

# A Simulation Architecture for Time-Triggered Transducer Networks

Martin Schlager<sup>1</sup>

<sup>1</sup>Institute for Computer Engineering,  
Technical University of Vienna, Vienna, Austria  
martin@vmars.tuwien.ac.at

**Abstract** — *Steadily growing microcontroller capabilities at an ever decreasing cost per operation encourage the development of more extensive and smarter applications in the domain of embedded systems. Intelligent systems, i. e., systems that react appropriately to changing situations without user input, provide an attractive solution in order to keep pace with the upcoming challenges of the embedded market. As the development of such intelligent embedded systems is a demanding and error-prone task, a great need for supporting methodologies evolves. The objective of this paper is to introduce an approach to build up a simulation architecture for emulating particular parts of a distributed embedded system. Well known advantages of simulation, like improved reproducibility of simulation runs, the possibility to simulate rare and maybe dangerous situations, and potential cost reduction during development will be supported by such an architecture.*

## 1 Introduction

Intelligent embedded systems, i. e., tightly integrated collections of signal processing modules or layers that are able to adapt to changes or new information, are employed for various reasons: intelligent embedded systems may lead to more efficiency, to greater autonomy, to cost savings or even to an increase in dependability by providing self-stability properties or by enabling graceful degradation.

Intelligent embedded systems, however, impose certain difficulties on the developer of such a system. Handling dependability and real-time constraints while minimizing cost, size, and power consumption, entails the need for new methods that assist in intelligent embedded systems development. Thus, a comprehensive development frameworks would be desirable.

This paper presents a simulation architecture for time-triggered field-bus networks – an approach to realize a major part of a development framework for embedded systems. Intelligent embedded systems development will be strongly supported by such a simulation architecture.

Developing a distributed intelligent embedded solution, like for example a neural network processing several sensor inputs to produce a joint result, is facilitated by the provision of simulation capabilities. Firstly, at early design stages it is easier to develop parts

of such an application while *simulating the missing parts*. Secondly, with simulation the whole system or parts of it can be *tested* under various conditions.

The remainder of the paper is structured as follows: Section 2 will investigate on certain definitions that are used throughout the rest of this paper. Section 3 will reveal several difficulties in embedded systems development and give an overview of the simulation architecture objectives. Section 4 will present a generic time-triggered simulation architecture. In section 5, tangible components to build up the proposed approach are introduced, whereas section 6 summarizes the ideas presented and further gives an outlook on future work.

## 2 Definitions

An *embedded system* is a computer system designed to perform a dedicated or narrow range of functions with a minimal user intervention. An *intelligent system* is a system that is able to react appropriately to changing situations without user input and hence, an attractive solution for an embedded system. [1]

Every embedded system that is used for control applications interacts with its environment (the *controlled object* [2]) by means of *transducers*. *Transducers* may generally be divided into two classes: sensors, which monitor a system; and actuators, which impose a condition on a system. Sensors and actuators are comprehensive classes of transducers, i.e., any transducer in operation is functioning at any given moment either as a sensor or as an actuator. [3]

According to [4] a *smart transducer (ST)* may comprise a hardware or software device consisting of a small, compact unit containing a sensor or actuator element (possibly both), a microcontroller, a communication controller and the associated software for signal conditioning, calibration, diagnostics, and communication.

Throughout this paper, a *distributed control system* is defined as a collection of autonomous smart transducer nodes linked by a fieldbus network. A *node* is a self contained computer with its own hardware (processor, memory, communication interface, interface to the controlled object) and software (application programs, operating system), which performs a set of well-defined functions within the distributed computer system. [2] Such a distributed control system must not only support predictable performance in the value domain but also in the temporal domain.

*Time-triggered systems* can guarantee the mandatory timeliness communication behavior. Communication activities in time-triggered systems only depend on the progression of time, i.e., communication is not effected by external events. Furthermore, due to the concept of a *temporal firewall* [5], i.e., an interface for the unidirectional exchange of state information over a time-triggered communication system, propagation of control signals is avoided.

## 3 Objectives

The development of applications for time-triggered distributed control systems imposes certain difficulties that include (but are not limited to):

**Achieving real-time behavior:** A distributed control system requires predictable behavior in the temporal domain. Thus, application tasks must finish execution within a

certain time span and every violation of this requirement may cause harmful incidents.

