Modeling and Analysis of a Field Bus Protocol Using UPPAAL*

Alexandre David† Ulf Hammar‡ Wang Yi§

September 27, 1999

Abstract

We present an industrial application of the verification tool UPPAAL, in which a commercial communication protocol (AF100) has been modelled and analyzed. During the case study, a number of imperfections in the implementation have been found; improvements have been suggested.

AF100 is a field bus protocol (Advant Fieldbuss 100) developed and implemented in ABB Automation Products for safety-critical applications e.g. process control. Due to the complexity of the protocol and various changes made over the years, it shows occasionally unexpected behaviours e.g. deadlock etc. It has been time and resource–consuming to maintain the current implementation. It turns out that many of the problems come from the synchronization and timing mechanisms in the implementation of the protocol, in particular, semaphores and timeouts.

To our knowledge, the case study is the largest where the UPPAAL tool has been applied, which involves hundreds of pages of protocol specification and thousands of lines of source codes. In this talk, we shall present our experience in dealing with the complexity of the protocol using various abstraction features provided in UPPAAL. As an example, we study the bus coupler of AF100, which serves as the data link layer of the protocol.

1 Protocol Overview

AF100 is a field bus communication protocol (Advant Fieldbus). It is designed for 80 stations communicating over a bus. A station acting as a "master", may initiate a dialog with up to 79 other stations acting as "slaves" in this dialog. The master requests information from a slave which only responds to it, thus the names master and slave. In fact the dialog is done between applications in those stations and a station may have several applications running (acting as masters or slaves).

The protocol has two main layers which are: *VFI* (*Virtual Field Interface*) and the *Bus Coupler*. Typically a client application will use the master part of VFI to send requests to another station where a server application will respond though the slave part of VFI. VFI communicates with the bus via the Bus Coupler which uses a low level bus queue.

Figure 1 gives the layered structure of the protocol of AF100. The API services are on top of the structure, VFI covers the *Service Data Transfer* and the *Message Transfer Protocol* layers and the Bus Coupler is the *Packet Transport Protocol*.

2 Verification Overview

The method used to verify this protocol follows 5 phases:

*See URL: http://www.docs.uu.se/uppaal

†DoCS, Uppsala University
‡ABB Automation Products
§DoCS, Uppsala University

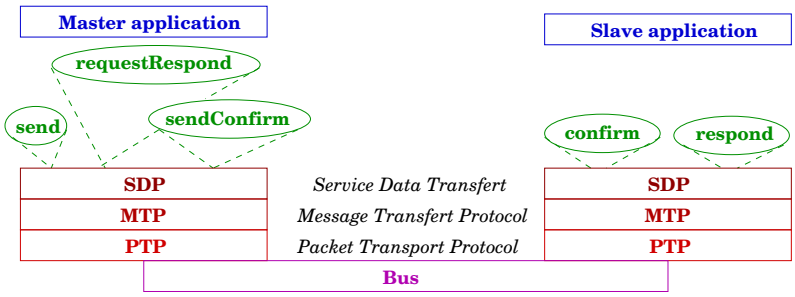

Figure 1: Layers of AF100.

1. model the Bus Coupler
2. derive an abstraction
3. model VFI master and slave separately
4. derive abstractions of the master and the slave part
5. compose complementary parts with a simplified Bus Coupler

The reasons of this approach are size and complexity. Because of state explosion such a complex model would not be possible to verify. We develop instead a composition verification by deriving abstract models from low level models. The low level models are those close to implementation.

We checked 82 properties concerning the Bus Coupler model and 35 properties for its abstraction. The properties are classified in the following cathegories:

1. safety properties, related to the logic of the protocol. Violating these properties result in inconsistent data read.
2. synchronization properties, related to the synchronisation of the components. Violating these properties could induce bad/wrong behaviour.
3. good behaviour properties, which are intuitively believed to hold with respect to the protocol. This is expected behaviour which has only performance impact.
4. validation properties, related to the model itself to validate it. The protocol works in practice and the model must work the same.

Different variations of the Bus Coupler model were checked and consumed 213MB, 320MB, 892MB and 600MB. Concerning the abstract model, the needed resources were for the different variants 3.8MB, 4.1MB, 5.0MB, 11MB and 14MB. These variants allowed us to control the exploration of the state space and to localize errors in the model.

The conclusion on the abstract model is that the protocol is implementable since the first model is valid. However the implementation has to avoid some possible race conditions as well as some delays. This is a feasability proof with warnings on the sensitivity of the behaviour. This is confirmed by the first variant of the model which prune such bad states, but detect their possible appearance.