System Level Simulation for Real-time Networks

Getting Started

Version 1.0

Franz Korf, Till Steinbach, Stefan Buschmann, Philipp Meyer
HAW Hamburg, Dept. Informatik
Berliner Tor 7, D–20099 Hamburg
Tel.: +49 40 42875 – 8406
eMail: {Franz.Korf,Till.Steinbach,Stefan.Buschmann,Philipp.Meyer}@haw-hamburg.de

May 25, 2016

Contents

1 Introduction

We developed a set of simulation models and tools to simplify the analysis of real-time networks using Ethernet based protocols as well as Fieldbusses (CAN, FlexRay). These models and tools are published in various open source projects. This manual gives you an overview over their contents and the dependencies between the different projects. It should help you getting started simulating your real-time networks.

Our toolchain consists of the following parts:

1.1 CoRE4INET

CoRE4INET is an extension to the INET-Framework for the event-based simulation of real-time Ethernet in the OMNEST/OMNeT++ simulation system. Currently CoRE4INET supports:

- TTEthernet (AS6802)
- IEEE 802.1 Audio Video Bridging (AVB) / Time-Sensitive Networking (TSN)
- IEEE 802.1Q / IEEE P802.1p VLANs and Priorities

1.2 FiCo4OMNeT

FiCo4OMNeT is an open source simulation model for the event-based simulation of fieldbus technologies in the OMNEST/OMNeT++ simulation system. Currently FiCo4OMNeT supports:

- CAN
- FlexRay

1.3 SignalsAndGateways

SignalsAndGateways is an open source simulation model for the event-based simulation of fieldbus technologies in the OMNEST/OMNeT++ simulation system. SignalsAndGateways using CoRE4INET and FiCo4OMNeT to enable a heterogeneous network simulation. It includes gateway components to connect Ethernet and Bus communication. Currently it supports:

- Gateway-Bus-Protokolls: CAN
- Gateway-Ethernet-Protokolls: Best Efford Crosstraffic, IEEE 802.1Q, AS6802

1.4 Abstract Network Description Language (ANDL)

The ANDL is a domain specific language that supports the easy generation of simulation configuration files for the frameworks CoRE4INET, FiCo4OMNeT and SignalsAndGateways. Because it is a plugin for the Eclipse IDE there are benefits like autocompletion, syntax highlighting, validation, autoformatting, renaming and scoping.

1.5 Overview

See figure 1 for an overview over the components and their interaction.

2 Preparations and Requirements

In general you can run the simulations on any popular platform, e.g. Windows, Linux or OS X. For our simulation models and tools to run there are certain requirements:

- You need a running OMNeT++(Academic License) or OMNEST(Commercial License) installation. See `http://omnetpp.org` or `http://omnest.com` for details.
- For the analysis tools you need a python environment. Python comes with most operating systems

