Technical Report

A Scalable and Efficient Approach for Obtaining Measurements in CAN-based Control Systems

Björn Andersson, Nuno Pereira, Wilfried Elmenreich

Eduardo Tovar, Filipe Pacheco and Nuno Cruz

TR- 061102

Version: 2.0

Date: July 2006; Revised: November 2007 and January 2008

A Scalable and Efficient Approach for Obtaining Measurements in CAN-based Control Systems (previously: Using a Prioritized MAC Protocol to Efficiently Compute Aggregated Quantities in a Single Broadcast Domain)

Björn ANDERSSON, Nuno PEREIRA, Wilfried ELMENREICH, Eduardo TOVAR, Filipe PACHECO and Nuno CRUZ

IPP-HURRAY!

Polytechnic Institute of Porto (ISEP-IPP)

Rua Dr. António Bernardino de Almeida, 431

4200-072 Porto

Portugal

Tel.: +351.22.8340509, Fax: +351.22.8340509

E-mail: {bandersson, npereira, emt,ffp, npsc}@dei.isep.ipp.pt; wil@vmars.tuwien.ac.at

http://www.hurray.isep.ipp.pt

Abstract

The availability of small inexpensive sensor elements enables the employment of large wired or wireless sensor networks for feeding control systems. Unfortunately, the need to transmit a large number of sensor measurements over a network negatively affects the timing parameters of the control loop. This paper presents a solution to this problem by representing sensor measurements with an approximate representation --- an interpolation of sensor measurements as a function of space coordinates. A priority-based medium access control (MAC) protocol is used to select the sensor messages with high information content. Thus, the information from a large number of sensor measurements is conveyed within a few messages. This approach greatly reduces the time for obtaining a snapshot of the environment state and therefore supports the real-time requirements of feedback control loops.

A Scalable and Efficient Approach for Obtaining Measurements in CAN-based Control Systems

Björn Andersson, *Member, IEEE,* Nuno Pereira, *Member, IEEE,* Wilfried Elmenreich, *Member, IEEE,* Eduardo Tovar, *Member, IEEE,* Filipe Pacheco, *Member, IEEE,* and Nuno Cruz

Abstract

The availability of small inexpensive sensor elements enables the employment of large wired or wireless sensor networks for feeding control systems. Unfortunately, the need to transmit a large number of sensor measurements over a network negatively affects the timing parameters of the control loop. This paper presents a solution to this problem by representing sensor measurements with an approximate representation — an interpolation of sensor measurements as a function of space coordinates. A priority-based medium access control (MAC) protocol is used to select the sensor messages with high information content. Thus, the information from a large number of sensor measurements is conveyed within a few messages. This approach greatly reduces the time for obtaining a snapshot of the environment state and therefore supports the real-time requirements of feedback control loops.

Index Terms

Process Control, Transducers, Data Processing, Measurement System Data Handling, Local Area Networks.

This work was supported in part by the Portuguese Foundation for Science and Technology (Fundação para a Ciência e Tecnologia - FCT) and the Network of Excellence on Embedded Systems Design ARTIST2 (IST-004527).

Björn Andersson is with the CISTER/IPP-Hurray Research Unit, at the Polytechnic Institute of Porto (e-mail: bandersson@dei.isep.ipp.pt).

Nuno Pereira is with the CISTER/IPP-Hurray Research Unit, at the Polytechnic Institute of Porto (e-mail: nap@isep.ipp.pt).

Wilfried Elmenreich is with Vienna University of Technology (e-mail:wil@vmars.tuwien.ac.at). He visited the CISTER/IPP-Hurray Research Unit, at the Polytechnic Institute of Porto during May-June 2007.

Eduardo Tovar is with the CISTER/IPP-Hurray Research Unit, at the Polytechnic Institute of Porto, Portugal (e-mail: emt@dei.isep.ipp.pt).

Filipe Pacheco is with the CISTER/IPP-Hurray Research Unit, at the Polytechnic Institute of Porto (e-mail: ffp@dei.isep.ipp.pt).

Nuno Cruz is with the CISTER/IPP-Hurray Research Unit, at the Polytechnic Institute of Porto (e-mail: npsc@isep.ipp.pt).

I. Introduction

A process control system needs to obtain values from sensors in order to infer the state of the physical process that is controlled. The availability of small inexpensive sensor elements has made it possible to deploy a large number of sensors for the purpose of receiving a better image of the physical process. However, since many sensors are typically placed within one broadcast domain, increasing the number of sensors increases also the time to obtain a full set of measurements accordingly. On the other hand, most control algorithms are quite sensitive to an increase of the measurement time and therefore would require a re-design of the control strategy for a particular number of sensors [1].

In this paper we show that a prioritized medium access control (MAC) protocol, like the one implemented in the Controller Area Network (CAN) bus, enables to dramatically reduce the number of messages that need to be transmitted, and this reduces the time complexity for obtaining certain aggregated quantities in a single broadcast domain. We show that the minimum value (MIN) can be obtained with a time complexity that is O(N_{PRIOBITS}), where N_{PRIOBITS} is the number of bits used to represent the sensor data. Note that, in this case, the message complexity (and thus, the time complexity) is independent of the number of sensor nodes. The same technique can be applied to obtain the maximum value (MAX).

An additional, and more elaborated, application of prioritized MAC protocols is exercised throughout the paper. In fact, it is often desired to know how physical quantities (such as temperature) vary over an area. Clearly the physical location of each node must be known then. For such systems, we propose an algorithm that produces an interpolation of the sensor data as a function of space coordinates. This interpolation is a compact representation of sensor data at a moment and it can be obtained efficiently.

We consider this result to be significant because (i) often networks of nodes that take sensor readings are designed to be large scale, dense networks and it is exactly for such scenarios that our algorithms excel and (ii) the techniques that we use exploit structural features that are available in Commercial Off-The-Shelf (COTS) technology. Our solution takes advantage of a prioritized MAC protocol that supports a very large range of priority levels and is collision-free given that priorities are unique. The widely used CAN bus [2] is a possible base protocol that supports the requirements for the algorithms presented in this paper. We have implemented and validated our protocol using

Commercial Off-The-Shelf (COTS) embedded computing platforms with CAN interfaces[1].

The remainder of this paper is structured as follows. Section II gives an application background and the main idea of how a prioritized MAC protocol can be exploited for obtaining aggregated quantities. It also overviews the system model used throughout the rest of the paper. The algorithm for efficiently obtaining an interpolation of measurements is then formally presented in Section III, whereas an implementation of the algorithm and experimental evaluation is presented in Section IV. The ability of previous work to solve the problem addressed in this paper is discussed in Section V. Finally, in Section VI, conclusions are drawn.

II. Preliminaries and Motivation

A networked control system often monitors and controls a physical quantity that is spatially distributed. For example, (i) paper machines need to keep the paper equal in thickness over an area, (ii) steel processes need to keep the temperature according to a specification over the workpiece and, (iii) in large-scale garages and tunnels, distributed control of fans for air refreshment can be optimized if a model of the measured distribution of CO or CO_2 is available (Examples of such applications can be found at [3]). In order to get an accurate image of the physical phenomenon it is required that the sensor network is dense, which causes a large number of sensors to be deployed.

Figure 1 presents the system architecture of the network-based control system we consider. A large number of sensor nodes take measurements and they are fed to a sensor recording unit. The sensor recording unit delivers a representation of all sensor measurements to the real-time controller which acts on the physical environment. It may seem natural to let the sensor recording unit request measurements from all sensors but for large-scale sensor-systems such an approach takes a long time, something that leads to a long sampling period and a large controller delay. For this reason, it is worthwhile to find a better approach.

We will discuss an approach where the sensor recording unit obtains a compact approximate representation of the measurements; an interpolation of all measurements is suitable for this purpose. An efficient algorithm for obtaining such an interpolation will be designed. In order to build up the intuition for understanding that algorithm, we will discuss efficient data-aggregation in general in the following subsection.

