Simulation of Process Control with WirelessHART Networks Subject to Packet Losses

Mauro De Biasi, Carlo Snickars, Krister Landernäs, and Alf J. Isaksson*
ABB AB, Corporate Research
SE-721 78 Västerås, Sweden

***Abstract*— This paper describes an extension of the open source Simulink package TRUETIME for simulation of networked control systems. This extension enables simulation of wireless control using the recently released WirelessHART standard. An example is also presented which illustrates the simulated effect of packet loss on control performance.**

I. INTRODUCTION

Wireless networks have been attracting considerable attention. For many applications where monitoring is the only requirement these networks offer considerable value. What about control?

Today there is really only one released standard for wireless communication in the process industry, viz. WirelessHART [2]. To enable simulation of wireless control using WirelessHART an extension has been made to the simulation environment TRUETIME [1].

This paper presents this WirelessHART extension to TRUETIME. It starts by a general introduction to the WirelessHART standard in Section II, followed by a brief introduction of the open source package TRUETIME. In section IV the implementation of WirelessHART into TRUETIME is described in some detail, and in Section V some other modifications are briefly described. This is followed by a small simulation example in Section VI and some conclusions in the last section.

II. WIRELESSHART DESCRIPTION

WirelessHART is an extension to the (wired) HART standard, that provides an optional low cost, relatively low speed (e.g. compared to IEEE 802.11g) wireless connection. It adopts the IEEE 802.15.4 physichal layer and operates in the 2.4 GHz ISM radio band using 15 different channels. The communication between the devices is performed using Time Division Multiple Access (TDMA) with time slots of 10 ms. A series of time slots form a superframe which can be of arbitrary length. WirelessHART also enables channel hopping to avoid interferences and reduce multi-path fading effects. One or more sources and one or more destination devices may be scheduled to communicate in a given slot. The slot may be dedicated to communication from a single source device or a slot may support shared communication. In the latter case, the MAC protocol used is CSMA/CA. HART is loosely organized around the ISO/OSI 7-layer model for communication protocols.

*Corresponding author email: alf.isaksson@se.abb.com

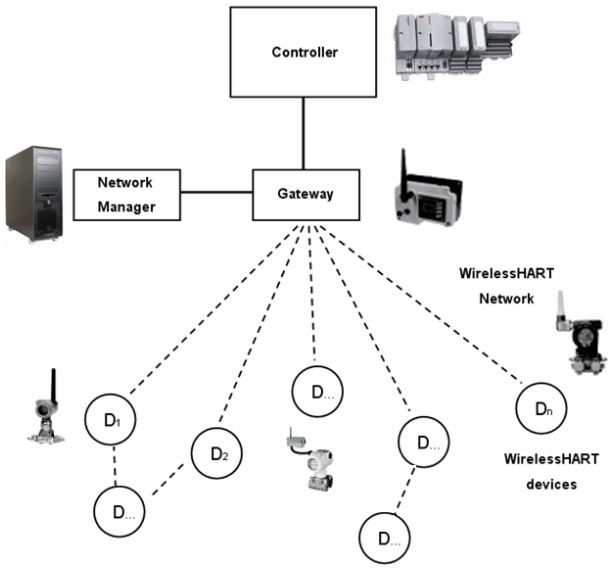

Fig. 1. The structure of a WirelessHART network.

The Structure of a WirelessHART network is shown in Fig. 1. All communications of the WirelessHART Network pass through the gateway. Consequently, the gateway must route packets to the specified destination. The gateway uses standard HART commands to communicate with network devices and host applications. The plant automation network could be a TCP-based network, a remote IO system, or a bus such as PROFIBUS DP. The Network Manager creates an initial superframe and configures the Gateway. A detailed description of the components of a WirelessHART network is given in [3] and [4].
The MAC protocol specifies when a device is allowed to transmit a message.