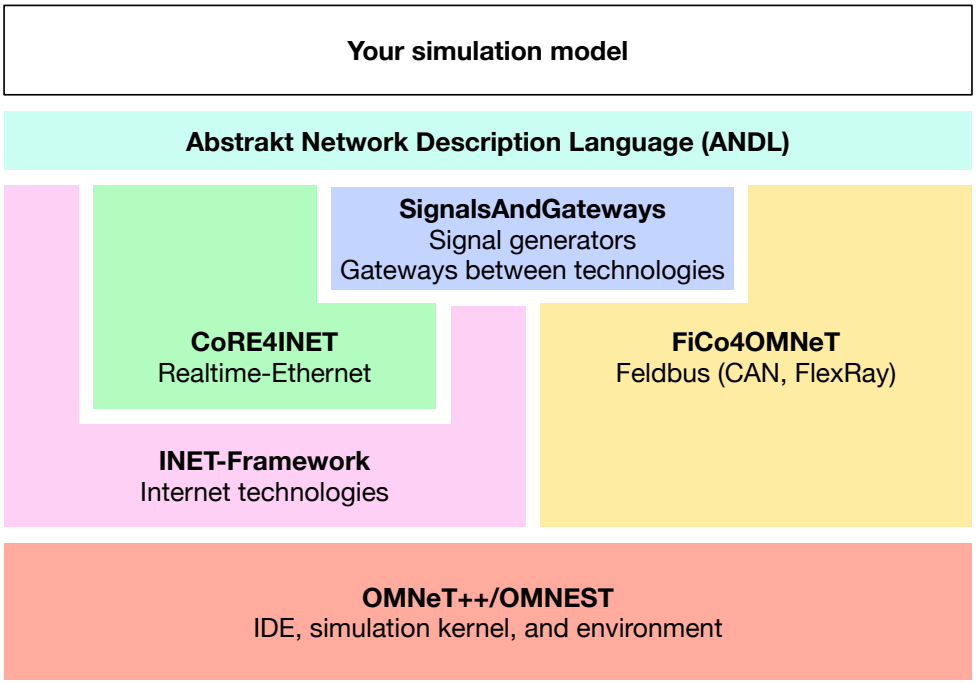

Figure 1: Overview over the components

3 Installation

3.1 OMNeT++

Download the latest version of OMNeT++/OMNEST from the website (e.g. `http://omnetpp.org` and unpack it. Afterwards you have to build the binaries:

3.1.1 Windows

- OMNEST:
  Execute the installation-wizard exe.
- OMNeT++:
  Extract the archive and open the mingwenv.cmd shell.

  Execute the following commands:

```
1  ./configure
2  make
```

3.1.2 Linux & Mac OS X

Open a shell and change into the folder in which OMNeT++/OMNEST was unpacked.
Execute the following commands:

```
1  ./configure
2  make
```

More information on the install procedure can be found in the doc/InstallGuide.pdf file of OMNeT++/OMNEST.

3.2 Simulation Models

3.2.1 Guided Installation

INET After the installation of OMNeT++ you will find a menu option "Help -> Install Simulation Models...". The wizard will guide you through the installation of the INET simulation model.

CoRE4INET, FiCo4OMNeT & SignalsAndGateways The easiest way to install the simulation models developed by the CoRE Group is the *CoRE Simulation Model Installer*. It is an OMNeT++ Plugin. Install the Plugin in your OMNeT++ IDE:

- Open your IDE (e.g. OMNeT++/OMNEST) and select: "Help -> Install New Software"
- Press the Button "Add..."
- Insert a name for the new repository in "Name", e.g. "CoRE"
- Insert the URL of our update page in "Location": `https://sim.core-rg.de/updates/`
- Select your new repository in "Work with...", this will take a while (showing "pending"). Be patient!
- Select the feature "CoRE Simulation Model Installer" and press "Next"
- The wizard will guide you through the installation...

After the installation you will find a new Menu option "Help -> Install CoRE Simulation Models...". The wizard will guide you through the installation of the simulation models. Please note that you also require the INET-Framework (see 3.2.1 on the preceding page).

3.2.2 Manual Installation

INET Start by installing the INET. Download the source from `https://inet.omnetpp.org/Download.html`. Extract the archive and then import the project into the OMNeT++/OMNEST IDE. Detailed information on how to install the INET framework can also be found in the inet folders INSTALL file.

CoRE4INET Now install the CoRE4INET models. Download the source from `http://sim.core-rg.de/trac/wiki/CoRE4INET_Download`. You can decide to use a stable release, a nightly version, or even directly clone our git-repository. Detailed information on how to install CoRE4INET can also be found in the CoRE4INET folders INSTALL file.

FiCo4OMNeT If your network design will contain fieldbusses you also have to install FiCo4OMNeT: Download the source from `http://sim.core-rg.de/trac/wiki/FiCo4OMNeT_Download`. You can decide to use a stable release, a nightly version, or even directly clone our git-repository. Detailed information on how to install FiCo4OMNeT can also be found in the FiCo4OMNeT folders INSTALL file.

SignalsAndGateways If your network design requires gateways between fieldbusses and/or real-time Ethernet, also install the SignalsAndGateways package. Download the source from `http://sim.core-rg.de/trac/wiki/SignalsAndGateways_Download`. You can decide to use a stable release, a nightly version, or even directly clone our git-repository. Detailed information on how to install SignalsAndGateways can also be found in the SignalsAndGateways folders INSTALL file.

