Simulation and emulation of realtime communication networks

Master Thesis

Franz Profelt

22.06.2016

General

Goal

Simulation of a real-time communication network

- Open Source simulation framework OMNeT++
- Real-time communication network POWERLINK
- Open Source implementation openPOWERLINK

Motivation

- Increasing requirements for embedded systems result in urge for testing
- Flexible testing scenarios realized by simulations

OMNeT++ Framework

General

- Object oriented modular discrete event network simulation
- Open Source simulation framework written in C++
- Discrete event simulation, with flexible simulation core components
- Built-in functionalities for real-time simulation, parallel simulation and emulation

Modelling components

- Different modules connected via channels
- Transmission of messages/packets

Design Evaluation

Different design for a simulation of an existing application without changing the existing structure and functionality.

Fundamental Designs

Monolithic Small number of modules representing complex functional units

Modular High number of modules representing simpler functional unit

Performance Measurement

- runtime
- created events
- real-time behavior

Results

Sequential

- Improved performance of monolithic design over a modular design, in average 3.044

Parallel

- Only runtime measurement applicable
- Improved performance of monolithic design over a modular design, in average 2.067
- Overhead by necessary communication for simulation time synchronization between logical partitions

openPOWERLINK Stack

General

- Open Source implementation of POWERLINK.
  - Real-time communication protocol
  - Transmission of synchronous and asynchronous data
- Distributed under the BSD license, available on GitHub and Sourceforge
- Introduction in POWERLINK

Structure

- Separation in kernel and user layer
- Various platforms/targets (Linux, Windows, Altera Nios II, Xilinx Microblaze, ...)

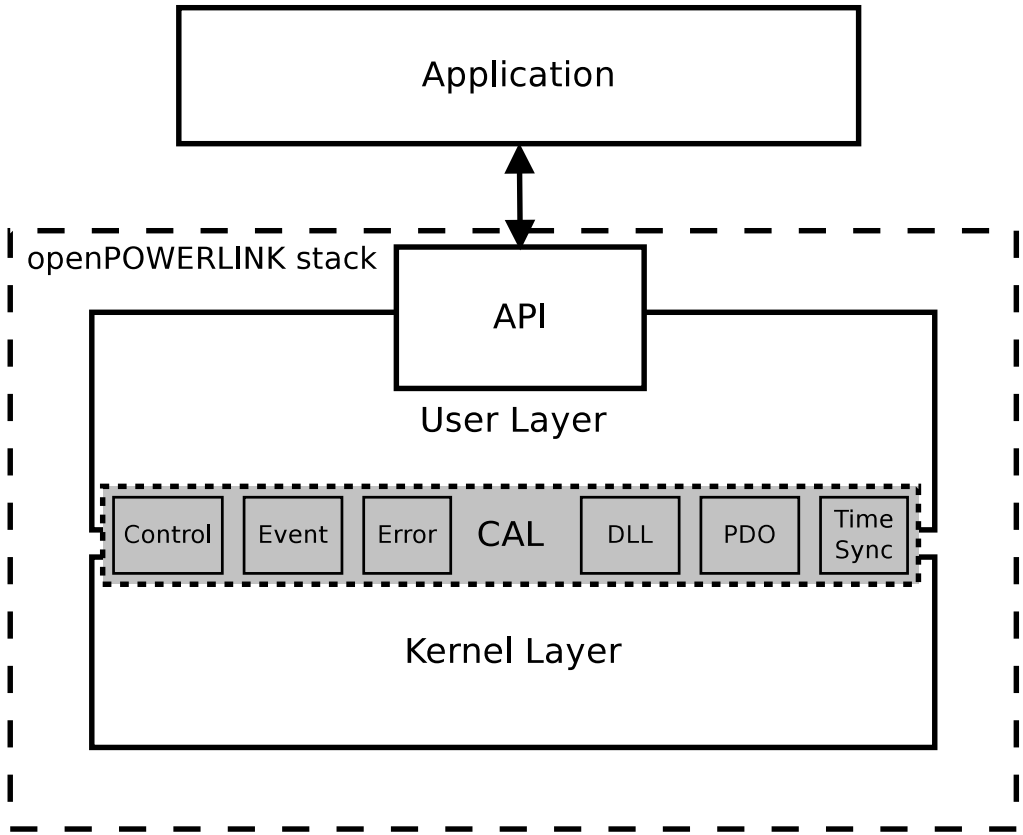
Structure

Architecture

User layer High level functionalities, API, Asynchronous transmission

Kernel layer Time critical functionalities, synchronization, drivers

CAL Connection between user layer and kernel Layer



Platform Dependency

Realized via common header files and platform specific implementations.

