Feasibility Analysis for Networked Control Systems by Simulation in Modelica

Liu Liu and Georg Frey
University of Kaiserslautern
Department of Electrical and Computer Engineering
Erwin-Schrödinger-Str.12
D-67663, Kaiserslautern, Germany
{liuliu | frey}@eit.uni-kl.de

Abstract

Technological advancements in communication and embedded computing lead to the increasing application of decentralized structures in automation and control. The decentralization adds new behavior in the form of – often non-deterministic – delays to the resulting networked control systems (NCS). To analyze a closed loop control system including network communication, simulation is a viable approach. In this work a library for the simulation of controller and network hardware to be used in combination with process models in closed loop is presented. The library is implemented in the Modelica language. To allow easy application a structure conserving component based modeling approach is followed. The applicability is shown by analyzing the performance of a robotics control system where communication is realized alternatively with Ethernet and CAN bus.

1. Introduction

The increasing utilization of networked technologies and embedded devices leads to distributed architectures in control and automation systems. The distribution takes place in two aspects, namely, the spatial allocation of devices and the spreading of control algorithms. On the one hand it allows the use of cost-effective embedded devices and on the other hand it requires communication and synchronization between devices and software tasks in a multitasking real-time kernel. Consequently, the complexity of the overall system poses new challenges to the analysis. For this reason, computer-based tools for the analysis of the functional and temporal behavior of the system are required.

To study closed control loops over networks, specialized tools covering individual aspects of a system – e.g. algorithm analysis, network analysis and process simulation – are not sufficient. The interactions between communication, computation and the controlled plant from both continuous and discrete domains influence the closed loop behavior. Obviously, a tool which covers all the above mentioned aspects is necessary.

The most direct way to analyze a complex system is using computer simulation. Since the controlled processes may contain parts from various engineering domains (electrical, chemical, mechanical, …), a general multi-domain simulation environment can satisfy the different needs to the greatest extent. Hence, Modelica, characterized by its multi-domain modeling capability and the object oriented modeling paradigm, offers a proper basis for the overall system analysis.

The paper presents an approach for the integrated simulation of communication and controller hardware in Modelica. Together with the process models, the complete control system in a closed loop can be analyzed from both temporal and functional perspectives. For the temporal behavior analysis, it is able to reveal the detailed control timing characteristics including periodic/aperiodic sampling and actuation, varying executing times of algorithms, blocking and waiting times of tasks due to scheduling, communication delays, etc. While from the functional behavior analysis, a more direct viewing of control performance can be shown. Thus feasibility studies can be performed in the early development phase and meanwhile the redesign or optimization of the control system can be achieved based on the study of detailed control timing characteristics. This paper focuses on the description of the developed Modelica library for the simulation of networked control systems and its possible application areas. Chapter 2 introduces the composition of the library and its functionality as well as its features. The developed library is implemented and tested under Dymola 6.1. In chapter 3, an application example is given to demonstrate the use of the library. Finally, conclusions are drawn and an outlook on future work on the library is given in chapter 4.

2. Modelica NC-Library

2.1. Blueprint of the library

Based on the identification of the networked control system, the blueprint of the developed Modelica library is drawn (Figure 1). The aim is to cover the most important effects of the system and meanwhile to offer maximal usability. In consequence, a structure conserving
modeling paradigm is utilized. The real world components are mapped to explicitly defined models. The most important advantage of this approach is the direct representation of the implemented system structure.

![Figure 1. Blueprint of the developed library.](image)

Basically, the components shown in the blueprint and their derived models can be organized in three domains.

The first domain is the process under control. In most cases, users from different engineering fields have dissimilar processes and demands. This part of task is managed by the standard Modelica library. Thanks to its multi-engineering modeling capabilities, process models from many fields such as mechanical, electrical, thermodynamic and etc. can be well and easily established. An example of building a robotic arm model using the Modelica MultiBody library is given in the next chapter.

The second domain (light shading) covers the components related to data handling and process interfacing. These components can be regarded as smart embedded devices. A full-function embedded device is usually equipped with a computation kernel (CPU), process interfaces (Sampling and Actuation) to couple with the controlled process, and network interfaces (Transceiver) to communicate with other devices. In some cases, the controllers in the higher automation level (Industrial PC or SCADA) do not have direct access to controlled process, thus the information acquisition is then accomplished by an I/O board. In contrast to the normal controller, the I/O board may have limited computation power and focuses on the interfacing between different automation layers and networks.

The third domain (dark shading) covers network communication. A network consists of one or more media. In case different media are used, intermediate systems are necessary to bridge them. The transceiver manages the communication protocols and interfaces the information processing and transmission.

There are three types of information exchange between components in Figure 1. The dotted lines describe continuous physical values. The dashed lines depict digitalized data interconnections between devices. Finally, the encapsulated data in the network frame is shown as solid lines.

### 2.2. Overview of the library

Based on the classification of components from chapter 2.1, a Modelica Network-Controller-Library has been implemented. The library consists of two sub-libraries covering the network and controller domains respectively. Together with the process models built by the standard Modelica library, the entire networked control system in a closed loop can be set up and analyzed.

To handle the three types of information interconnections, the original Modelica language offers well defined interface class to manage the physical values. But for the information systems such as controller and network devices where the amount, format and length of information may vary, it is not possible to define a general interface in the Modelica language directly. This gap is filled in the developed library by using an external information exchange system defined in C language. This system offers services like a data base. The data to be exchanged is labeled by unique identifiers (for more details see [1]). The models of the library can access the services by using external function interface of Modelica. In this way the complexity of data handling is hidden from the Modelica models and meanwhile all types of information interconnection within the system can be realized.