[1]Source code and description of tests performed with the implementation can be found at http://www.hurray.isep.ipp.pt/activities/WISE-CAN/

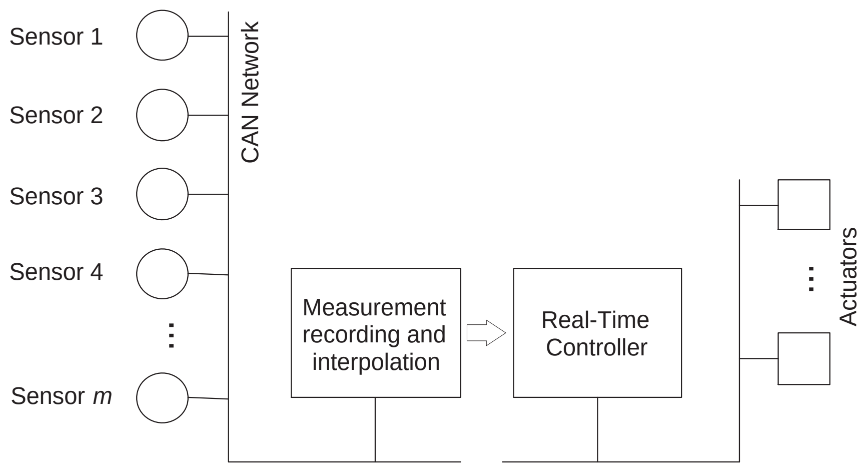


Fig. 1. The system architecture of the network-based control system we consider.

The basic premise for this work is the use of a prioritized MAC protocol. This implies that the MAC protocol assures that out of all nodes contending for the medium at a given moment, the one(s) with the highest priority gain access to it. This is inspired by Dominance/Binary-Countdown protocols [4]. In such protocols, messages are assigned unique priorities, and before nodes try to transmit they perform a contention resolution phase named arbitration such that the node trying to transmit the highest-priority message succeeds.

During the arbitration (depicted in Figure 2a), each node sends the message priority bit-by-bit, starting with the most significant one, while simultaneously monitoring the medium. The medium must be devised in such a way that nodes will only detect a ”1” value if no other node is transmitting a ”0”. Otherwise, every node detects a ”0” value regardless of what the node itself is sending. For this reason, a ”0” is said to be a dominant bit, while a ”1” is said to be a recessive bit. Therefore, low numbers in the priority field of a message represent high priorities. If a node contends with a recessive bit but hears a dominant bit, then it will refrain from transmitting any further bits, and will proceed only monitoring the medium. Finally, exactly one node reaches the end of the arbitration phase, and this node (the winning node) proceeds with transmitting the data part of the message. As a result of the contention for the medium, all participating nodes will have knowledge of the winner’s priority.

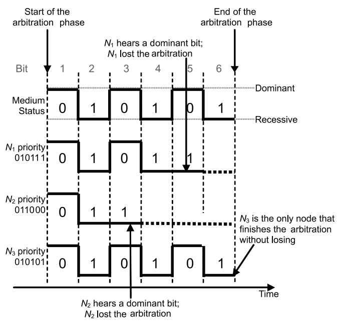
The CAN bus [2] is an example of a technology that offers such a MAC behavior. It is used in a wide range of applications, ranging from vehicles to factory automation (the reader is referred to [5] for more examples of application fields and figures about the wide adoption of CAN). Its wide application fostered the development of robust error detection and fault confinement mechanisms, while at the same time maintaining its cost effectiveness. An interesting feature of CAN is that the maximum length of a bus can be traded-off for lower data rates. It is possible to have a CAN bus with a bit rate of 1Mbit/s for a maximum bus length of 30 meters, or a bus 1000 meters long (with no repeaters) using a bit rate of 50 Kbit/s. While the typical number of nodes in a CAN bus is usually smaller than 100, with careful design (selecting appropriate bus-line cross section, drop line length and quality of couplers, wires and transceivers) of the network it is possible to go well above this value. For example, CAN networks with more than a thousand nodes have been deployed and they operate in a single broadcast domain (such networks have been built; see for example [6]).

The focus of this paper will be on exploiting a prioritized MAC protocol for efficiently obtaining data from distributed sensors. We propose distributed algorithms that can directly be applied to wired CAN networks, a well established and disseminated technology widely used in systems that incorporate a large number of nodes that take sensor readings. The use of such a prioritized MAC protocol is proposed to be in a way that priorities are dynamically established during runtime as a function of the sensed values involved in the specific distributed computation. We show that such a MAC protocol enables efficient distributed computations of aggregated quantities in networks composed of many embedded nodes.

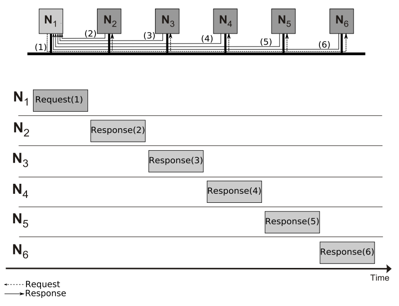
A. The Main Idea

The problem of obtaining aggregated quantities in a single broadcast domain can be solved with a naïve algorithm: every node broadcasts its sensor reading sequentially. Hence, all nodes know all sensor readings and then they can obtain the aggregated quantity. This has the drawback that in a broadcast domain with m nodes, at least m broadcasts are required to be performed. Considering a network designed for $m \geq 100$, the naïve approach can be inefficient; it causes a large delay.

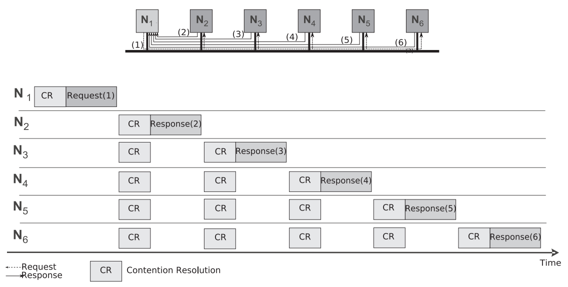
Let us consider the simple application scenario as depicted in Figure 2b, where a node (node N_1) needs to know the minimum (MIN) temperature reading among its neighbors. Let us assume that no other node attempts to access the medium before this node. A naïve approach would imply that N_1 broadcasts a request to all its neighbors and then N_1 would wait for the corresponding replies from all of them. As a simplification, assume that nodes orderly access the medium in a time division multiple access (TDMA) fashion, and that the initiator node knows the number of neighbor nodes. Then, N_1 can derive a waiting timeout for replies based on this knowledge. Clearly,


Start of the arbitration phase
End of the arbitration phase
N_1 hears a dominant bit; N_1 lost the arbitration
Bit 1 2 3 4 5 6
Dominant
Medium Status 0 1 0 1 0 1
Recessive
N_1 priority 010111 0 1 0 1 1
N_2 priority 011000 0 1 1
N_3 is the only node that finishes the arbitration without losing
N_3 priority 010101 0 1 0 1 0 1
Time
N_2 hears a dominant bit; N_2 lost the arbitration

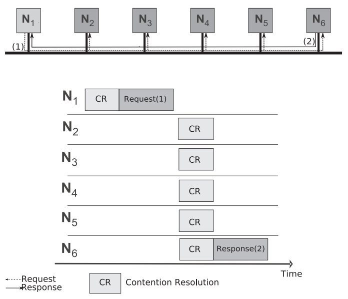
(a) CAN arbitration



(b) Naïve algorithm (TDMA-like MAC)



(c) Naïve algorithm (CAN-like MAC)



(d) Novel algorithm (CAN-like MAC)

Fig. 2. Dominance/Binary-Countdown Arbitration Motivating Examples. (a) Example of bitwise arbitration; (b) example application where N_1 needs to know the minimum (MIN) temperature reading among its neighbors (N_2 to N_6); (c) possible solution for the example application using a CAN-like MAC, using fixed priorities for the messages; (d) possible solution for the example application exploiting the properties of a CAN-like MAC, where priorities are assigned at runtime according to the sensed values.

with this approach, the execution time depends on the number of neighbor nodes (m). Figure 2c depicts another naïve approach, but using a CAN-like MAC protocol.

Assume in that case that the priorities the nodes use to access the medium are ordered according to the nodes' ID, and are statically defined prior to runtime. Note that in order to send a message, nodes have to perform arbitration before accessing the medium. When a node wins it sends its response and stops trying to access the medium. It is clear that using a naïve approach with CAN brings no timing advantages as compared to the other naïve solution (Figure 2b).