Implemented modules for minimal dependency

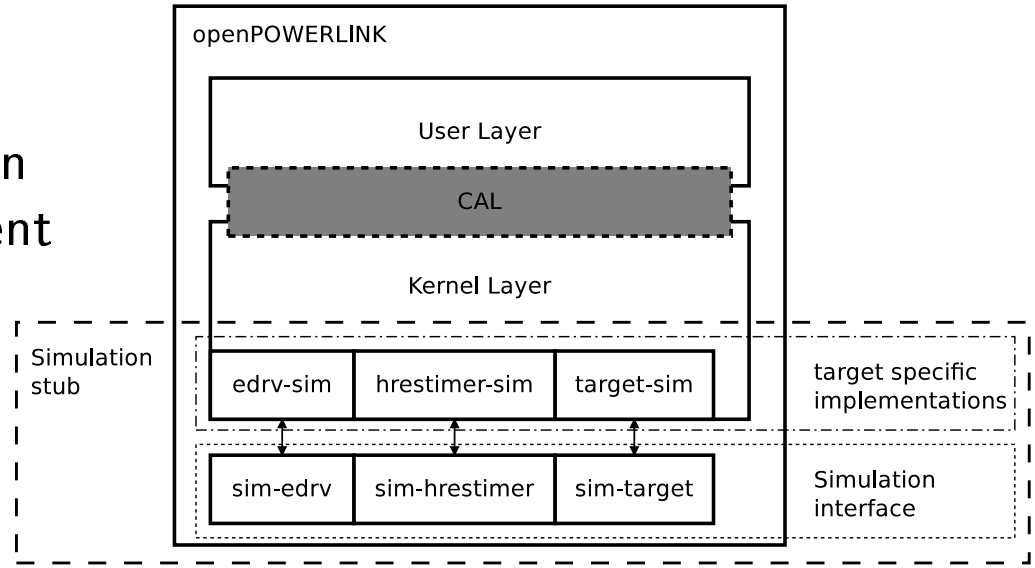
target	General target specific functionalities (Led, IP Address, Default Gateway, Tickcount, sleep)
edrv	Ethernet driver
hrestimer	High resolution timer
sdoudp	Asynchronous data transmission via UDP
trace	Trace output for debugging

Simulation

Separation of simulation and openPOWERLINK stack.

Simulation Stub

- Target-specific implementations
  - Implementation of platform dependent modules for *sim* target
  - Connection to simulation interface
- Simulation interface
  - Connection to simulation environment (independent of OMNeT++)
  - Calling stored function pointer with instance handle



