TRUETIME: SIMULATION OF NETWORKED COMPUTER CONTROL SYSTEMS

Dan Henriksson, Anton Cervin, Martin Andersson, Karl-Erik Årzén

Department of Automatic Control
Lund University
Box 118, SE-22100 Lund, Sweden
Email: {dan,anton,mandersson,karlerik}@control.lth.se

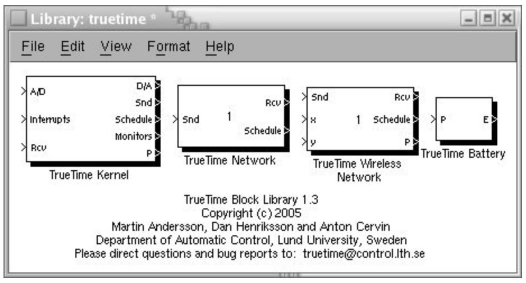
1. INTRODUCTION

Computer-based control systems and networked control systems are hybrid systems where continuous time-driven dynamics and discrete event-driven dynamics interact. The temporal non-determinism introduced by computing and communication in the form of delays and jitter can lead to significant performance degradation. Software tools are needed to analyze and simulate how the timing affects the control performance.

Timed automata and piecewise linear systems are common modeling formalisms for hybrid systems. TrueTime is a MATLAB/Simulink-based simulation tool that takes a completely different approach. Using TrueTime it is possible to simulate the temporal aspects of multi-tasking real-time kernels and wired or wireless networks within Simulink together with the continuous-time dynamics of the controlled plant. The approach allows simulation at the same level of detail as in the true system. For complete TrueTime descriptions, see (Andersson *et al.*, 2005a; Cervin *et al.*, 2003; Andersson *et al.*, 2005b). TrueTime is available for free download at `http://www.control.lth.se/user/dan/truetime` *Copyright © 2006 IFAC*

2. SIMULATION ENVIRONMENT

TrueTime consists of a block library with a computer kernel block and wired and wireless network blocks, as shown in Figure 1. The blocks are variable-step, discrete, MATLAB S-functions written in C++. The kernel block executes user-defined tasks and interrupt handlers representing, e.g., I/O tasks, control algorithms, and network interfaces. The scheduling policy



TrueTime Block Library 1.3
Copyright (c) 2005
Martin Andersson, Dan Henriksson and Anton Cervin
Department of Automatic Control, Lund University, Sweden
Please direct questions and bug reports to: truetime@control.lth.se

Fig. 1 The TrueTime block library.

of the kernel block is arbitrary and decided by the user. The network blocks distribute messages between computer nodes according to a chosen network model. The blocks are connected with ordinary continuous-time Simulink blocks to form a real-time control system.

All blocks are event-driven, with the execution determined both by internal and external events. Internal events correspond to events such as "a timer has expired," "a task has finished its execution," or "a message has completed its transmission." External events correspond to external interrupts, such as "a message arrived on the network" or "the crank angle passed zero degrees." All outputs are discrete-time signals. The Schedule and Monitors outputs display the allocation of common resources (CPU, monitors, network) during the simulation.

2.1 *The Kernel Block*

The kernel block S-function simulates a computer with a simple but flexible real-time kernel, A/D and D/A converters, a network interface, and external interrupt channels. Internally, the kernel maintains several data structures that are commonly found in a real-time kernel: a ready queue, a time queue, and records for tasks, interrupt handlers, monitors and timers that have been created for the simulation. The execution of tasks and interrupt handlers is defined by code functions, written either in C++ or MATLAB code. Control algorithms may also be defined graphically using ordinary discrete Simulink block diagrams.