Consider now that instead of using their priorities to access the medium, nodes use the value of its sensor reading as priority. Assume that the range of the analog to digital converters (ADC) on the nodes is known, and that the MAC protocol can, at least, represent as many priority levels This assumption typically holds since ADC tend to have a data width of 8,10,12 or 16-bit while the CAN bus offers up to 29 priority bits. This alternative would allow an approach as depicted in Figure 2(d). With such an approach, to obtain the minimum temperature among its neighbors, node N_1 needs to perform a broadcast request that will trigger all its neighbors to contend for the medium using the prioritized MAC protocol. If neighbors access the medium using the value of their temperature reading as the priority, the priority winning the contention for the medium will be the minimum temperature reading. With this scheme, more than one node can win the contention for the medium. But, considering that at the end of the arbitration the priority of the winner is known to all nodes, no more information needs to be transmitted by the winning node. In this scenario, the time to obtain the minimum temperature reading only depends on the time to perform the contention for the medium, not on m. If, for example, one wishes that the winning node transmits information (such as its location) in the data packet, then one can code the priority of the nodes by adding a unique number (for example, the node ID) in the least significant bits, such that priorities will be unique. Such use, results in a time complexity of log m.

A similar approach can be used to obtain the maximum (MAX) temperature reading. In that case, instead of directly coding the priority with the temperature reading, nodes will use the bitwise negation of the temperature reading as the priority. Upon completion of the medium access contention, given the winning priority, nodes perform bitwise negation again to know the maximum temperature value.

MIN and MAX are just two simple and pretty much obvious examples of how aggregate quantities can be obtained with a minimum message complexity (and therefore time complexity) if message priorities are dynamically assigned at runtime upon the values of the sensed quantity. In Section III we will introduce a more complex aggregated quantity that can also be efficiently obtained by using such a MAC approach.

B. System Model

The network consists of m nodes that take sensor readings where a node is given a unique identifier in the range 1..m. MAXNNODES denotes an upper bound on m and we assume that MAXNNODES is known by the designer of the control system before run-time. Nodes do not have a shared memory and all data variables are local to each node.

Each node has a transceiver and is able to transmit to or receive from a single channel. Every node has an implementation of a prioritized MAC protocol with the characteristics as described earlier. Nodes perform requests to transmit, and each transmission request has an associated priority. Priorities are integers in the range [0, $MAXP$], where lower numbers correspond to higher priorities. Let $N_{PRIOBITS}$ denote the number of priority bits. This parameter has the same value for all nodes. Since N_{PRIOBITS} is used to denote the number of bits used to represent the priority, the priority is a number in the range of 0 to $2^{NPRIOBITS}-1$. Clearly, $MAXP = 2^{NPRIOBITS}-1$.

A node can request to transmit an *empty packet*; that is, a node can request to the MAC protocol to perform the contention for the medium, but not send any data. This is clarified later in this section. All nodes share a single reliable broadcast domain.

A program on a node can access the communication system via the following interface. The `send` system call takes two parameters, one describing the priority of the packet and another one describing the data to be transmitted. If a node calling `send` wins the contention, then it transmits its packet and the program making the call unblocks. If a node calling `send` loses the contention, then it waits until the contention resolution phase has finished and the winner has transmitted its packet (assuming that the winner did not send an empty packet). Then, the node contends for the channel again. The system call `send` blocks until it has won the contention and transmitted a packet. The function `send_empty` takes only one parameter, which is a priority and causes the node only to perform the contention but not to send any data after the contention. In addition, when the contention is over (regardless of whether the node wins or loses), the function `send_empty` gives the control back to the application and returns the priority of the winner.

The system call `send_and_rcv` takes two parameters, priority and data to be transmitted. The contention is performed with the given priority and then the data is transmitted if the node wins. Regardless of whether the node wins or loses, the system call returns the priority and data transmitted by the winner and then unblocks the application.

A node N_i takes a sensor reading s_i. It is an integer in the range [0, $MAXS$] and it is assumed that.

III. Interpolation of Sensor Data With Location

Having seen the main idea of how to take advantage of a prioritized MAC protocol, we are now in positiion to present our approach for obtaining an interpolation of measurements. We will do so formally with pseudocode; this pseudocode returns an upper bound on the error of the interpolation. We assume that nodes take sensor readings, but we will also assume that a node N_i knows its location given by two coordinates (x_i,y_i). With this knowledge, it is possible to obtain an interpolation of sensor data over space. This offers a compact representation of the sensor data and it can be used to compute virtually anything.

We let $f(x,y)$ denote the function that interpolates the sensor data. Also let e_i denote the magnitude of the error at node N_i; that is:

$$e_i = |s_i - f(x_i, y_i)| \tag{1}$$

and let e denote the global error; that is:

$$e = \max_{i=1..m} e_i \tag{2}$$

The goal is to find $f(x,y)$ that minimizes e subject to the following constraints: (i) the time required for computing f at a specific point should be low; and (ii) the time required to obtain the function $f(x,y)$ from measurements should be low. The latter is motivated by the fact that it is interesting to track physical quantities that change quickly; it may be necessary to update the interpolation periodically in order to track, for example, how the concentration of hazardous gases move. For this reason, we will use weighted-average interpolation (WAI) [7] (also used in [8], [9]). WAI is defined as follows:

$$f(x,y) = \begin{cases} 0 & \text{if } S = \emptyset; \\ s_i & \text{if } \exists N_i \in S\text{: } x_i = x \wedge y_i = y; \\ \frac{\sum_{i \in S} s_i \cdot w_i(x,y)}{\sum_{i \in S} w_i(x,y)} & \text{otherwise.} \end{cases} \tag{3}$$

where S is a set of nodes used for interpolation. The weights $w_i(x,y)$ are given by:

$$w_i(x,y) = \frac{1}{(x_i - x)^2 + (y_i - y)^2} \tag{4}$$

Intuitively, Equations 3 and 4 state that the interpolated value is a weighted average of all data points in S and the weight is the inverse of the square of the distance. There are many possible choices on how the weight should be computed as a function of distance; the way we have selected is intended to avoid calculations of square root in order to make the execution time small on platforms that lack hardware support for floating point calculations. This is the case for typical sensor network platforms [10]–[12].

The original version [7] of weighted-average interpolation uses all available sensor measurements for interpolation. But this would imply that computing Equation 3 from sensor readings has a time complexity of O(m). Fortunately, it is often the case [13] that sensor readings exhibit spatial locality; that is, nodes that are close in space give similar sensor readings. For this reason, the interpolation will offer a low error even if only a small number of carefully selected nodes are in S.

Algorithm 1 Finding a subset of nodes to be used in WAI

```
Require: All nodes start Algorithm 1 simultaneously.
Require: $k$ denotes the desired number of interpolation points.
Require: A node $N_i$ knows $x_i$,$y_i$ and $s_i$.
Require: (MAXS+1) $\times$ (MAXNNODES+1) + MAXNNODES $\leq$ MAXP.
 1: function find_nodes() return a set of packets
 2:     S $\leftarrow \emptyset$
 3:     for j $\leftarrow$ 1 to k do
 4:         Calculate f($x_i$,$y_i$) in Equation 3 and assign it to the variable "myinterpolatedvalue"
 5:         error $\leftarrow$ abs( $s_i$ - to_integer(myinterpolatedvalue) )
 6:         temp_prio $\leftarrow$ error $\times$ (MAXNNODES + 1) + i
 7:         prio $\leftarrow$ (MAXP+1) - temp_prio
 8:         snd_pack $\leftarrow <s_i,x_i,y_i>$
 9:         $<$winning_prio, rcv_pack$>$ $\leftarrow$ send_and_rcv( prio, snd_pack)
10:         S $\leftarrow$ S $\bigcup$ rcv_pack
11:     end for
12:     return $S$
13: end function
```

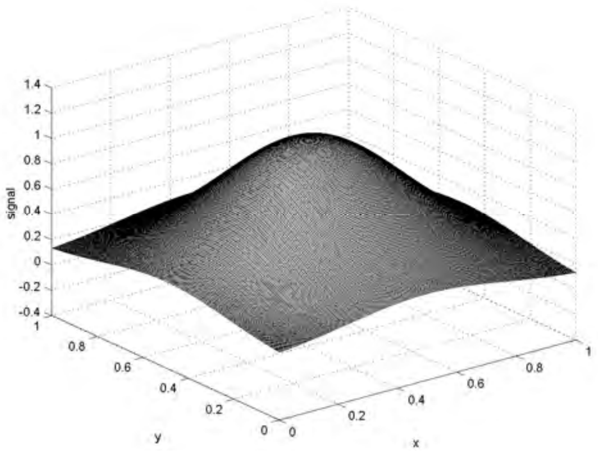
Hence, the goal is now to find those nodes that contribute to producing a low error in the interpolation as given by Equation 3. We select a number of k nodes that constitute the interpolation, where k is a parameter of the algorithm that will control the accuracy of the interpolation. Recall that a prioritized MAC protocol can find the maximum among sensor readings. We can exploit this feature to find k nodes that offer a low value of the error. For this, the proposed distributed algorithm starts with an interpolation being a flat surface and then performs k iterations, where at each iteration the node with largest magnitude of the error between its sensor reading and the interpolated value will be the winner of the contention.