**Complexity through parallelism:** The nodes of a distributed control system work in parallel which increases the complexity of application development compared to sequential programming.

**Limited debugging support:** The increase of complexity introduced by parallel computing is often accompanied by weak debugging capabilities which makes application development even more difficult. Debugging procedures often influence the application runtime behavior and therefore invalidate the observed results. Furthermore, embedded system controllers typically suffer from a lack of monitoring capabilities, i.e., they are not connected to a display unit by default, thus, causing effort to be taken until monitoring is possible at all.

**Resource constraints:** Software engineering methods that require significant computational overhead are not applicable due to limited processing power and/or memory restrictions of embedded devices.

According to [6] simulation is one of the most widely used and accepted tools in operations research and systems analysis. Some of the well known advantages of simulation, like improved reproducibility of simulation runs, the possibility to simulate rare and maybe dangerous situations, and potential cost reduction during development are utilized by the simulation architecture.

Replacing several nodes of a distributed control system through a simulation is attractive since isolation of the corresponding non-simulated system reduces complexity and facilitates the debugging process. Furthermore, simulation of transducer behavior leads to improved reproducibility of system tests and the possibility to simulate rare and maybe dangerous situations.

Transducer simulation is closely related to cluster simulation<sup>1</sup> – a technique where a dedicated node simulates the behavior of multiple nodes of a distributed application. While the concept of cluster simulation is applicable to a broader range of simulation domains, transducer simulation merely emphasizes on the simulation of smart sensors and actuators connected to a fieldbus network.

The development process of distributed intelligent embedded systems, dedicated to be executed on several low-cost transducer nodes of a fieldbus network, will be significantly improved with the provision of simulation. For example, a neural network application, designed to filter several sensor values (e. g., pressure sensors) and to trigger a set of actuators (e. g., valves of a vat) could be implemented without presence of the controlled object, i. e., the vat. Furthermore, the behavior of the neural network algorithm can easily be tested in dangerous situations like for example a rapid increase of pressure. With simulation, the application can be implemented partially (e. g., only one of three sensors physically exists), parts of the distributed system can be tested in isolation and multiple executions of equal tests is possible.

Simulation in a time-triggered system is difficult, since simulation usually performs a compression in time or space. If the instant of a measurement is not known a priori,

---

<sup>1</sup>A comprehensive overview on cluster simulation can be found in [7].

the temporal accuracy of a simulated measurement may be insufficient and non-constant – a problem that has to be overcome by increased performance of the simulation host computer.

It is the objective of this paper to introduce a simulation architecture for emulating smart transducer nodes within a time-triggered control system. The enhancements through simulation will show significant improvements as regards design complexity and debugging capabilities while preserving real-time behavior.

## 4 Simulation Architecture Concepts

To simulate a transducer in a network (cluster) of transducers, a simulation node that runs the task of emulating the missing transducers has to be set up. In practice it is not sufficient to simulate only one transducer, so methods on simulating a set of different sensor or actuator units have to be found. The simulation node will have to behave in a way that the remaining components of the cluster do not notice a difference to a real network. Thus, the communication interface has to resemble the interface to a real transducer in terms of encoding, temporal behavior and control flow.

When emulating one or more transducers in a cluster, it is necessary to simulate the behavior of the real transducers and their effects on the controlled object, i. e., the environment. Thus, the simulation node will also have to model the part of the environment the simulated transducers are interacting with. Especially the dynamic modeling of the controlled object can be very complex and resource demanding.

Figure 1 depicts the key components of the proposed Simulation Architecture.