A. MAC Protocol Description

The main tasks of the Medium Access Control (MAC) protocol are:

- slot synchronization
- identification of devices that need to access the medium
- propagation of messages received from the Network Layer
- to listen for packets being propagated from neighbors

The MAC sub-layer is, hence, responsible for propagating Data-Link packets (DLPDUs) across a link. To permit this, the device includes:

- Tables of neighbors, superframes, links, and graphs that configure the communication between the device and its neighbors (see Subsection IV-B). These tables are normally populated by the Network Manager. In addition the neighbors table is populated as neighbors are discovered.
- A link scheduler that evaluates the device tables and chooses the next slot to be serviced by listening for a packet or by sending a packet.
- State machines that control the propagation of packets through the MAC sub-layer. MAC Operation consists of schedule maintenance and service slots. MAC operation is fundamentally event driven and responds to service primitive invocations and the start of slots needing servicing.

In the next subsections a brief view of the WirelessHART MAC protocol is introduced. A deeper explanation of the protocol is given in [2].

B. Time Division Multiple Access (TDMA)

WirelessHART uses TDMA and channel hopping to control access to the network. TDMA is a widely used Medium Access Control technique that provides collision free, deterministic communications. It uses time slots where communications between devices occur. A series of time slots form a TDMA superframe (see Fig. 2).

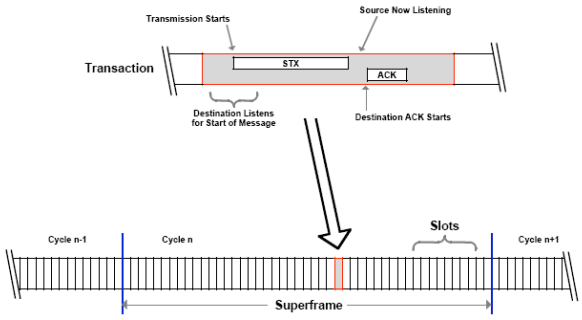

Fig. 2. The SuperFrame structure [2]

All devices must support multiple superframes. Slot sizes and the superframe length (number of slots) are fixed a priori and form a network cycle with a fixed repetition rate. Superframes are repeated continuously. For successful and efficient TDMA communications, synchronization of clocks between devices in the network is critical. Consequently, tolerances on time keeping and time synchronization mechanisms are specified to ensure network-wide device clock synchronization. It is imperative that devices know when the start of a slot occurs. Within the slot, transmission of the source message starts at a specified time after the beginning of a slot. This short time delay allows the source and destination to set their frequency channel and allows the receiver to begin listening on the specified channel. Since there is a tolerance on clocks, the receiver must start to listen before the ideal transmission start time and continue listening after that ideal time. The maximum total packet length is 127 bytes, where 80 bytes may be user specific payload.
Once the transmission is complete the destination device indicates, by transmitting an ACK, whether it received the source device data-link packet successfully or with a specific class of detected errors. To enhance reliability, channel hopping is combined with TDMA. Channel hopping provides frequency diversity, which can avoid interferences and reduce multi-path fading effects. Channel blacklisting is another special feature provided by WirelessHART. Communicating devices are assigned to a superframe, slot, and channel offset.

C. Shared Slot

WirelessHART allows to define shared slots in which more than one device may try to transmit a message. Consequently, collisions may occur within such a slot. If a collision occurs, the destination device will not be able to successfully receive any source transmission and will not produce an acknowledgment to any of them. To reduce the probability of repeated collisions, source devices shall use random back-off delay when their transmission in a shared slot is not acknowledged. A device maintains two variables for each neighbor: Back-Off Exponent (BOExp) and Back-Off Counter (BOCntr). Both of these variables are initialized to 0. When a transaction in a shared slot fails the random back-off period is calculated based on the BOExp. For each unsuccessful attempt by the source device in a shared slot the BOExp is incremented and a sequential set of numbers calculated. The set of numbers consists of the whole numbers $\{0, 1, ...L\}$ where

$$L = (2^{BOExp} - 1) \tag{1}$$

D. Communication Tables

All devices maintain a series of tables that control the communications performed by the device. The communication tables and the relationships between them are shown in Fig. 3. The WirelessHART simulator uses a simplified

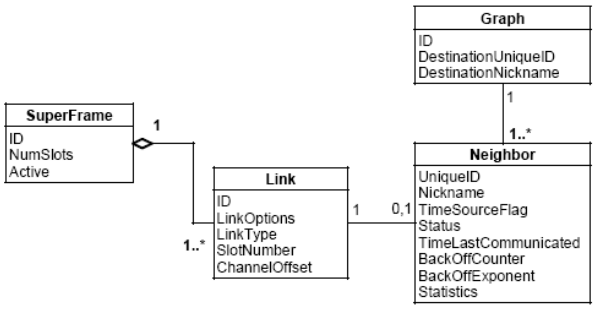

Fig. 3. The communication tables [2]

version of these tables.