Algorithm 1 is designed based on this principle. It computes (on line 5) the error. This error is concatenated with the identifier of the node (together this forms the priority of the message) ensuring that all priorities are unique. All nodes send their messages in parallel (on line 9) and exactly one will win the contention. Recall from Section II-B that when nodes call `send_and_rcv`, then both the priority of the winner and the data transmitted by the winner are returned to the application on every node. This packet is added (on line 10) to the set S, which keeps track of all received packets related to the problem of creating an interpolation.

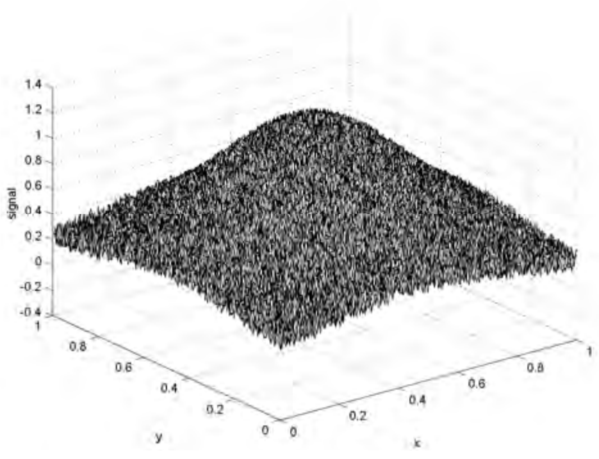
Figures 3 and 5 illustrate the operation of our interpolation scheme[2]. It can be seen that the interpolation result is smooth and that it tracks well the original signal. However, performing weighted-average interpolation with 6 randomly selected nodes gives poor interpolation. This is illustrated in Figure 3d.

Figure 4 illustrates another signal. It can be seen that our interpolation technique also works well for this signal. The quality of the interpolation clearly depends on the characteristics of the signal however. In Appendix A, other

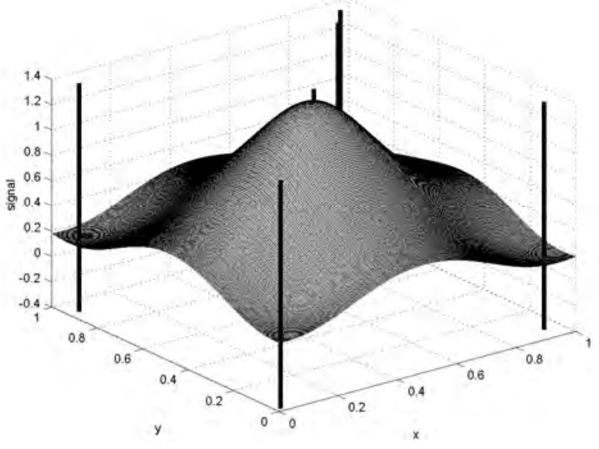
[2]Code and simulation results are available at http://www.hurray.isep.ipp.pt/activities/WISE-CAN/



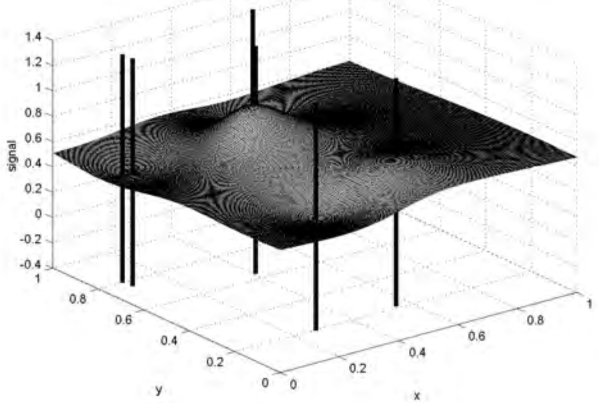
(a) Original Signal



(b) Original Signal with Noise



(c) WAI w/ Carefully Selected Points

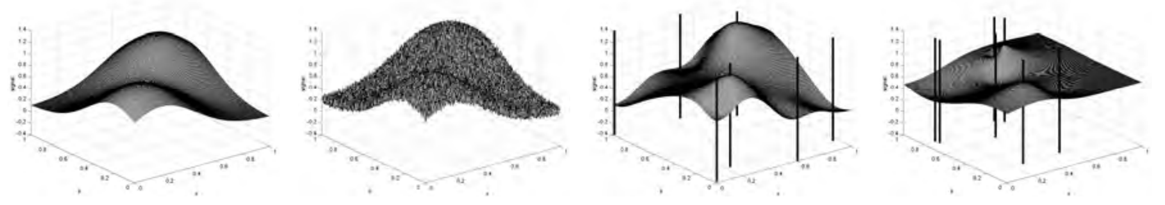


(d) WAI w/ Randomly Selected Points

Fig. 3. Interpolation Example 1. (a) the original signal that varies over space; (b) noise added to the original signal; (c) the result of the interpolation given by our algorithm. The location of the subset of $k = 6$ nodes that were selected to be used in the interpolation is indicated with vertical lines; (d) example result of the interpolation when nodes are selected randomly.

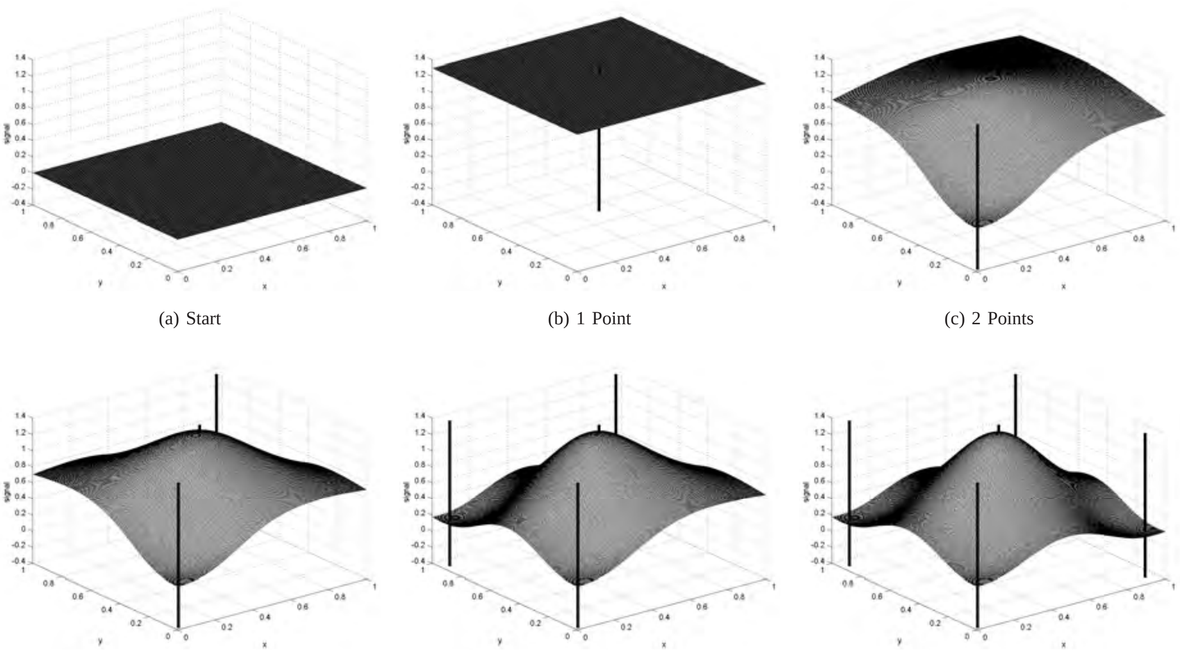
examples that illustrate the performance of our scheme can be found. These examples show that our interpolation technique performs well as long as the signal does not change too abruptly. This is often the case of signals that describe physical phenomena like temperature, light, or dispersion of a gas.