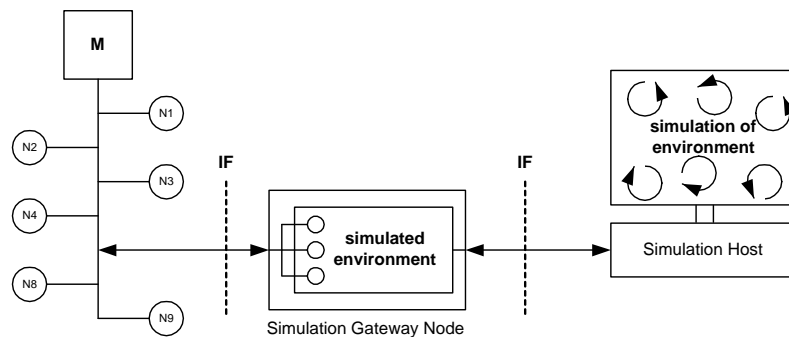


Figure 1: Simulation Architecture

To emulate transducer nodes of a *Smart Transducer Network* (left side of Figure 1) a distributed approach consisting of a *Simulation Gateway Node*, a *Simulation Host* and the respective *interfaces* is outlined. The Simulation Gateway Node is part of the smart transducer network. Its purpose is to timely provide the other transducer nodes with simulation data. The simulation data is created at the Simulation Host – a computer with sufficient processing power to calculate even complex simulation models online, i.e., during the simulation. The next sections will cover in-depth explanation of the four major Simulation Architecture parts.

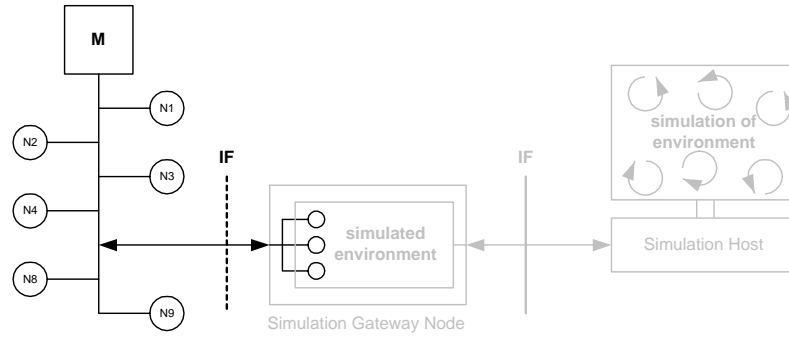


Figure 2: Smart Transducer Network

#### 4.1 Smart Transducer Network

The information transfer between a smart transducer and its client is achieved by sharing information that is contained in an internal interface file system (IFS), which is encapsulated in each smart transducer. The IFS provides a unique address scheme for transducer data, configuration data, self-describing information and internal state reports of a smart transducer [8].

A time-triggered sensor bus will perform a periodical time-triggered communication to copy data from the IFS to the fieldbus and write received data into the IFS. Thus, the IFS is the source and sink for all communication activities. Furthermore, the IFS acts as a temporal firewall [5] that decouples the local transducer application from the communication activities. A temporal firewall is a fully specified interface for the unidirectional exchange of state information between a sender/receiver over a time-triggered communication system. The basic data and control transfer of a temporal firewall interface is depicted in Figure 3, showing the data and control flow between a sender and a receiver. The IFS at the sender forms the output firewall of the sender and the IFS of the receiver forms the input firewall of the receiver.

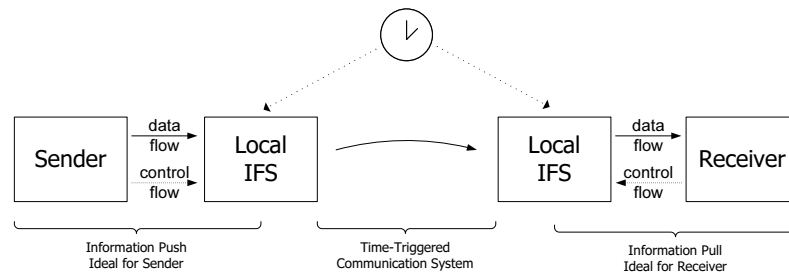


Figure 3: Temporal Firewall

The concept of a temporal firewall rules out propagation of control signals within a Smart Transducer Network. Besides, it facilitates the emulation of transducers by hiding temporal inconsistencies introduced by the simulation process, i. e., the simulator has to provide simulation values only within certain time-spans and not at determined points in time.

Thus, due to the concept of a temporal firewall the communication interface between the

Simulation Gateway Node and the real transducers – as depicted in Figure 2 – perfectly resembles the interface between real transducers. When communicating with simulated transducer nodes, the remaining – physically existing – nodes do not perceive a difference to a real network as regards the encoding scheme and the temporal behavior of the simulated nodes.

