Modeling and Analysis of Heterogeneous Industrial Networks Architectures

F.Fummi* S. Martini# M. Monguzzi‡ G. Perbellini# M. Poncino*

* Università di Verona, Verona, Italy
‡ Sitek S.p.A., S.G. Lupatoto, Italy
# Embedded Systems Design Center, Verona, Italy 37134

1. Introduction

In a modern factory automation plant, heterogeneous networks (point-to-point and multi-point links, field-bus) have become the rule, and the possibility of modeling and simulating them has become an essential feature. Designers leverage on connectivity support to integrate modules in different configurations and/or interfacing the whole system to legacy networks and field-bus.

A complex network in a factory automation application usually includes heterogeneous devices; Figure 1 shows a scenario, that will be used in this work as our generic template of a system. Central to this architecture is the Ultimodule ([1]) master, that is used to drive the following components:

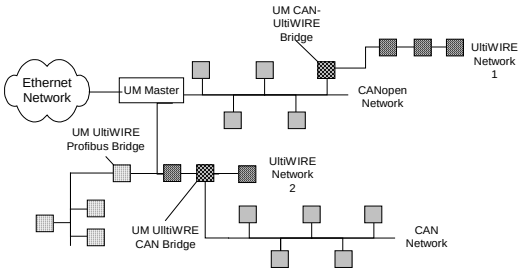

Figure 1. Typical Heterogeneous Network in a Factory Automation Environment.

(i) A network of CANopen devices that controls time constrained actuators typical of a plant; (ii) The UltiWIRE Network 1, that is connected through the CAN bus to a CAN-UltiWIRE protocol bridge; (iii) The UltiWIRE Network 2, that bridges the connectivity to further CANopen and Profibus devices. Profibus extends the system flexibility allowing to communicate with some other legacy COTS devices already installed in the plant; (iv) an Ethernet network that allows to interface with some local network.

We assume that the Ultimodule master is running a soft *PLC* (Programmable Logic Controller) application, with suitable drivers for accessing all the connected networks. In this paper we present a strategy to simulate an heterogeneous network and investigating what UltiWIRE specification would better fit a given timing requirement for the soft PLC.

2. Modeling

The heterogeneous network scenario reported in Figure 1 requires tools to model and analyze the performance of the various network components. We propose at first to use an homogeneous modeling strategy using the network simulator (NS-2), and then discuss the role of SystemC and the use of an instruction set simulator (ISS) and a board to improve the overall modeling methodology.

2.1. NS-2 only

Network Simulator 2 (NS-2) is used to build the network topology and to model as follows all devices of Figure 1.

An **UltiWIRE** network is a daisy chain network with one Master and one or more Slaves. Further details are available in [2]. To model the UltiWIRE bus protocol under NS-2 it is necessary: to define (i) the two frame types (Tx and Rx), and (ii) model the protocol by creating two new `Agent`s, one for the master (`UltiWIREmaster`) and one for the slave (`UltiWIREslave`). To allow the user to use the new objects, the simulator has to be modified to establish a link between the C++ objects and the OTcl interpreter. Each UltiWIRE node is modeled with an `Agent` attached on the NS-2 node: the agent can be the `UltiWIREmaster` or the `UltiWIREslave`.

The NS-2 model of **CAN** protocol is a C++ class extending the MAC (Media Access Control) C++ class of the NS-2 hierarchy. The NS-2 CAN object implements CSMA/CD to handle access to the shared bus. In CAN 2.0, a packet is defined to simulate the data communication via two different *Message Transfer*: The *Data Frame*, that carries data from a transmitter to the receivers in broadcast mode, and the *Remote Frame*, transmitted by a bus unit to request the transmission of the Data Frame with the same identifier.

The NS-2 model of the **CANopen** protocol is a C++ class derived from the NS-2 `Agent` class. To simulate a CANopen Network have been defined a new NS-2 Agent called `CANopenAgent`, that models the nodes on a CANopen network. The `CANopenAgent` provides the features to support a desired traffic profile. The CANopenAgent implements two operations to read/write from/to

1530-1591/04 $20.00 (c) 2004 IEEE

the memory of another CANopen node. These operation are exported to the applications running on it.
The NS-2 model of a **PLC** is a C++ class derived from the NS-2 `Application` class. An application is attached on an agent and it sends data through the underlying agent or by calling a set of methods defined in the agent.
Another application used to increase the traffic on the network is an **Interrupt Generator**. This application models the interrupt behavior of an UltiWIRE device. The interrupts are used by a UltiWIRE slave to inform the master of an event. The *Interrupt Generator* interrupts the master by setting a proper bit in the Rx frame. When the Rx frame reaches the master, the interrupt polling routine is called to determine which slave has generated the interrupt. After that, the master calls the ISR (Interrupt Service Routine) associated to the slave that has generated the interrupt. This ISR is modeled by a *Interrupt Handler* application that is attached on the UltiWIRE Master agent.