The choice k will also affect the quality of the interpolation. The value of k depends on the distribution of the signal in space. In general, a higher value of k offers a smaller error. The application developer needs to select a value of k that is a function of the number of the interesting points typically found in the physical phenomenon. More details on the the selection of k can be found in Appendix B.



(a) Original Signal (b) Original Signal with Noise (c) WAI w/ Carefully Selected Points (d) WAI w/ Randomly Selected Points

Fig. 4. Interpolation Example 2. (a) the original signal that varies over space; (b) noise added to the original signal; (c) the result of the interpolation given by our algorithm. The location of the subset of $k = 6$ nodes that were selected to be used in the interpolation is indicated with vertical lines; (d) example result of the interpolation when nodes are selected randomly.



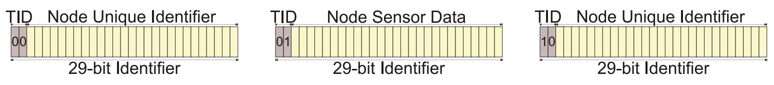
(a) Start (b) 1 Point (c) 2 Points

(d) 3 Points (e) 4 Points (f) 5 Points

Fig. 5. Iterations concerning interpolation Example 1. (a) at the beginning, there is no point included in S. Therefore, from the definition of $f(x,y)$ in Equation 3 it results that $f(x,y)$=0. This gives a plane surface; (b) then, each node calculates the error between its sensor reading and the starting plane surface. This error is used in the contention of the MAC protocol, causing the node with the largest error to win, and thus, it is inserted into S; (c)-(e) nodes proceed similarly, calculating their error to the current interpolated surface and adding the node with the largest error to the interpolation, until the set S has k points; (f) we finally obtain the result of the final iteration, for $k = 6$.

IV. Implementation and Experimental Evaluation

In this section we will study the performance and feasibility of our approach. In Section A, we provide an overview of an implementation of our interpolation scheme; whereas Section B provides the detailed description of the implementation. Section C demonstrates the superior performance of the approach and finally, in Section D,



(a) TID=00 (b) TID=01 (c) TID=10

Fig. 6. CAN Extended 29-bit Message Identifier Segmentation. The 29-bit CAN message identifier is segmented in two parts. Two bits are used for the TID and the 27 remaining for the node's identifier. The TID is the protocol type identifier and can be one of three possibilities. (a) request to start the computation of the interpolation (binary value 00); (b) reply with the node's sensor data (binary value 01) and (c) any other messages that do not pertain to the computation of aggregated quantities (binary value 10). Notice that the values of the TID are such that the request to start the computation is always the highest-priority message in the system, the reply with the node's sensor data the second highest priority message and other messages in the system have lower priority.

we showcase a test-bed application in order to demonstrate that the interpolation scheme works in practice.

A. Overview of Implementation

To evaluate the feasibility of the execution of our algorithm in real-life setups, we implemented Algorithm 1 on an embedded computer platform.

1) The Platform: The embedded computer platform has an Atmel AVR AT90CAN128 microcontroller that can run at a clock speed up to 16 MHz. This microcontroller provides a CAN controller compliant with V2.0A and V2.0B CAN standards. Additionally, our platform also provides luminance and temperature sensors.

2) Protocol: To this point, we have assumed that all nodes start the execution of the protocol simultaneously. In practice, we have to arrange a protocol that achieves this. Our approach uses a master node (which is elected dynamically, as we will see) that issues a request to compute the aggregated quantity. This request is broadcasted to all nodes and provides a time reference. Then, by analyzing the worst-case execution time of the interpolation algorithm, we can set up a timer in each node that defines the time when each iteration of the algorithm starts, so that all nodes start each iteration at the same time.

In CAN, the priority of the message is transmitted in the message identifier. Because our implementation uses the extended frame format, as defined by the CAN 2.0 specification [2], this identifier is called the *extended message identifier*. We have segmented the CAN extended message identifier as depicted in Figure 6.

Two bits of the 29-bit identifier are reserved for the protocol type identifier (TID), and the remaining 27 bits are free to be used for the encoding of the node's message identifier and measurement value. The TID is used to identify the type of frame being transmitted. According to the CAN specification [2], the 7 most significant bits of the identifier cannot be all recessive, thus we cannot use the binary value 11 for the TID.

Any node may request to start the computation of the aggregated quantity at any time when no computation of the aggregated quantity is taking place. The node encodes the message priorities according to Figure 6a, with TID=00 and using its unique identifier. Because nodes will not send a request to start the computation of the aggregated quantity after this message is transmitted, this will ensure that only one node initiates the computation, and thus becomes the master (mutual exclusion between nodes that perform the request at the same time is guaranteed by the CAN arbitration).

After the reception of a request to compute the aggregated quantity, all nodes use this as a time reference that will trigger each iteration of the algorithm, such that all nodes perform the iterations at the same time.

B. Details of Implementation

Algorithm 2 gives detailed pseudocode that corresponds to our implementation[3]. The types and variables are declared on lines 1-13. Observe that only integer variables are used; `uint16` means an unsigned integer of size 16 bits. `uint32` and `uint64` are defined analogously.

Lines 15-17 is a simple event handler that executes the code when an application request that an interpolation should be computed. It sends a message requesting that an interpolation should be computed (with TID=0) and when this message is received the event handler of lines 19-24 is executed. The time is read (on line 20) and it provides all nodes with a common reference point in time. It is checked if the message is of the type that an interpolation should be created and then calls the function `find_nodes()`, which is implemented on lines 26-57.

The function `find_nodes()` is basically an optimized version and more elaborate description of Algorithm 1. It is optimized in two ways. First, no floating-point operations are used and this offers approximately a factor of 10 in improvement in speed. Second, each computer node needs to compute the interpolation at itself in order to know the error. One can see that partial sums in the third case in Equation 3, can be maintained during the iterations and hence it is not necessary for a computer node to compute the interpolated value at itself from scratch. This offers approximately a factor of k in improvement in speed.

Observe the delay until statement on line 36 ensures that the execution is delayed until the time given as parameter and this ensures that in each of the iterations, all computer nodes start the tournament at the same time. It depends

[3]The source-code of this implementation is available at http://www.hurray.isep.ipp.pt/activities/WISE-CAN/

on two parameters $TIMEA$ and $TIMEB$. These two parameters are set according to the worst-case execution time of the algorithm on the sensor nodes and are independent of the number of nodes. In case of a change of the hardware configuration of one or several nodes, the changed nodes may have a different timing as long as the worst-case assumptions hold. According to our measurements, reported Section IV-C, we let $TIMEA = 0.2$ ms and $TIMEB = 0.75$ ms.

The notation `((uint64) 1)` $<<$ `32` means *convert the value one to a 64-bit integer and shift the value 32 bits to the left* and it is used to obtain fixed-point operations on 64 bit numbers such that 32 bits represent a fractional part. All timing parameters assume that a CAN bus of speed 100 Kbit is used.

C. Experimental Evaluation

We use our implementation presented in Algorithm 2 for the experimental evaluation. As already pointed out, we can see that the time to execute the computations should be constant over the several iterations. This was observed in practice, and the worst-case execution time measured for each iteration was $TIMEB = 0.75$ ms.

The worst-case execution time for each iteration of the loop in Algorithm 2 includes the worst-case execution time of the computations performed in each iteration and also the time to transmit a message on the CAN bus:

$$C_{iteration} = TIMEB + C_{packet\_tx} \tag{5}$$

where C_{packet_tx} is the time to perform an arbitration and transmit a packet on the CAN bus, and is given by the equation in [14] for 29-bit identifiers, that accounts for the worst-case bit stuffing in the CAN bus:

$$C_{packet\_tx} = (80 + 10 \times size_m) \times \tau_{bit} \tag{6}$$

where $size_m$ is the number of data bytes in the message and τ_{bit} is the transmission time for a single bit.

Finally, we can see that the execution time of our algorithm is given by:

$$C_{interpolation} = TIMEA + k \times C_{iteration} \tag{7}$$

where k is the number of nodes that contribute the most for the interpolation.