## 4.2 Simulation Gateway Node

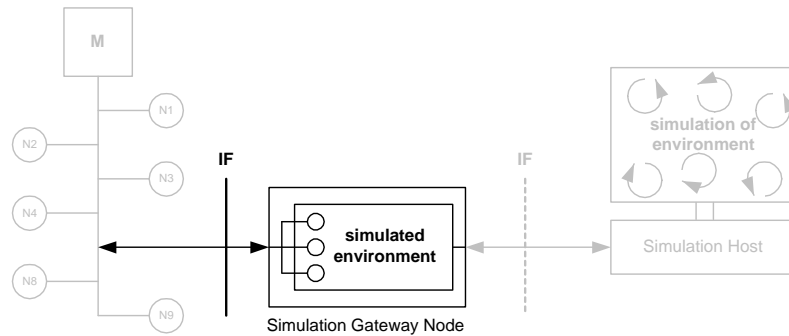


Figure 4: Simulation Gateway Node

To emulate nodes of a smart transducer network the proposed Simulation Architecture has to fulfill the following two requirements:

**Transparency:** The previously introduced concept of a temporal firewall has to be fully supported by the architecture. Nodes within a transducer network shall not be able to tell the difference between the partly emulated network and the network without simulation.

**Processing Power:** Simulation of the environment can soon become expensive in terms of CPU power. The Simulation Architecture has to provide enough processing power to timely generate simulation data.

The Simulation Gateway node guarantees *transparency in the time domain* by simulating the communication interface of the emulated nodes. All messages are sent at exactly the same time as they would have been sent from the emulated nodes.

While the Simulation Gateway Node targets at the first of the upper mentioned requirements (*transparency*), processing power is a bottleneck in cheap embedded devices. Consequently, the Simulation Gateway Node is not supposed to perform complex calculations for producing simulation values. Caused by this lack of processing power, in many cases it will be necessary to implement a Simulation Host to compute simulation data.

On the one hand the Simulation Gateway Node is part of the smart transducer network. It transmits simulation data to and receives data from the remaining nodes of the transducer network. On the other hand – due to limited processing power – it interacts with the powerful Simulation Host and receives / forwards the respective data from / to the Simulation Host.

There are several reasons for implementing a separate Simulation Gateway Node instead of connecting the Simulation Host directly to the transducer network.

- The physical behavior (e. g., voltage, signal shapes) shall be lifelike. Thus, it seems advantageous to use node devices that are also applied in the transducer network and therefore "behave like a smart transducer node".
- The separation of the simulation process and communication activities reduces the design complexity when developing new simulation models ("divide and conquer").
- It is not necessary to implement the fieldbus protocol at the Simulation Host, making it feasible to support different hardware devices that can be chosen according to individual purposes.

### 4.3 Simulation Host

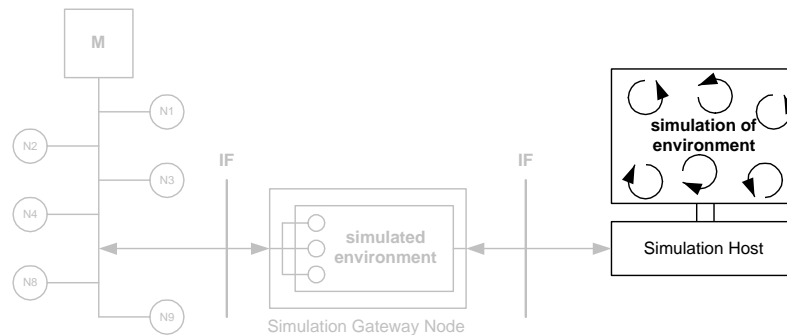


Figure 5: Simulation Host

When implementing Transducer Simulation the task of providing all necessary simulation values at statically specified points in time has to be handled. As mentioned in section 4.2, the Simulation Host has to provide sufficient processing power to deal with this stringent requirement. Furthermore, mechanisms to achieve real-time behavior of the Simulation Host, i. e., real-time operating system, tasks with known worst case execution time, static task schedule, etc., have to be implemented.

Since the interface between simulated transducers and environment model is an internal interface of the simulation process, the simulation programmer has some freedom in designing the interface properties – this interface may not be fully lifelike. A perfect simulator would generate messages indistinguishable from the messages which would have been sent by the real nodes. Nevertheless, in most cases the task of generating values for the emulated transducers can only be approximated.