III. DESCRIPTION OF TRUETIME

This section describes the use of the original Matlab/Simulink-based simulator TRUETIME [1], which permits to design networked control systems simulating real-time kernels, network transmissions (using wired or wireless networks), and continuous plant dynamics. TRUETIME consists of a six blocks library (see Fig. 4) and a collection of C++ functions with corresponding MATLAB MEX-interfaces.

These functions are divided into two groups. One permits to configure the simulation by creating tasks, interrupt handlers, monitors, timers, etc. The other functions are real-time primitives that are called from the task code during execution and provides for AD-DA conversion, changing task attributes, entering and leaving monitors, sending and receiving network messages, etc. TRUETIME is developed to work together with MATLAB/Simulink, which allows to interface the kernel and the tasks with models for continuous-time processes.

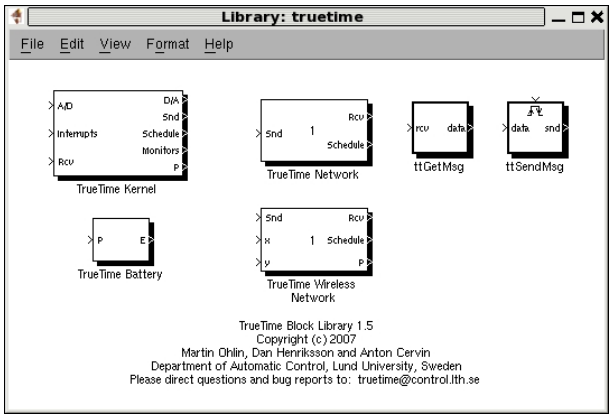

Fig. 4. The TRUETIME 1.5 block library.

A. The TRUETIME *Kernel* Block

One of the blocks contained in the library is the Kernel block. It is a MATLAB S-function that simulates a CPU with a real-time kernel, A/D and D/A converters, a network interface, and external interrupt channels. The kernel is designed following a real-time model with a ready queue and a time queue. It is also characterized by records for tasks, interrupt handlers, monitors and timers that have been created for the simulation. The kernel executes an arbitrary number of user-defined tasks and interrupt handlers that may also be created dynamically at run-time. Tasks may be periodic to simulate activities such as controller and I/O tasks, or aperiodic to represent activities like communication tasks and event-driven controllers. Aperiodic tasks are executed by the creation of task instances (jobs).

B. The TRUETIME *Network* Block

The TRUETIME network block permits to simulate medium access and packet transmission in a local area network choosing different communication protocols: CSMA/CD (e.g. Ethernet), CSMA/AMP (e.g. CAN), Round Robin (e.g. Token Bus), FDMA, TDMA (e.g. TTP), and Switched Ethernet. Only packet-level simulation is supported. It is, in fact, assumed that the messages have been divided into packets at higher protocol levels. When a node tries to transmit a message (using the primitive `ttSendMsg`), a triggering signal is sent to the network block on the corresponding input channel. At the end of the transmission, the network block sends a new triggering signal on the output channel corresponding to the receiving node. Each receiving node has a buffer in which the transmitted message is put. A message may be characterized by optional real-time attributes such as a priority or a deadline. The network block parameters (i.e. datarate, minimum frame size, loss probability) may be set using a graphic mask.

C. The TRUETIME *Standalone Network Blocks*

The standalone network blocks are two, the `ttSendMsg` and the `ttGetMsg`. They can be used to send messages using the network blocks without using kernel blocks. This permits (not having to initialize kernels, create and install interrupt handlers, etc.) to build quickly a simulation, without creating any Matlab M-files.