3.3 Abstract Network Description Language (ANDL)

If you want to use the Abstract Network Description Language (ANDL) to generate your simulation configuration (see section 5) you have to install the ANDL-IDE plugin:

- Open your IDE (e.g. OMNeT++/OMNEST) and select: "Help -> Install New Software"
- Press the Button "Add..."
- Insert a name for the new repository in "Name", e.g. "CoRE"
- Insert the URL of the updatesite in "Location": "`http://sim.core-rg.de/updates/`"
- Select your new repository in "Work with...", this will take a while (showing "pending"). Be patient!
- Select the feature "Abstract Network Description Language" and press "Next"

- The wizard will guide you through the installation...

For nightly versions of the ANDL plugin also add the `http://sim.core-rg.de/updates/andl-nightly` updatesite.

4 First Steps

You can start by simulating the examples in the examples folders of the simulation models.

- Open the folder that contains the example in the IDE.
- Right-click on the omnetpp.ini file and select "Run As -> OMNeT++ Simulation".
- A default launch configuration will be generated and started. This is usually the simulation with a graphical user interface.
- You can change the launch settings using "Run -> Run Configurations". If you can to disable the graphical user interface select Cmdenv for example.

5 Generating Networks

5.1 Basics

- **Create new project:**
  Create a new Project (File->New->Project). Choose a name.
- **Create andl file:**
  Right click on the src folder and select new->other->General->File. The filename must end with ".andl". When Eclipse asks to add the Xtext Nature to the project select yes.
- **Configure network:**
  The network configuration takes place in the empty andl file. Autocompletion (Ctrl+Space) is a great tool to support the process. It shows all possibilities the can be placed in the editing area.
- **Generate simulation files:**
  If the Process did not start automatically after saving the build process must be started (Ctrl+B) (right click on .andl file->Generate Network). After the build process is finished the simulation files can be found in the simulations folder of the project.
- **Autoformatting:**
  To initate the Autoformatting press Ctrl+Shift+F.

5.2 Language

The language contains three main components. They are types, settings and network. The first two are optional. The last one is the mandatory network description.

5.2.1 types

Major elements of the network can instantiate a type to reduce the parameter redundancy in the network description part. This major elements are ethernetLink, canLink, node, switch, gateway and message. It is possible to set any number of types sections. Types can be extended by other types. There are two types sections in the following example. The types names are std and sntypes. The message MSG type in sntypes extends the message MAX_CANMSG in std.

```
1 types std {
2     ethernetLink ETH_100MBIT {
3         bandwidth 100Mb/s;
4     }
5
6     canLink CANBUS_500KBIT {
```

```
7           bandwidth 500Kb/s;
8       }
9
10      switch ETH_SWITCH {
11          hardwareDelay 8us;
12      }
13
14      message MAX_CANMSG {
15          payload 8B;
16      }
17  }
18
19  types sntypes{
20      message MSG extends std.MAX_CANMSG{
21          sender canNode1;
22          receivers canNode2;
23      }
24  }
```

5.2.2 settings

The optional settings part configures simulation file generation related parameters. For example the behavior of the automatic time-triggered scheduler. The following code shows how to deactivate the scheduling of time-triggered messages for the simulation configuration.

```
1  settings{
2      tteScheduling false;
3  }
```

5.2.3 network

This is the mandatory part of the ANDL. It contains of 3 mandatory subsections. They are devices, connections and communication. The first one is used to describe the network participants. The second one sets the connection between them. In the last section all messages are defined that should traverse the network. The next example shows the start of the network component. The network name is specified as smallNetwork.

```
1  network smallNetwork {
```