We can now instantiate these equations for our system. We have selected a data rate of 100 Kbit per second for the CAN bus (target systems of our algorithm will typically have a bus length $\geq$ 1000 m, thus a higher bit rate should typically not be applicable) and a data payload of 6 bytes (2 bytes for the x coordinate, 2 bytes for the y

Algorithm 2 Implementation details of an algorithm for finding a subset of nodes to be used in WAI

```
Require: (MAXS+1) × (MAXNNODES+1) + MAXNNODES ≤ MAXP.
Require: TIMEA is the maximum time required for executing from the reception of a message with TID=0 until the execution reaches line 36.
Require: TIMEB is the maximum time required for executing from line 37 until line 36 in the next iteration of the loop is just about to be
    executed.
 1: type interpolation_pack = record
 2:         sensor_reading : uint16
 3:         x : uint16
 4:         y : uint16
 5:     end record
 6: prio, winning_prio, temp_prio, temp_prio_MAC_order: uint32
 7: snd_pack, rcv_pack : interpolation_pack
 8: S : set
 9: t : time
10: max_error, error, myinterpolatedvalue : uint16
11: dx, dy, sq : uint32
12: num, denom, temp : uint64
13: update_myinterpolation : boolean
14:
15: when a node requests that an interpolation should be computed do
16:     prio ← i
17:     send_empty( prio)
18:
19: when a node receives a message of length 0 or more do
20:     t ← read_current_time
21:     ti ← TID of the received message
22:     if ti=0 then
23:         find_nodes()
24:     end if
25:
26: function find_nodes() return a set of packets and the error of the interpolation
27:     myinterpolatedvalue ← num ← denom ← 0
28:     S ← ∅
29:     update_myinterpolation ← TRUE
30:     for j ← 1 to k do
31:         error ← abs( $s_i$ - myinterpolatedvalue )
32:         temp_prio ← ((uint32) error) × (MAXNNODES + 1) + i
33:         temp_prio_MAC_order ← ( (1 << 27) - 1) - temp_prio
34:         prio ← (1 << 27) + temp_prio_MAC_order
35:         snd_pack ← <$s_i$,$x_i$,$y_i$>
36:         delay until t+TIMEA+TIMEB*(j-1)
37:         <winning_prio, rcv_pack> ← send_and_rcv( prio, snd_pack)
38:         if winning_prio = prio then
39:             update_myinterpolation ← FALSE
40:             myinterpolatedvalue ← $s_i$
41:         end if
42:         if update_myinterpolation = TRUE then
43:             Give a command to the CAN controller to abort the message that was requested to be transmitted
44:             dx ← $x_i$ - recv_pack.x
45:             dy ← $y_i$ - recv_pack.y
46:             sq ← dx*dx + dy*dy
47:             temp ← ((uint64) recv_pack.value) << 32
48:             num ← num + temp div sq
49:             temp ← ((uint64) 1) << 32
50:             denom ← denom + temp div sq
51:             myinterpolatedvalue ← num div denom
52:         end if
53:         $S \leftarrow S \bigcup rec\_packet$
54:     end for
55:     max_error ← (winning_prio bitwiseand MASKOUT_TID) div (MAXNNODES + 1)
56:     return (S,max_error)
57: end function
```

coordinate and 2 bytes for the sensor value) $C_{packet\_tx} = (80 + 10 \times 6) \times 0.01 = 1.40$ ms. Considering $m = 100$ nodes, $k = 6$, and $TIMEA = 0.20$ ms we have that $C_{interpolation} = 0.20 + 6 \times 2.15 = 13.10$ ms.

We can compare this with the execution of a naïve algorithm as depicted in Figure 2c. The execution time of the naïve algorithm depicted in Figure 2c is given by:

$$C_{naive} = m \times C_{packet\_tx} \tag{8}$$

where m is the number of nodes and C_{packet_tx} is the time to transmit a message with the data point record (containing 2 bytes for the x coordinate, 2 bytes for the y coordinate and 2 bytes for the sensor value).

Using a naïve algorithm like the one depicted in Figure 2c with the same parameters, we have $C_{naive} = 100 \times 1.40 = 140$ ms.

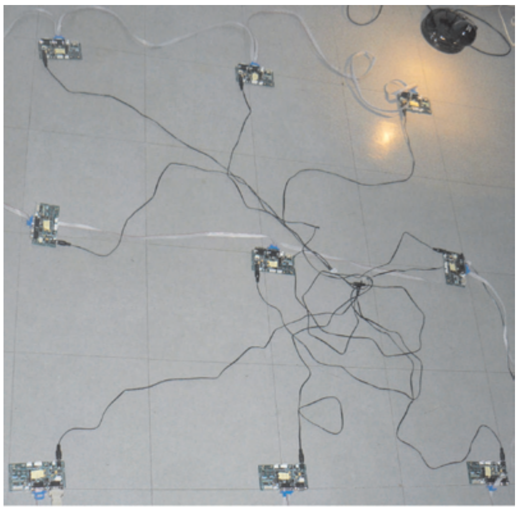
Note that, with our scheme, $C_{interpolation}$ will be constant for any number of nodes m, while with other approaches, the execution time will always increase as m increases. It is our belief that our new interpolation scheme is a fundamental building block for distributed control systems with large number of sensor nodes. The full advantage of our scheme is realized in control systems of thousands of sensor nodes that must collaborate within tight timing constraints. Such systems already exist today, and it is expected that in the near future, building such systems will be common practice.

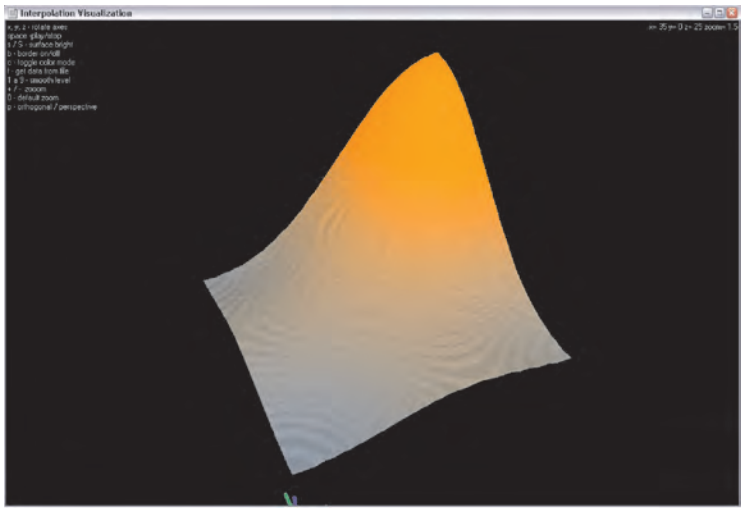
D. Demonstration of the Interpolation

We will now present an application in order to showcase a possible use of our protocol. The purpose of this application is twofold. We validate the implementation, showing a working application that employs our interpolation and exercise the use of our interpolation scheme using COTS hardware.

For our application, we were limited to the sensors available in our embedded computer platform, and thus we have used the luminance sensors available because these sensors react fast to changes in the environment. The application consists in a set of nodes, placed at fixed positions. We positioned nine nodes in a grid of 2.4×2.4 meters with equal spacing between them. The nodes were installed in a low light environment and we make a light source move inside the area were the nodes are installed. One of these nodes is connected to a desktop computer and acts as the interface between the network of sensors and a visualization application.

At pre-runtime, nodes are statically assigned their coordinates in an x and y plan, then nodes perform an interpolation with $k = 6$ repeatedly. Every time a new interpolation is made, the node connected to the desktop computer sends the set of points selected to a visualization application via the serial port. Based on the interpolation points received, this application then constructs a visual representation of the interpolation in three dimensions.


(a) Node Placement and Light Source



(b) Interpolation Result

Fig. 7. Demonstration application setup. (a) node placement with light source in upper right corner. (b) visualization application depicting actual result from the setup in Figure 7a.

Figure 7 depicts the setup of the nodes for our demonstration application and a snapshot of the visualization application in the desktop computer. Figure 7a also shows a light source placed in the upper right corner, and Figure 7b shows the result of the interpolation given by this setup using our visualization application.

V. Previous and Related Work

To our knowledge, there exists no previous work that is directly related to our approach in the way that the MAC layer of the CAN protocol is used to support an efficient spatial interpolation scheme. There is, however, interesting related work regarding data aggregation, message scheduling in CAN and wireless networks.