D. The TRUETIME *Wireless Network Block Behaviour*

The wireless network block simulates medium access and packet transmission. Originally it implemented two kinds of communication protocols, 802.11b/g (WLAN) and 802.15.4 (ZigBee). The possibility to use also WirelessHART has now been added. The block permits to simulate packets lost using a packet error probability or specifying when a packet has to be discharged. It permits to compare more simulations together taking into account how the same sequence of packets lost affects them.

1) Calculation of Error Probabilities: When a message is sent the SNR in the receiver is calculated, and a probabilistic measure is used to determine the number of bit errors. The digital modulation assumed is BPSK. A configurable error coding threshold is used to determine whether the package can be reconstructed. Defining the number of bits in the message, n, and the probability that a certain bit is erroneous, p , the number of bit errors, X, belongs to a binomial distribution $X \in Bin(n,p)$. This assumption can be made considering that bit errors in a message are uncorrelated. Using the central limit theorem, if the value of n is large, the binomial distribution can be approximated with a normal distribution. This gives that $X \in N(np, \sqrt{npq})$ where $q = 1 - p$. The message can be reconstructed if bn, where b is the error coding threshold, is larger than the total number of bit errors in that message.

This probability is calculated using

$$P(X \leq bn) = \begin{cases} \Phi(\frac{bn-np}{\sqrt{npq}}) & \text{if} \quad bn - np > 0 \\ 1 - \Phi(\frac{|bn-np|}{\sqrt{npq}}) & \text{if} \quad bn - np \leq 0 \end{cases} \tag{2}$$

where Φ is the standard normal cumulative distribution function.

2) User-Defined Path-Loss Function: The default path-loss function (or propagation model) used in the TRUETIME wireless simulations is [6]:

$$P_{receiver} = \frac{1}{d^a} P_{sender} \tag{3}$$

where P is the power, d is the distance in meters, and a is a parameter that can be chosen to model different environments. This model is often used in simulations, but if the user prefers to implement his own function, he may do it by writing a Matlab M-file where all the built-in functions available in Matlab or Simulink could be called.

IV. IMPLEMENTATION OF WIRELESS HART IN TRUETIME

The WirelessHART MAC protocol has been implemented with some C++ functions with corresponding MATLAB MEX-interfaces. The main function (ttMAC.cpp) implements the algorithm that permits the access to the medium. In the following subsections some technical details will be described.

A. MAC Protocol

As described in Section II WirelessHART uses TDMA and channel hopping to control the access to the medium. Each device has a table in which all the information for the communication is specified (see Subsection IV-B). When a device wants to transmit a message it must call the MAC function, that reads the device table and checks if the device is allowed to transmit. If yes, the transmission is permitted otherwise it is blocked. The channel hopping technique permits that different devices transmit in the same time slot using different channels. WirelessHART also allows the use of shared slots, for that reason a collision detection system has been implemented. When a device tries to transmit in a shared slot the MAC function verifies the status of the channel. If the channel is occupied a back-off is computed in the way explained in Section II-C. To understand better how the MAC protocol is implemented the pseudo-code of the algorithm is shown in Fig. 5.

```
* Read the actual simulation time
* Read the device table
 * if device will communicate in this slot
    * In case device has to receive
       * set the state to reception
       * if(slot=shared && BOcntr>0)
          * BOcntr=BOcntr-1;

    * in case device has to transmit
       * if (slot=shared && BOcntr=0
        && channel=occupied)
          * set the state to COLLISION;
          * Collision detection and handling:
             * BOexp=BOexp+1;
             * BOcntr=random(0,(pow(2,BOexp)

              -1));
       * else if(slot=shared && BOcntr>0)
          * set the state to COLLISION
          * BOcntr=BOcntr-1

       * else if(slot=shared && channel!=
       occupied && BOcntr=0)
          * set the state to transmission
          * set the channel to occupied
          * reset the BOexp (BOexp=0)

       * else if(slot!=shared)
          * set the state to transmission
          * set the channel to occupied
          * reset the BOexp (BOexp=0)

* if actual slot not reserved for this
device and is not a shared slot
    * reset the BOexp (BOexp=0)
    * reset the BOcntr (BOcntr=0)
    * set the state to BLOCKED,
      this device cannot access medium

      in this time slot
```

Fig. 5. MAC Algorithm

B. Devices Tables

In the WirelessHART protocol it is specified that each device must have a particular table in which all the communication details are specified. The main information that the table must contain are the time slots in which the device has to communicate, the information for the channel hopping technique, the device with which the communication has to be done and the information about each link (for example if the link is shared).