### 4.4 Communication: Simulation Gateway Node ↔ Simulation Host

The distributed Simulation Architecture approach – as presented above – requires the transfer of data from the Simulation Host to the Simulation Gateway Node and vice versa. This communication is critical and bounded by the real-time requirements of the Simulation Gateway Node. From a conceptual point of view, two different approaches can be distinguished. As depicted in figures 7 and 8 the communication process can follow an event-triggered or a time-triggered scheme.

In this section only data transmission from the Simulation Host to the Simulation Gateway Node will be taken into consideration. Communication from Simulation Gateway

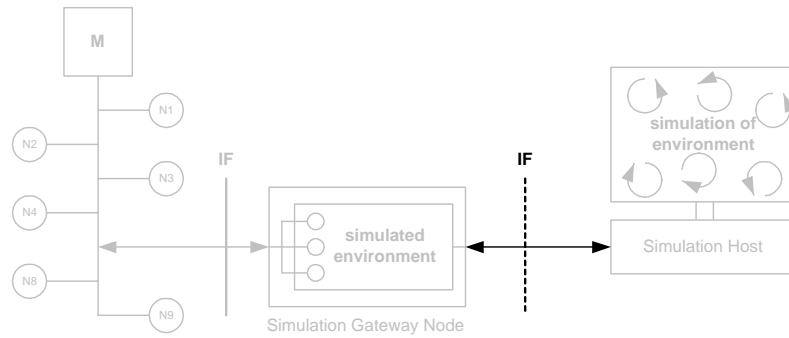


Figure 6: Communication between Simulation Gateway Node and Simulation Host

Node to Simulation Host is expected to happen likewise.

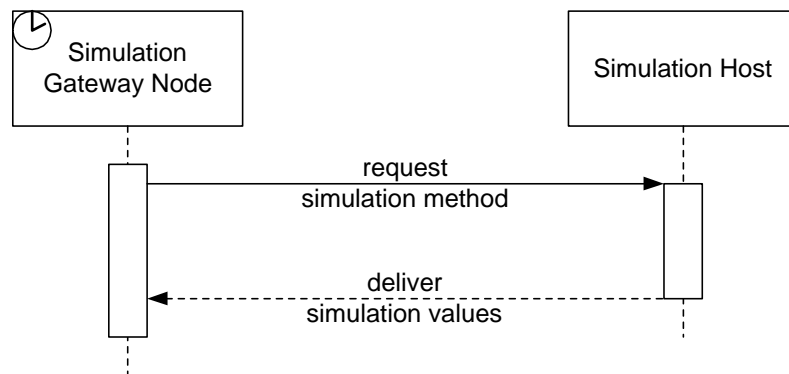


Figure 7: Event-triggered communication

In case of event-triggered communication, the Simulation Gateway Node requests the delivery of simulation data on demand. The clock of the Simulation Host is not synchronized to the clock of the Simulation Gateway Node. The Simulation Host waits until a request of the Simulation Gateway Node appears. Then it starts to calculate the required simulation values and sends them back within a certain time-span.

The time-triggered communication approach (see figure 8) requires the clock of the Simulation Host to be synchronized to the Simulation Gateway Node's clock. The Simulation Host starts to process the calculation of simulation data at some predefined point in time. The delivery of data to the Simulation Gateway Node also follows a statically defined schedule.

#### 4.4.1 Advantages of event-triggered approach:

Without the need of clock synchronization and communication schedules at the host side, the event-triggered approach is easier to implement, i. e., less effort is required to design the Simulation Host.