Tasks and Interrupt Handlers Tasks are used to simulate both periodic activities, such as controller and I/O tasks, and aperiodic activities, such as communication tasks and event-driven controllers. Each task is defined by a set of attributes (priority, deadline, period, etc.) and a code function. Interrupts may be generated in two ways: externally or internally. An external interrupt is associated with one of the external interrupt channels of the kernel block. The interrupt is triggered when the signal of the corresponding channel changes value. This type of interrupt may be used to simulate engine controllers that are sampled against the rotation of the motor or distributed controllers that execute when measurements arrive on the network. Internal interrupts are associated with timers. Both periodic timers and one-shot timers can be created

Code The code associated with tasks and interrupt handlers is divided in segments, where the code of each segment is executed instantaneously during simulation. The code can interact with other tasks and with the environment at the beginning of each code segment. This execution model makes it possible to model input-output delays, blocking when accessing shared resources, etc. The simulated execution time of each segment is returned by the code function. Besides A/D and D/A conversion, many other kernel primitives exist that can be called from the code functions, e.g., functions to send and receive messages over the network, create and remove timers, perform monitor operations, and change task attributes.

2.2 *The Network Blocks*

The TrueTime network blocks distribute messages between computer nodes according to chosen network models. For wired networks, six of the most common medium access control protocols are supported (CSMA/CD (Ethernet), switched Ethernet, CSMA/CA (CAN), token-ring, FDMA, and TDMA). The wireless network block supports simulation of the IEEE 802.11 WLAN and IEEE 802.15.4 ZigBee standards. For a description of the simple radio model used for simulation of wireless communication, see (Andersson *et al.*, 2005b).

In the same way that code execution is modelled by segments as opposed to execution of individual statements, the network transmissions are not modelled on bit level. Rather, only the interactions between nodes relevant for the timing behavior of the transmissions are modelled. That includes pre- and post-processing delays, collision detection and collision avoidance mechanisms, and probabilities of lost packets. A message contains information about the sending and the receiving computer node, arbitrary user data (typically measurement signals or control signals), the length of the message, and optional real-time attributes such as a priority. When the simulated transmission of a message has completed, it is put in a buffer at the receiving computer node, which is notified by a hardware interrupt.

3. SIMULINK TIMING DETAILS

The TrueTime blocks are event-driven and support external interrupt handling. Therefore, the blocks have a continuous sample time, and the timing of the block is implemented using the Simulink zero-crossing functionality. The next time the kernel (or network block) should wake up (e.g., because a task is to be released from the time queue, a task has finished its execution, or a message transmssion has been completed) is denoted `nextHit`. If there is no known wake-up time, this variable is set to infinity. The basic structure of the zero-crossing function is

```
static void mdlZeroCrossings(SimStruct *S) {
   Store all inputs;
   if (any external interrupt input has
                              changed value) {
      nextHit = ssGetT(S);
   }
   ssGetNonsampledZCs(S)[0] = nextHit - ssGetT(S);
}
```

This will ensure that the Simulink call-back function `mdlOutputs` executes every time an internal or external event has occurred. The kernel and network functions are only called from `mdlOutputs` since this is where the outputs (D/A, schedule, network) can be changed. Further, since several kernel and network blocks may be connected in a circular fashion, *direct feedthrough* is not allowed. We exploit the fact that, when an input changes as a step, `mdlOutputs` is called, followed by `mdlZeroCrossings`. Since direct feedthrough is not allowed, the inputs may only be checked for changes in `mdlZeroCrossings`. There, the zero-crossing function is changed so that the next major step occurs at the current time.

REFERENCES

Andersson, M., D. Henriksson, and A. Cervin (2005a): *TrueTime 1.3—Reference Manual*. Department of Automatic Control, Lund University, Sweden.

Andersson, M., D. Henriksson, A. Cervin, and K.-E. Årzén (2005b): "Simulation of wireless networked control systems." In *Proceedings of the 44th IEEE Conference on Decision and Control and European Control Conference ECC 2005*. Seville, Spain.

Cervin, A., D. Henriksson, B. Lincoln, J. Eker, and K.-E. Årzén (2003): "How does control timing affect performance?" *IEEE Control Systems Magazine*, **23:3**, pp. 16–30.