C. Synchronization of Device Tasks

In TDMA communication each device must know when the start of a time slot occurs. For that reason the first operation in the MAC function is to read the actual simulation time from the MATLAB environment. With this value the device is able to compute the actual slot time :

$$ActualSlotNumber = \left(\frac{Actual\_Sim\_time + exectime}{SlotSize} \% SuperframeSize\right) + 1 \tag{4}$$

where the *exectime* is the execution time of the device task that has called the MAC function. The *SlotSize* is fixed to 10 ms in WirelessHART and the *SuperframeSize* is the number of slots contained in a superframe. Clock drift is not taken into account. All the devices of the network are considered synchronized according to the WirelessHART specification.

V. MODIFIED TRUETIME

In this section some new utilities in TRUETIME, implemented to improve the behavior and the use of the network, are described.

A. Nodes in the 3D Space

Originally TRUETIME allows to collocate the devices on a 2D plane. It does not permit to build up a real scenario in which the nodes could be placed at different heights, for example attached to the same column. For that reason, the possibility to specify the (X,Y,Z) coordinates (meters) in the space for each node has been introduced.
This new information has to be considered in the computation of the signal power received during a transmission (3). Considering the new coordinates, the distance between the node i and the node j is:

$$d = \sqrt{(X_i - X_j)^2 + (Y_i - Y_j)^2 + (Z_i - Z_j)^2} \tag{5}$$

This new computation of the distance is used in the calculation of path loss,

B. External Noise Port

The TRUETIME default error probability function takes into account a signal to noise ratio SNR, for the transmitted packet *i*, computed in the following way:

$$SNR_i = \frac{P_{receiver_i}}{\sum_{j=1}^{N} P_{receiver_j} - P_{receiver_i}} \qquad (6)$$

where $P_{receiver_i}$ is the power of the packets *i*, and $\sum_{j=1}^{N} P_{receiver_j}$ is the total power at the receiver node. This formula considers the interferences due to other nodes, but does not take into account other sources of interferences like thermal noise, fading or other radio traffic not belonging to this network. For that reason an input port is added. It permits to specify for each device an external source of noise that can be, for example, a white noise block or a sequence of measurements taken from a real case (see Fig. 6).

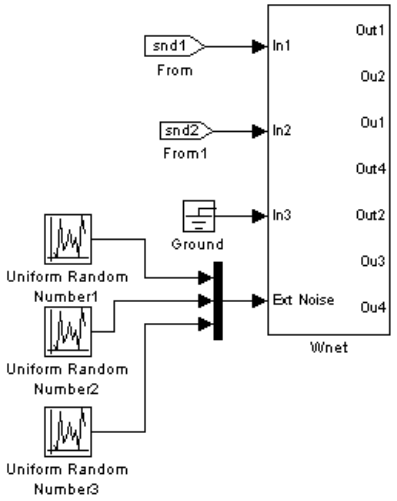

Fig. 6. The wireless network block with the external noise input port.

This external contribution is introduced in (6) as

$$SNR_i = \frac{P_{receiver_i}}{\sum_{j=1}^{N} P_{receiver_j} - P_{receiver_i} + N_i} \qquad (7)$$

where N_i is the contribution of this new component to the SNR of the transmitted packet *i*.