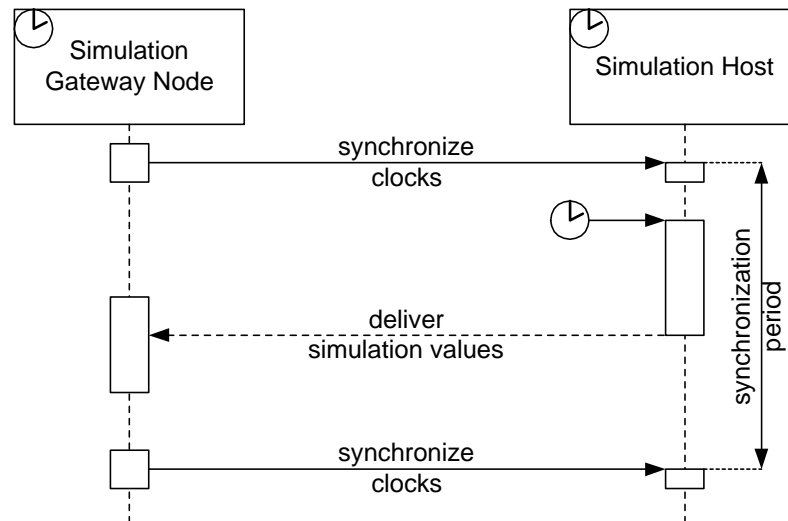


Figure 8: Time-triggered communication

#### 4.4.2 Advantages of time-triggered approach:

The time-triggered communication approach utilizes the Simulation Host CPU in a more efficient way. The Simulation Host is not forced to wait until a request from the Simulation Gateway Node occurs but may start a simulation process at any given point in time (statically scheduled). Furthermore, with time-triggered communication deadline violations are avoided that may occur in the event-triggered case for reason of unsynchronized clocks. It can be guaranteed that at least some "fall-back"-values are sent timely – even if the Simulation Host can't produce the expected simulation values within a given time interval.

## 5 Simulation Architecture Implementation

The integration of a simulation process into the above described Simulation Architecture is straight forward. First, the communication schedule of the system will be created including the communication slots for the nodes that will be simulated. The communication slots that have to be provided by the simulation are then assigned to the Simulation Gateway Node. In cases where in-the-loop tests are applied, tasks running at the Simulation Host will also need certain messages that are sent by the other (non-simulated) nodes. Thus, the Simulation Gateway Node will probably also receive messages and relay these messages to the Simulation Host.

To proof concepts of the Simulation Architecture presented in this paper, the proposed approach has been partly implemented on low-cost communication nodes that support the time-triggered fieldbus protocol TTP/A [9]. A smart transducer network (cluster) consisting of four TTP/A nodes has been set up. Figure 9 depicts the hardware set-up whereas the large component in the upper left (A) is the programmer and not part of the cluster.

The master (B), based on an Atmel AT90S8515, controls the application by triggering task execution and communication activities. Two slave nodes (C, D), both based on AT90S4433 microcontroller, represent the physically existing nodes, serving as ir-sensor

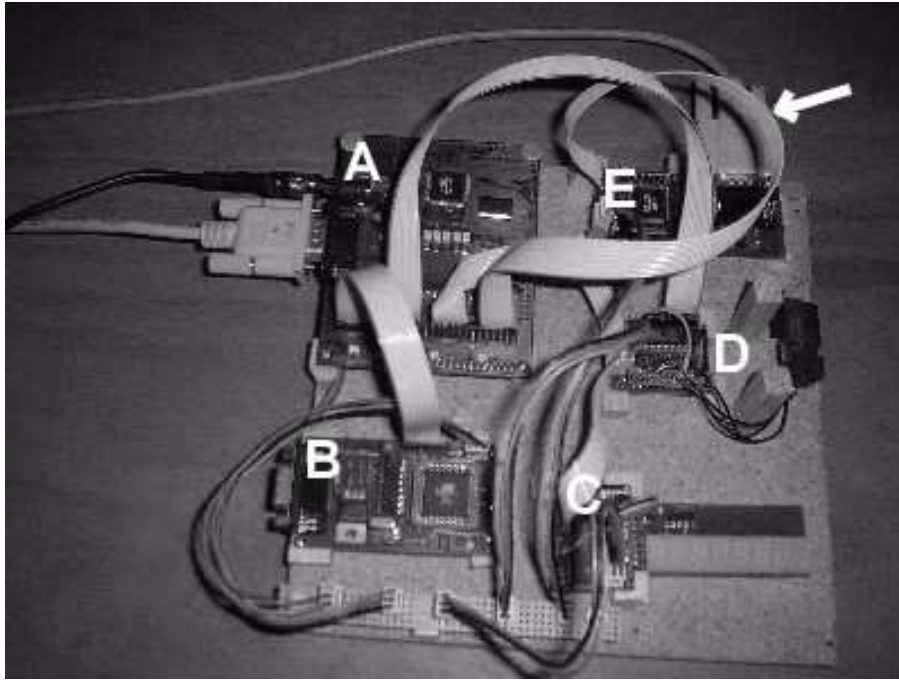


Figure 9: TTP/A-Cluster

unit (D) and display unit (C). Finally the Simulation Gateway Node (E), based on an ATmega128 chip, is connected to the cluster. The Simulation Gateway Node provides an additional interface that is accessible via RS232 (arrow in figure 9).