### 2.3. Network sub-library

The network library consists of fundamental models derived from the area of network communication. Generally, they are categorized into three basic types, namely the communication media, intermediate systems and transceivers. For different transmission protocols, they may vary in form and functions. Thus for a better usability, the components are arranged in packets according to protocols (Figure 2).

![Figure 2. Network sub-library.](image)
Media Access Control mechanism to model the non-deterministic behavior of network access. While on the physical layer, the data-length-dependent transmission duration is simulated. The modeling of these two layers offers a basis of transmission schema, thus higher layer protocols such as TCP or UDP can be realized by programming the controller model accordingly.

Thanks to the structure conserving modeling, the network is not considered as a single component (like for example in TrueTime [2]) but all the basic components are explicitly modeled. The main advantage is that the network topology, which can have significant influences of the performance, is visible.

The library benefits from the object oriented paradigm of Modelica language. For instance, intensive using of the inheritance concept allows a variety of models based on a small number of basic classes. Furthermore, along with the Modelica keyword `replaceable`, it is easy to switch between derived models from the same parent class on the GUI layer of Dymola. An example for this is the switch from WLAN to ZigBee since both models are derived from a single class.

2.4. Controller sub-library

The controller sub-library Figure 3 contains models to describe the behavior of embedded controller devices. The components are arranged into sub-libraries according to their functionalities.

![Controller sub-library](image)

The board sub-library describes the host devices which structurally represent the embedded devices. A generic board model contains a computation kernel from CPU library to run certain algorithms, process interfaces for sampling and actuation and network interface supplied by the network library to interchange information through network. Furthermore, a board model hosts memory models to enable the internal information exchange between above mentioned components.

The CPU library contains detailed controller models to represent real timing behavior. They are featured by elaborated models of software tasks that compete for the processing time and scheduler models for the task arbitration. In opposite to the simulations of control systems without those timing characters, the effects of preemption, blocking and waiting of software tasks are considered in retrieving system behavior which provides more realistic simulation results.

To include control algorithm there are two options. For simple algorithms the direct implementation using the Modelica language is proposed. More complex algorithms or already existing implementations in other programming languages can be included via an external function interface. Due to the predefined internal structure and interconnections in the CPU model, the complexity of algorithm integration is minimized. For the first case, using the inheritance mechanism of Modelica, one has to derive a new class from the predefined basic class *Task* and specify the task relevant parameters like execution time, sleep time, etc. The final step is to insert the code segment into the derived model. For the second case, the parameterization is left to the definition of an algorithm in another programming language. In the Modelica model only the designation to the external algorithm has to be given.

3. Application example

The various timing effects due to the distributed nature of the system can lead to significant performance degradation of a closed loop control system. Thus the constraints of hardware implementation have to be taken into consideration in the early system design phase. The simulation of a networked control system in closed loop enables a direct assessment of quality criteria and by this allows feasibility analysis. The following example reveals how the hardware implementations influence the control quality (see [3] for an example of open loop response time analysis and [4] for a comparison of the simulation approach to formal validation).

3.1. Trajectory control of a robotic arm

Figure 4 shows the experiment setup of a distributed control system with a robotic arm using fully-switched-Ethernet. In Figure 5 the same system is implemented using CAN bus for communication.

![Robotic arm experiment using fully-switched-Ethernet](image)
Modeling of the mechanical structure of the robotic arm is done using the Modelica Multibody library. The irregular forms of links are composed by components in regular forms such as cylinder and box. The robotic arm consists of five revolute joints connected to five drive axes. Each axis consists of an axis controller (C1…5), three sensors providing information about angle, angular velocity and torque, a DC motor and gearboxes (see the bottom-right in Figure 4). The sensor values and setting voltage are periodically (2.5 ms) sampled/updated by the AD and DA converters of the controller board.

The trajectory plan controller (C0) periodically (10 ms) sends request messages to all the axis controllers through the network. The axis controllers reply with the newest sampled sensor values. When all sensor values have been received by C0, it begins to compute the new torque references according to a defined control algorithm and sends them to C1…5 respectively which compute the setting voltage accordingly.

The control timings are illustrated in Figure 7. For the ideal controller, the delay between sampling and actuation is constant. The Ethernet solution causes additional transmission and processing delays, but due to the high transmission speed, the total delay stays within the sampling interval and thus only to slight performance degradation results. The CAN solution suffers from the slow network. In some situations, the assumption of equidistant sampling is violated. The two dashed arrows show such a situation. At the proposed sampling time, sending the actuation data is not yet finished, the sampling request has to wait in the sending queue until the bus is free. Hence, sampling and actuation is delayed. The overall delay exceeds the assumed sampling interval resulting in significant performance degradation.

4. Summary and outlook

To analyze closed-loop control systems the automation hardware including networks and controllers have to be taken into account. This paper presented a modeling and simulation approach for the analysis of NCS. The presented models follow a component based structure conserving approach and are implemented using the Modelica language. A robotics example showed how the models can be used to check the feasibility of different automation architectures for a given problem. Further work will concentrate on refinement of the models and the addition of further network types (LIN, Flexray). The library is available at http://www.eit.uni-kl.de/frey/.

References