Using medium access control to avoid the transmission of redundant sensor measurements has been suggested in several related work. In previous work, we showed how aggregated values of sensor data relating to the same physical entity can be efficiently computed by employing a priority-based MAC protocol [15], [16]. Most other approaches on data aggregation are based on an aggregate-and-forward architecture in multi-hop wireless sensor networks. An overview on such techniques can be found in [17]. However, since these approaches do not benefit from using the MAC layer in order to avoid sending redundant sensor data, they are less efficient in terms of time and message complexity.

The idea of using CAN to schedule messages in order of their importance for the system as a whole is exercised by [18], although this work is not designed to scale for large systems and does not consider to refrain from transmitting messages which are already in the scheduling queue, if the data from the message before is sufficiently redundant.

Jamieson et al. [19] suggest a protocol for wireless sensor networks that employs a randomized CSMA protocol with carefully selected contention windows that allow for a spatially-correlated contention of sensor nodes.

Ringwald and Römer introduced a collision-free MAC protocol named BitMAC that supports the computation of aggregated values over a set of networked sensors in [20]. BitMAC represents a more flexible alternative to CAN in the sense of providing an underlying dominance protocol, since it also supports wireless communication, however BitMAC is currently not established in industrial automation.

Popovski et al. [21] and Caccamo et al. [22] use the MAC layer to build local clusters which may aggregate the measurements from several sensors. While this greatly reduces the overall number of messages, the MAC layer is not exploited in a special way within a particular cluster, which gives an message and time complexity of $O(m)$ for the communication within a cluster of m nodes. Note that we use the MAC protocol to directly reduce the number of messages within a single broadcast domain to a few number of samples k that is smaller and possibly independent of m.

Vuran and Akyildiz present a network protocol for wireless systems that uses the MAC layer to select sensor measurements based on spatial correlation [23]. Their approach features a CSMA/CA access scheme using an RTS/CTS/DATA/ACK structure as it can be found in IEEE 802.11 standard [24]. After one node having transmitted, nodes within a so-called correlation radius of the winning node suppress their data transfer in order to enable nodes with lesser correlation to access the medium. In contrast, in our work the MAC access scheme is used in a different way that allows a more fine grained control of the nodes' sending priorities. Moreover, the correlation function in [23] is explicitly based on spatial information, while in our case, the spatial interpolation is created as an intrinsic property of our algorithm.

The problem of finding an appropriate interpolation scheme that models the process state from a set of sensor measurements is attracting increasing attention in the research community. Formally speaking, *interpolation* requires that the function crosses all data points; if it only approximates the data points then it is called *curve-fitting* or *regression*. However, research papers (e.g., [8]) frequently use the term interpolation even if the function does not cross all data points, and thus we decided also to follow this use of names by calling our algorithm an *interpolation algorithm*. While these works are related to our approach, the interpolation solutions proposed in the literature [8], [9], [13] are mainly designed for multi-hop networks, so that their complexity in a single broadcast domain, unlike our algorithm, heavily depends on the number of nodes.

VI. Conclusions

The contributions of this paper are twofold: First, we have depicted how a prioritized MAC protocol can be applied to a multi-sensor array to efficiently compute a model of sensor measurements and interpolations with a small and known maximum deviation between the sensor measurements that are not represented in the model. The presented approach dramatically reduces the time complexity and message complexity in comparison to existing approaches in sensor networks.

The quality of the interpolation clearly depends the characteristics of the signal. Our interpolation technique performs well as long as the signal does not change too abruptly (see Appendix A for details). This is often the case of signals that describe physical phenomena like temperature, light, or dispersion of a gas.

The interpolation technique presented was not designed to deal with sensor faults. However, mechanisms to deal with sensor faults can be introduced. A preliminary idea on how to integrate outlier rejection with the interpolation technique is available in Appendix C. For future research, it is intended to pursue techniques that improve the robustness of the interpolation scheme presented. One such technique, that could be combined with our scheme, is the algorithm known as RANSAC [25].

A second contribution of the paper is the presentation of a prototype implementation on a CAN network, a protocol widely used in the industrial domain. The CAN controller had to be instrumented in a specific way in order achieve the appropriate timing for our approach, but the employed hardware consists of commercial-off-the-shelf components. Thus, existing sensors with a CAN interface could be adjusted to our protocol by a mere software upgrade.

The real-time capabilities that scale even for a very large number of sensors and the fact that our system is based on CAN, makes our approach suitable for many industrial applications. Possible application scenarios include all kind of multi-sensor arrays with real time constraints, for example measuring the structural health using strain sensors in machines, large workpieces, or vehicles. Thus, an interpolation of the current strain in the material provides detailed information in real-time that can be used to control the operation of an industrial machine in extreme operation conditions but still within the safety limit. Furthermore, production machines like rolling mills or paper machines can benefit from a sensor array providing in real-time a spatially distributed measurement of parameters like thickness, heat, or weight that can be used to control or fine-tuning the feedstock advancement and processing. This type of collaborative processing is potentially also useful in adaptive optics in planned extremely large telescopes, sizing up to 100 meters.

REFERENCES

[1] B. Wittenmark, J. Nilsson, and M. Torngren, "Timing problems in real-time control systems," *American Control Conference, 1995. Proceedings of the*, vol. 3, pp. 2000–2004 vol.3, 21-23 Jun 1995.

[2] *CAN Specification, ver. 2.0*, Bosch GmbH, Stuttgart, Germany, 1991.

[3] GreenPeak, ""applications Website section"." [Online]. Available: http://www.greenpeak.com/Application/Applications.html

[4] A. K. Mok and S. Ward, "Distributed broadcast channel access," *Computer Networks*, vol. 3, pp. 327–335, 1979.

[5] (CiA), "CAN in Automation Website." [Online]. Available: http://www.can-cia.org

[6] Kimaldi, ""network of readers Website"." [Online]. Available: http://www.kimaldi.com/kimaldi_eng/productos/lectores_de_tarjetas/red_de_lectores_can/red_de_lectores_ampliacion_de_informacion

[7] D. Shepard, "A two-dimensional interpolation function for irregularly-spaced data," in *Proceedings of the 1968 23rd ACM national conference*, 1968, pp. 517 – 524.

[8] R. Tynan, G. OHare, D. Marsh, and D. OKane, "Interpolation for Wireless Sensor Network Coverage," in *Proceedings of the the Second IEEE Workshop on Embedded Networked Sensors*, 2005, pp. 123– 131.

[9] M. Sharifzadeh and C. Shahabi, "Supporting spatial aggregation in sensor network databases," in *Proceedings of the 12th annual ACM international workshop on Geographic information*, 2004, pp. 166 – 175.

[10] Crossbow, "MICA2 - wireless measurement system product datasheet," http://www.xbow.com/Products/Product_pdf_files/Wireless_pdf/MICA2_Datasheet.pdf.

[11] Crossbow., "MicaZ - wireless measurement system product datasheet," http://www.xbow.com/Products/Product_pdf_files/Wireless_pdf/MICAz_Datasheet.pdf.

[12] J. Polastre, R. Szewczyk, and D. Culler, "Telos: Enabling ultra-low power wireless research," in *Proceedings of the Fourth International Conference on Information Processing in Sensor Networks: Special track on Platform Tools and Design Methods for Network Embedded Sensors (IPSN/SPOTS'05)*. IEEE Computer Society, 2005, pp. 364– 369.

[13] C. Guestrin, P. Bodik, R. Thibaux, M. Paskin, and S. Madden, "Distributed regression: an efficient framework for modeling sensor network data," in *Proceedings of the Third International Conference on Information Processing in Sensor Networks (IPSN04)*, 2004.

[14] R. I. Davis, A. Burns, R. J. Bril, and J. J. Lukkien, "Controller area network (CAN) schedulability analysis: Refuted, revisited and revised," *Journal of Real-Time Systems*, vol. 35, no. 3, pp. 239–272, 2007.

[15] B. Andersson, N. Pereira, and E. Tovar, "A scalable and efficient approach to obtain measurements in can-based control systems," IPP-HURRAY! Research Group, Institute Polytechnic Porto, Porto, Portugal, Tech. Rep. HURRAY-TR-061102, 2006. [Online]. Available: http://www.hurray.isep.ipp.pt/privfiles/tr-hurray-0601102.pdf

[16] ——, "Exploiting a prioritized MAC protocol to efficiently compute min and max in multihop networks," in *5th Workshop on Intelligent Solutions in Embedded Systems (WISES'07)*, Madrid, Spain, 2007.