2.2. NS-2/SystemC

The previous section was centered around a homogeneous simulation scheme, where all components of the architecture of Figure 1 are implemented as NS-2 components, and all simulation occurs inside NS-2.
There exist situations, however, in which we would like to have the possibility of re-using previously specified (hardware) blocks, usually in a Hardware Description Language (HDL). In this section, we will describe how such HDL modules (specifically, SystemC description) can be integrated into NS-2 simulation to allow a heterogeneous simulation scheme. A motivation for such hybrid modeling is the possibility to analyze the functionality and the performance of a HDL reused protocol when attached to and existing network topology, before implementing it.
The NS-2 and SystemC integration has been realized using the co-simulation methodology proposed in [3], that provides a complete framework for time-accurate co-simulation of hybrid descriptions containing network topologies and hardware descriptions.

2.3. Board-based

The designer might be interested in applying realistic workloads by plugging real industrial applications to the heterogeneous network. For instance, one may to replace the PLC implementation in NS of Section 2.1, with a a real *softPLC* application; in this case, we have to communicate with the actual hardware controller running the softPLC, that is, the UltiWIRE master. This communication can be implemented by a driver for the RTOS running on the board implementing the controller (the UltiWIRE controller). If there is no hardware controller available to simulate this situation, an Instruction Set Simulator (ISS) can be used as a model of the controller. This *Driver/NS-2* co-simulation scheme is based on establishing the communication by means of a device driver, implemented inside the OS running on the board, manipulated to communicate with NS-2.

3. Performance Analysis

Figure 2 shows the actual system used to carry out the exploration of the various Ultimodule configurations.

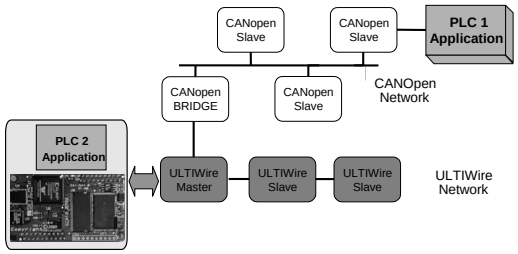

Figure 2. Case Study.

The heterogeneous network consists of two sub-networks: a UltiWIRE bus and a CANopen-over-CAN network. A Ultimodule SCM220 board (the box in Figure 2) hosts a CANopen controller and a UltiWIRE master controller. This configuration allows the board to be connected to a UltiWIRE bus and to a CANopen network. The PLC application running on the CANopen node reads/writes data from/to the CANopen controller on the board. The CANopen controller and the UltiWIRE controller are connected together through a CAN-UltiWIRE bridge. This bridge receives CANopen frames from the CANopen interface and executes the requests on a UltiWIRE slave. In this way, the PLC application on the CANopen node reads/writes data from/to a UltiWIRE slave. Another PLC application can be attached on the UltiWIRE master. This application adds traffic on the UltiWIRE bus and reduces the available bandwidth for the first PLC application. Therefore, if the first PLC application has to executes each burst in a given time (*maximum cycle time*), the UltiWIRE bus must be fast enough to guarantee the correct behavior of the PLC application.
The tests consisted of measuring the maximum cycle time of the PLC application, using two different network configurations corresponding to different traffic conditions.
Experimental analysis shows that the network cannot support the workload generated by the two PLCs, thus the UltiWIRE bandwidth must be increased.

References

[1] Ultimodule Inc., `http://www.ultimodule.com`.

[2] N. Drago, F. Fummi, M. Monguzzi, G. Perbellini, M. Poncino, “Estimation of Bus Performance for a Tuplespace in an Embedded Architecture”. *DATE’03: Design Automation and Test in Europe*, Munich, 2003.

[3] F. Fummi, P. Gallo, S. Martini, G. Perbellini, M. Poncino, F. Ricciato, “A Timing Accurate Modeling and Simulation Environment for Networked Systems”. *DAC-40: 40th Design Automation Conference*, Anaheim, CA, June 2003.