C. Noise and Time Correlation

In a real environment, where the nodes are fixed in one place and not in movement, if an event causes interferences, the effects produced by this will not vanish immediately, but will have an impact also in subsequent moments. Considering this fact, a time correlation is introduced in the computation of the interferences:

$$N_i = a \cdot N_{i-1} + V_i, \qquad V_i \sim \mathcal{N}(\mu_V, \sigma_V^2) \qquad (8)$$

where a (currently fixed at the value of 0.9) is a coefficient that indicates the correlation between two samples, and $\mathcal{N}(\mu_V, \sigma_V^2)$ a normal distribution with mean μ_V and variance σ_V^2. The initial value of the sequence, N_0, is picked from the external noise port.
To chose with which order of magnitude of interference to work, it is possible to set μ_V and σ_V^2 of the gaussian noise from the mask interface of the wireless network block.

D. Packets Lost Signal

To have a complete mastery of the simulation, it is useful to monitor the packets lost situation. For this reason, a new graphics function has been added. It allows to highlight in a chart when a packet is lost, due to a collision, external interference or excessive attenuation of signal. This permits to study in a simpler way how it affects the control performance (see Fig. 10).

E. Fixed Packet Loss Functionality

To study the problem of packet loss in the wireless control networks and to compare the results of different solutions, the possibility to fix the time intervals in which the packets are lost has also been added. This is done by supplying in the mask for the wireless network block a vector

$$[\Delta t_{1start} \;\; \Delta t_{1stop} \;\; \Delta t_{2start} \;\; \Delta t_{2stop} \;\; \ldots \;\; \Delta t_{nstart} \;\; \Delta t_{nstop}]$$

where Δt_{istart} is the instant, in seconds, when the network start to lose packets and Δt_{istop} is the final instant of the interval.

F. Wireless Network Mask Modification

The possibility to chose the WirelessHART MAC protocol in the wireless network block mask has been added. WirelessHART uses the same parameters as for Wlan and ZigBee with the addition of three fields to set the number of channels, the size of the slot and the superframe length. All the features can be configured using this mask, except for the device tables that have to be specified into a MATLAB M-file.

G. Integrating the MAC Into Each Device

In the original version of TRUETIME the wireless network block is responsible for the implementation of the MAC protocols of all the devices, as well as the simulation of packets loss and graphic plot (Fig. 7). This does not correspond perfectly to reality. In fact, in a real environment, each device has its own MAC sub-layer. When a device wants to access the network it has to interrogate the MAC in a local way (see Fig. 8).

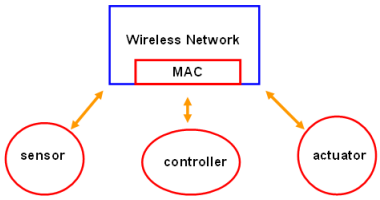

Fig. 7. MAC protocol location in TRUETIME

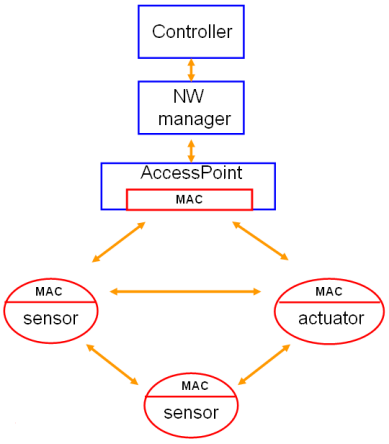

Fig. 8. MAC protocol location in a real environment

VI. Simulation Example

This section contains simulations of control with two alternative communcation solutions; WirelessHART and ZigBee. These simulations are included only to show the possibilities provided by the modified tool. The test scenario is a modification of the DC motor demo that comes with TrueTime. Here wireless PD control of two DC motors given by

$$G(s) = \frac{50}{s(25s+1)}$$

is studied. The PD-controllers are implemented according to:

$$\begin{aligned} P_k &= K \cdot (r_k - y_k) \\ D_k &= a_d D_{k-1} + b_d(y_{k-1} - y_k) \\ u_k &= P_k + D_k \end{aligned} \tag{9}$$

where $a_d = \frac{T_d}{NT_s+T_d}$, $b_d = \frac{NKT_d}{NT_s+T_d}$, T_s is the sample time, T_d is the derivative constant, K is the gain of the controller and N is a factor for the filtering of the derivative part:

$$D(s) = -\frac{sKT_d}{1+\frac{sT_d}{N}}Y(s) \tag{10}$$

Hence the ideal derivative sT_d is filtered by a first order system with the time constant T_d/N. This means that high frequency measurement noise is amplified at most by a factor KN (more details could be found in [7]). The parameters in the digital implementation are $T_s = 0.05, T_d = 0.26, K = 14$.

