riebl/artery
3. *Inet*: https://github.com/riebl/artery/blob/master/extern/inet/INSTALL
4. *Veins*: http://veins.car2x.org/

Congrats, now you have all prerequisites to move forward and try to launch the first *Artery* example. How this is done is explained in the next section!

Running an Example

As *Artery* is quite complex, it is recommended to start by understanding *OMNeT++*. A good point to start is the ***TicToc*** tutorial. It can be found at https://omnetpp.org/doc/omnetpp/tictoc-tutorial/. This tutorial provides an overview on the basic functionality of *OMNeT++*. This includes, among others, the *NED* language and the definition of *Networks*.

If you already came in touch with *OMNeT++* you can try to start the example shipped with *Artery*. To do so, follow these steps:

1. Navigate to *Artery's* `build` directory (which was created while building)
2. Type `make run_example`. This command checks for code changes (like `make` usually does) and then launches *OMNeT++* using the `omnetpp.ini` in the `scenarios/artery` folder.
3. If you want `SUMO` to open its GUI while simulating, add this line to your `omnetpp.ini`: `*.traci.launcher.sumo = "sumo-gui"`

Running an example in debug mode

If you want to investigate the code while your simulation is running, you have to build *Artery* in debug mode. To do so it's recommended to use `ccmake`.

1. Navigate to your `build` directory of *Artery*
2. Type `ccmake .`
3. Move to entry `CMAKE_BUILD_TYPE` (usually this entry is at the top first position) press [enter] and type: `Debug`. Than again, press [enter].
4. Press [c] to configure.
5. Press [q] to quit.
6. Again, type `make` in the `build` directory.
7. To run the example in debug mode type `make debug_example`
8. After the *GDB* debugger is ready, type `run`
9. Feel free to press [ctrl + c] to step into the debugger and set your breakpoints

Adding my own ITS-G5 Service: Where should I start?

To start with creating your own service, take a look at the base class of each service, which is called `ItsG5Service`. Also you can look at the `CamService` and the `DenmService` as they are already implemented services. All application related files and classes can be found in the `artery/application` subfolder.

The initialization of a service is done by the `ItsG5Middleware`. Before adding your service, provide a proper C++ class and a *.ned* file belonging to your class. Your service class must be derived from `ItsG5Service`. Than, put your service in `examples/yourExample/services.xml`. As a starting point, you can copy `CamService` entry (`examples/artery/services.xml`) and change the port number.

The `ItsG5Middleware` also invokes the services each simulation step by calling `ItsG5Service::trigger()`. This means, if you want to do something periodically with your service, override this method and put your code in there.

Read Simulation of VANETs using OMNeT++ and Artery online: https://riptutorial.com/omnetplusplus/topic/9675/simulation-of--vanets-using-omnetplusplus-and-artery

Credits

S. No	Chapters	Contributors
1	Getting started with omnet++	Community
2	Simulation of VANETs using OMNeT++ and Artery	Ventu