Although all three node types are based on the same microcontroller family, in principle, the TTPA protocol can be implemented on any microcontroller providing at least 64 bytes RAM and 2 kB ROM [8].

Currently, the cluster is connected to a standard Linux workstation which does not provide real-time functionality. At the present implementation stage, communication works in both directions (Simulation Gateway Node  $\leftrightarrow$  Simulation Host) but is only used for sending monitoring values from the Simulation Gateway Node to the Simulation Host.

The use of an operating system with real-time capabilities is a major prerequisite to achieve real-time behavior of the Simulation Host. Therefore, the next steps will include investigation on the usability of different real-time operating systems and the development of a real-time Simulation Host.

## 6 Conclusion and Outlook

In this paper a distributed approach to simulate transducer nodes in a time-triggered field-bus network has been presented. The existence of simulation methods strongly assist the development of embedded intelligent systems, i. e., systems that are able to react appropriately to changing situations without user input.

The partitioning of the proposed Simulation Architecture into a *Simulation Gateway Node* and a *Simulation Host* gives the developer some freedom when designing a simulation. The Simulation Gateway Node relieves the developer from timing considerations that are necessary to preserve the exact behavior of the simulated transducers. The Sim-

ulation Host provides sufficient processing power to implement even complex simulation models.

The proposed communication architecture has been partly implemented on a TTP/A fieldbus network. Based on this implementation the following future goals could be identified:

- Most important will be the completion of the missing parts of the Simulation Architecture, i. e., the implementation of a real-time Simulation Host and real-time communicating between the Simulation Host and the Simulation Gateway Node.
- An important matter is the predictability of the computation time for simulation tasks. The sending slot of the communication depicts also a deadline for the corresponding result. If the simulation task does not finish before this deadline, the temporal firewall cannot mask the timing error. Thus, an analysis of the worst-case execution time for the simulation task will be necessary.
- Usability of the simulation concepts is also an emerging topic. With the introduction of complex mathematical models for simulating transducer behavior the importance of engineering support even increases. Future work could improve the design of transducer simulation models by offering tool support for model generation.

## References

- [1] W. Elmenreich. *Intelligent Solutions for Embedded Systems*. Institute for Computer Engineering, Technical University of Vienna, Vienna, Austria, June 2003.
- [2] H. Kopetz. *Real-Time Systems, Design Principles for Distributed Embedded Applications*. Kluwer Academic Publishers, Boston, Dordrecht, London, 1997.
- [3] I. J. Busch-Vishniac. *Electromechanical Sensors and Actuators*. Springer, 1999.
- [4] Object Management Group (OMG). *Smart Transducers Interface Final Adopted Specification*, August 2002. Available at <http://www.omg.org> as document ptc/2002-10-02.
- [5] H. Kopetz and R. Nossal. Temporal firewalls in large distributed real-time systems. *Proceedings of the 6th IEEE Workshop on Future Trends of Distributed Computing Systems (FTDCS '97)*, pages 310–315, 1997.
- [6] J. Banks, J. S. Carson, B. L. Nelson, and D. M. Nicol. *Discrete-Event System Simulation*. Prentice Hall, Upper Saddle River, NJ, 2001.
- [7] T. Gallia. *Cluster Simulation in Time-Triggered Real-Time Systems*. PhD thesis, Technische Universität Wien, Institut für Technische Informatik, Vienna, Austria, 1999.
- [8] H. Kopetz, M. Holzmann, and W. Elmenreich. A universal smart transducer interface: TTP/A. *International Journal of Computer System Science & Engineering*, 16(2):71–77, March 2001.
- [9] H. Kopetz et al. Specification of the TTP/A protocol. Technical report, Technische Universität Wien, Institut für Technische Informatik, Vienna, Austria, September 2002. Version 2.00, Available at <http://www.ttpforum.org>.