- **devices:**
  In the devices section network participants are configured. The syntax is like in the types component. On device can extend an device type. In the following example are 2 canLink(canbus1, canbus2),1 ethernetLink(ethcable1), 1 switch(switch1), 2 gateways(gateway1, gateway2) and 2 nodes(canNode1, canNode2) defined. The semicolons can be replaced with an "{}" block for the definition of additional parameters.

```
1  devices {
2      canLink canbus1 extends std.CANBUS_500KBIT;
3      canLink canbus2 extends std.CANBUS_500KBIT;
4
5      ethernetLink ethcable1 extends std.ETH_100MBIT;
6
7      switch switch1 extends std.ETH_SWITCH;
8      gateway gateway1;
9      gateway gateway2;
10     node canNode1;
11     node canNode2;
12 }
```

- **connections:**
  The elements in the devices section can be used now to build the network infrastructure. Each connection is a member of a segment. Segments are later be used to define the different message representations. For Ethernet connections there are two network participants connected via one ethernetLink. For CAN connections one participant plugged to one canLink. The connections example shows to segements(backbone and canbus) for the individual connections. A special feature for ethernetLink is the "new" keyword. It enables the possibility to instance a new ethernetLink with a defined type. This reduces the configuration effort instead of defining each cable in the devices section.

```
1  connections{
2      segment backbone {
3          gateway1 <--> ethcable1 <--> switch1;
4          gateway2 <--> {new std.ETH_100MBIT} <--> switch1;
5      }
6      segment canbus {
7          gateway1 <--> canbus1;
8          canNode1 <--> canbus1;
9          gateway2 <--> canbus2;
10         canNode2 <--> canbus2;
11     }
12 }
```

- **communication:**
  Communication is defined over messages. Each message can enxtend a message type and has parameters like the devices. Additionaly a message has a mapping section. In this section the represantation in different parts of the networks is defined. There is no neccassary order of the mapping entires. At the moment parts are segements and gateways. In the following example are three messages(canmsg1, canmsg2 and canmsg3). All parameterize an period and extends a parameterized message type. For canmsg1 the mapping says that the message is represented thrue the CAN id 1 in the canbus segment. In the backbone segment the message is represented by a best effort ethernet frame. Additionally the involved gateway are gateway1 and gateway2.

```
1  communication{
2      message canmsg1 extends sntypes.MSG{
3          period 10ms;
4          mapping {
5              canbus : can{id 1;};
6              gateway1;
7              backbone : be;
8              gateway2;
9          }
10     }
11
12     message canmsg2 extends sntypes.MSG{
13         period 15ms;
14         mapping {
15             canbus : can{id 2;};
16             gateway1;
17             backbone : be;
18             gateway2;
19         }
20     }
21
22     message canmsg3 extends sntypes.MSG{
23         period 20ms;
24         mapping {
25             canbus : can{id 3;};
26             gateway1;
27             backbone : be;
```

```
28                  gateway2;
29              }
30          }
31      }
```

At last the network section must be closed and it is ready for the generation process.

```
1   }
```

6 Analyzing Results

OMNeT++/OMNEST features a build in analysis tool. You find your simulation results after the simulation run (press finish) in the results folder of your simulation. *.vec* files are the recordings of vectors, *.sca* files are the recorded scalars. If enabled you also find the eventlog (*.elog*). For result browsing you can generate a result analysis file (*.anf*) by double clicking a .vec or .sca file. In the tabs below the editor change to "Browse Data". You can browse through all recorded metrics and by double clicking generate simple plots. For more sophisticated plots you can use the Datasets tab. More information about result analysis can be found in the OMNeT++ manual in chapter *12.4 The Analysis Tool in the Simulation IDE*

7 Next Steps

After your first steps you probably want to dive deeper into the simulation models. You can find further information in the documentations of the Simulation Models

- CoRE4INET Documentation:
  `http://core4inet.core-rg.de/trac/wiki/CoRE4INET_Documentation`
- FICo4OMNeT Documentation:
  `http://core4inet.core-rg.de/trac/wiki/FiCo4OMNeT_Documentation`
- SignalsAndGateways Documentation:
  `http://core4inet.core-rg.de/trac/wiki/SignalsAndGateways_Documentation`

The Documentation has two parts. In the "NED documentation", the Simulation modules and parameters are explained. You need the NED documentation to find suitable parameters for your configuration. In the sourcecode (doxygen) documentation the source is documented. This is useful if you want to extend the available models.