In the case of WirelessHART, loop 1 uses multi-hop routing and loop 2 single-hop. The used superframe is shown in Fig. 9, including the scheduling of the two controller execution tasks who are assumed synchronized with the Hart network. For ZigBee the communication is contention based and three retransmissions are allowed before dropping a packet.

A comparison of the simulation results is shown in Fig. 10. For this particular (random) pattern of packet loss WirelessHART with multi-hop gives a significantly improved control compared to single-hop, and is also superior to ZigBee. Notice, however, that no effort has been made to optimize the control performance.

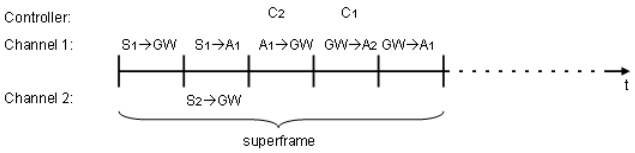

Fig. 9. Superframe scheduling for WirelessHART.

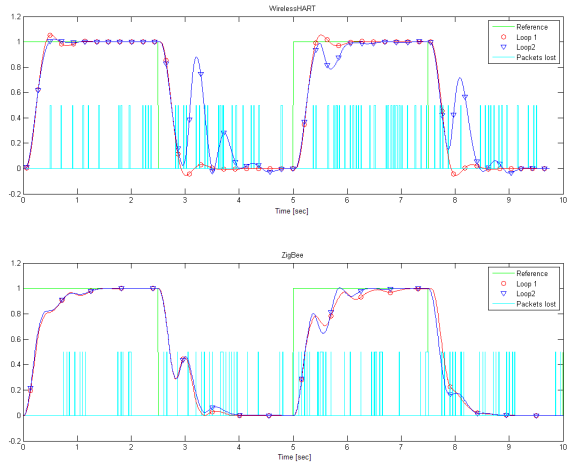

Fig. 10. Simulation of WirelessHART (top) and Zigbee (bottom)

VII. Conclusions

This paper has described the implementation of WirelessHART in TrueTime. Discussions have been initiated to make this part of the open source TrueTime provided by Lund University. This work forms an excellent basis for further study of the use of WirelessHART for closed-loop process control. As an illustration in this paper we chose a rather challenging one with PD control of two motors that have to be sampled at 50 ms or less. Our research in the future will mainly focus on slower more typical plants in process industry, but with more control loops per superframe.

VIII. Acknowledgments

The authors gratefully acknowledge many valuable discussions with our colleagues within the EU project SOCRADES.

References

[1] M. Ohlin, D. Henriksson, A. Cervin, Department of Automatic Control Lund University - TrueTime *1.5-Reference Manual* - January 2007.

[2] HART communication foundation, *TDMA Data Link Layer, HCF_SPEC – 075 Revision 1.0*, 30 August 2007.

[3] HART communication foundation, *Wireless Devices Specification, HCF_SPEC – 290 Revision 1.0*, 5 September, 2007.

[4] HART communication foundation, *Network Management Specification, HCF_SPEC – 085, Revision 1.0*, 27 August, 2007.

[5] J. Song, S. Han, A.K. Mok, D. Chen, M. Lucas, M. Nixon, W. Pratt, *WirelessHART: Applying Wireless Technology in Real-Time Industrial Process Control*, 14th IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS), 2007

[6] S. Mangold, S. Choi, N. Esseling, *An Error for Radio Transmitting Of Wireless LANs at 5GHz*, in Proc. 10th Aachen Symposium on Signal Theory, pages 209-214, September 2001

[7] K. J. Åström and T. Hägglund , *PID control - Theory, Design, and Tuning* , ISA, 1995