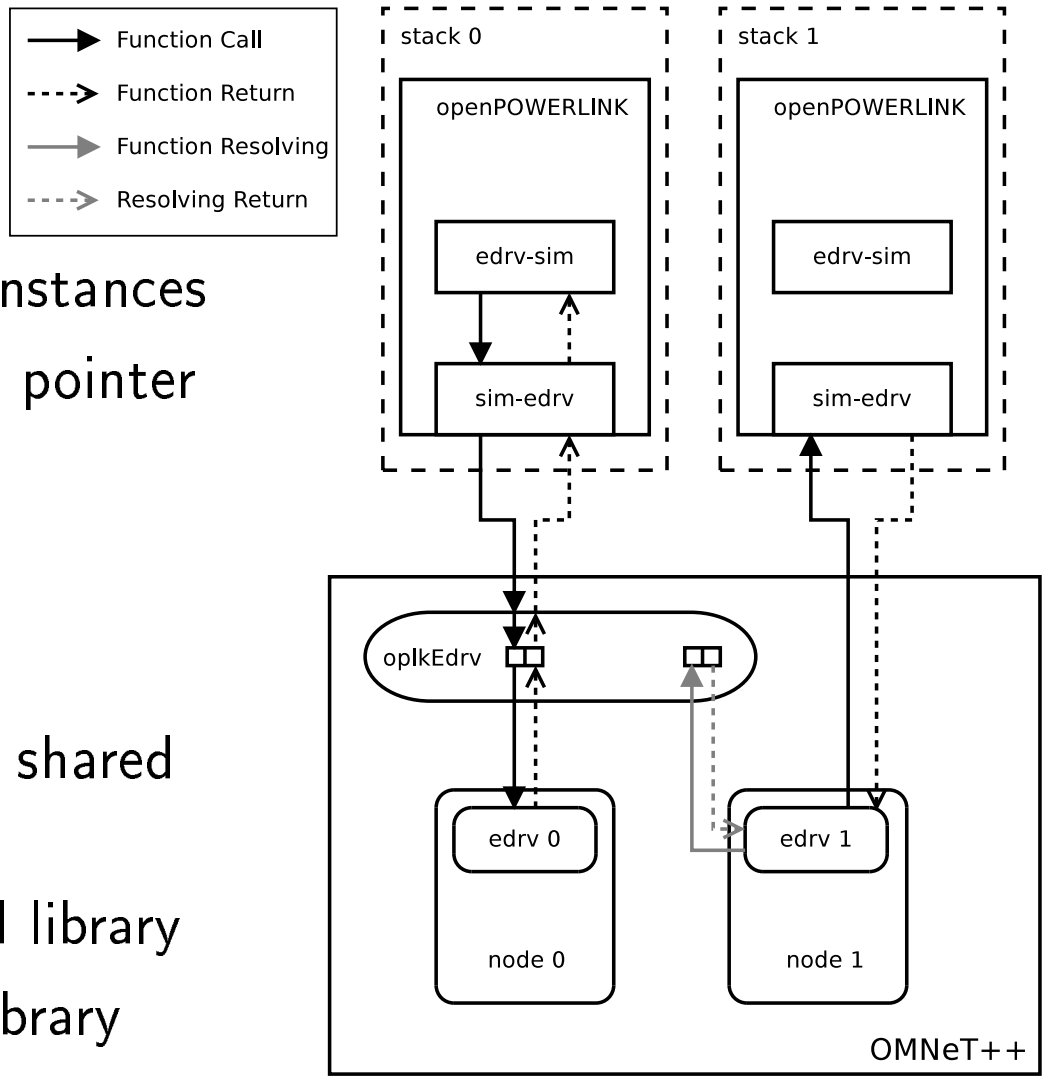
Simulation Hierarchy

Stack interface

- Connection to stack
- Handling of multiple stack instances
- Static methods for function pointer

Multiple instances

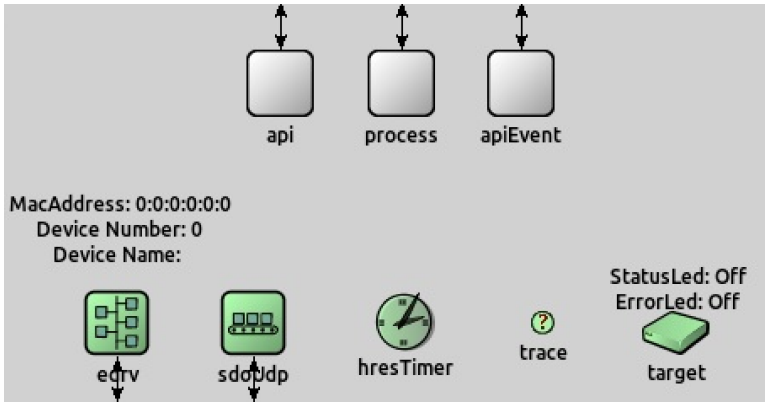
- Usage of static variables
- openPOWERLINK stack as shared library
- Multiple instances of shared library
- Different copies of shared library



Simulation modules

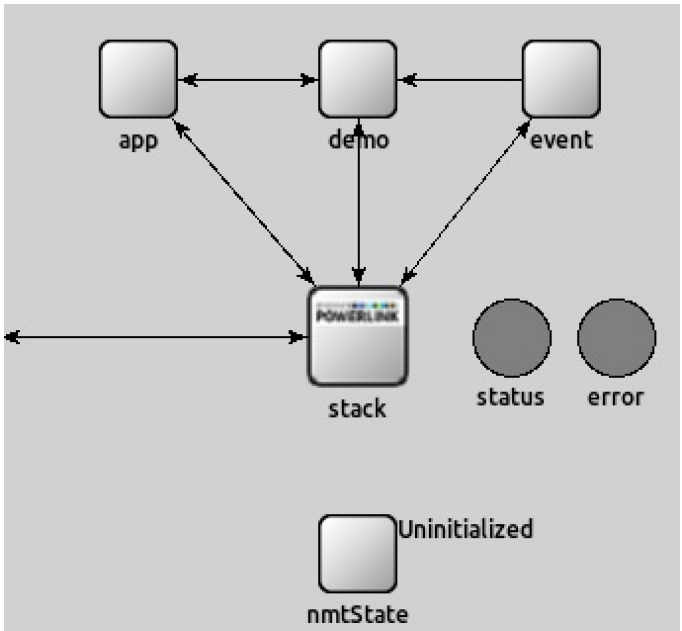
Stack module

- Structure of simulated openPOWERLINK stack



Generic node

- Basic functionalities for each node (MN, CN)
- Message handling in between modules
- Base classes for specific implementations



Further Development

Enhancements

- Implementation of different modularities
- Implementation of multiple demo networks/applications
- Integration of standardized network interfaces (INET library)

Publication

- Integration of the simulation stub within the openPOWERLINK stack 2.5.0
- Hosting on GitHub `https://github.com/OpenAutomationTechnologies/openPOWERLINK_omnetpp`