[17] A. Skordylis, N. Trigoni, and A. Guitton, "A study of approximate data management techniques for sensor networks," in *Proceedings of the fourth Workshop on Intelligent Solutions in Embedded Systems WISES'06*, 2006.

[18] M. Velasco, P. Martí, R. Castañé, J. Guardia, and J. M. Fuertes, "A CAN application profile for control optimization in networked embedded systems," in *32th Annual Conference of the IEEE Industrial Electronics Society (IECON06)*, Paris, France, 2006.

[19] K. Jamieson, H. Balakrishnan, and Y. C. Tay, “Sift: a MAC protocol for event-driven wireless sensor networks,” in *Proceedings of the third European Workshop on Wireless Sensor Networks (EWSN’06)*, 2006, pp. 260–275.

[20] M. Ringwald and K. Römer, “BitMAC: A deterministic, collision-free, and robust MAC protocol for sensor networks,” in *Proceedings of 2nd European Workshop on Wireless Sensor Networks (EWSN 2005)*, Istanbul, Turkey, Jan. 2005, pp. 57–69.

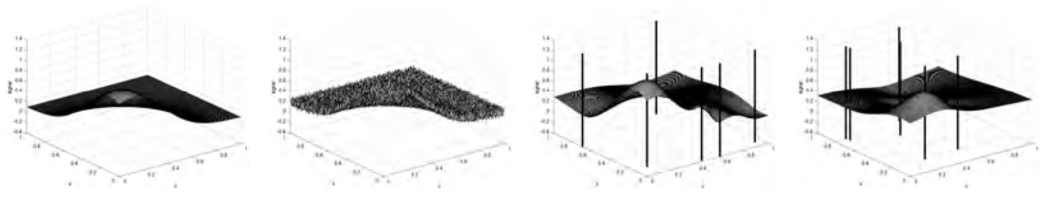
[21] P. Popovski, F. H. P. Fitzek, H. Yomo, T. K. Madsen, and R. Prasad, “MAC-layer approach for cluster-based aggregation in sensor networks,” in *Proceedings of the International Workshop on Wireless Ad-hoc Networks*, Oulu, Finland, May–Jun. 2004, pp. 89–93.

[22] M. Caccamo, L. Y. Zhang, L. Sha, and G. Buttazzo, “An implicit prioritized access protocol for wireless sensor networks,” in *Proceedings of the 23th IEEE Real-Time Systems Symposium (RTSS’02)*, Austin, TX, USA, Dec. 2002, pp. 39–48.

[23] M. C. Vuran and I. F. Akyildiz, “Spatial correlation-based collaborative medium access control in wireless sensor networks,” *IEEE/ACM Transactions on Networks*, vol. 14, no. 2, pp. 316–329, Apr. 2006.

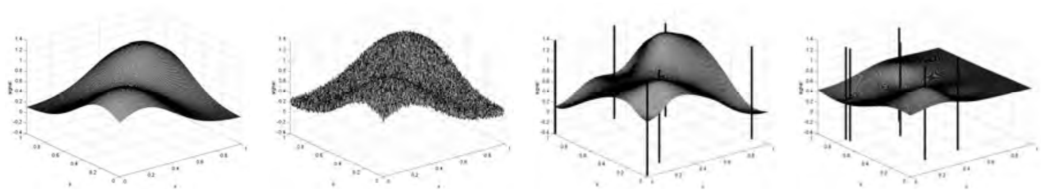
[24] *IEEE Std 802.11, (ISO/IEC 8802-11: 1999) IEEE Standards for Information Technology – Telecommunications and Information Exchange between Systems – Local and Metropolitan Area Network – Specific Requirements – Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications*, Institute of Electrical and Electronics Engineers, Inc., 1999.

[25] M. A. Fischler and R. C. Bolles, “Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography,” *Commun. ACM*, vol. 24, no. 6, pp. 381–395, 1981.



(a) Original Signal (b) Original Signal with Noise (c) WAI with Carefully Selected Data Points (d) WAI with Randomly Selected Data Points

Fig. 8. Interpolation of a plot with a single peak close to the origin.



(a) Original Signal (b) Original Signal with Noise (c) WAI with Carefully Selected Data Points (d) WAI with Randomly Selected Data Points

Fig. 9. Interpolation of a plot with two peaks.

APPENDIX A
PERFORMANCE OF THE INTERPOLATION

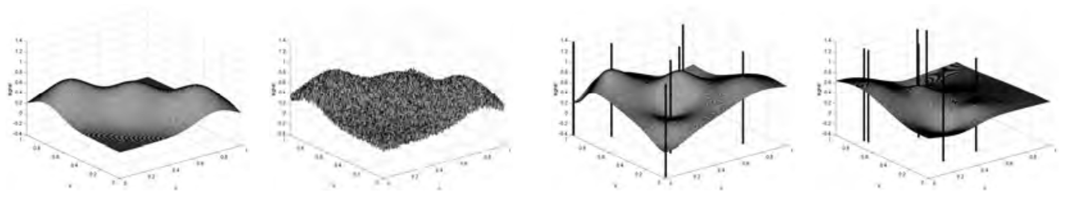
We will now study the operation of our interpolation scheme in more detail. We explore other signal forms and see that our scheme performs well even for them as long as the signal does not change too abruptly when the location changes. Then we explore how our scheme behaves for different values of k.

A. Performance of the Interpolation for Several Example Signals

Figure 8 shows a signal with a single peak located close to the origin. Figure 9 shows a signal with two peaks and Figure 10 presents a signal with three peaks. We can see that our scheme performs well for these examples.

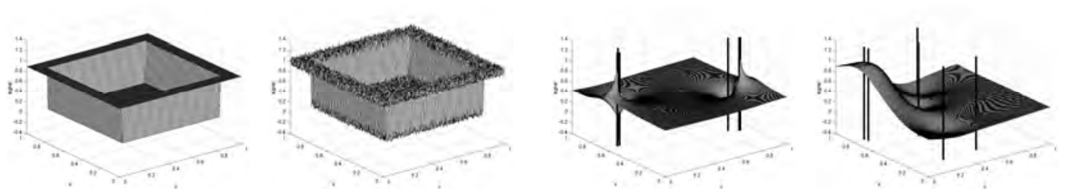
Consider Figure 11. This signal makes abrupt changes. We see that our interpolation scheme does not do a good job and neither does a random scheme. The reason for this is that we are using a smooth signal to interpolate a non-smooth signal. This is a common problem in interpolation. Some knowledge about the signal must be assumed in order to select an appropriate interpolation function.

If the signal changes less abruptly, as illustrated by Figure 12, then our interpolation scheme performs better (although still distant from the original signal). Another more smooth signal gives the same result (Figure 13).



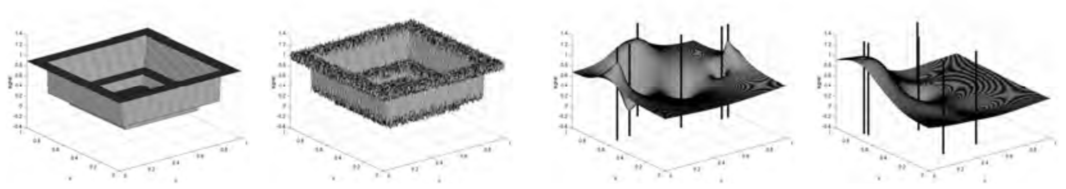
(a) Original Signal (b) Original Signal with Noise (c) WAI with Carefully Selected Data Points (d) WAI with Randomly Selected Data Points

Fig. 10. Interpolation of a plot with three peaks



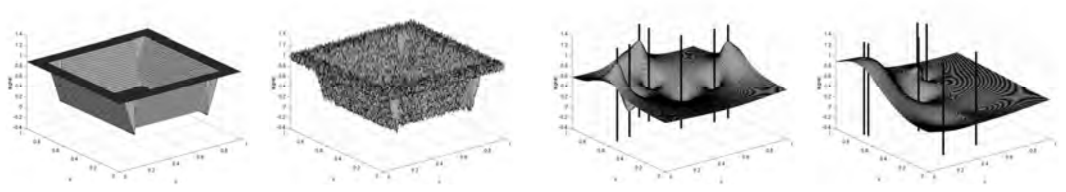
(a) Original Signal (b) Original Signal with Noise (c) WAI with Carefully Selected Data Points (d) WAI with Randomly Selected Data Points

Fig. 11. Interpolation of a plot with a shape of a swimming pool.



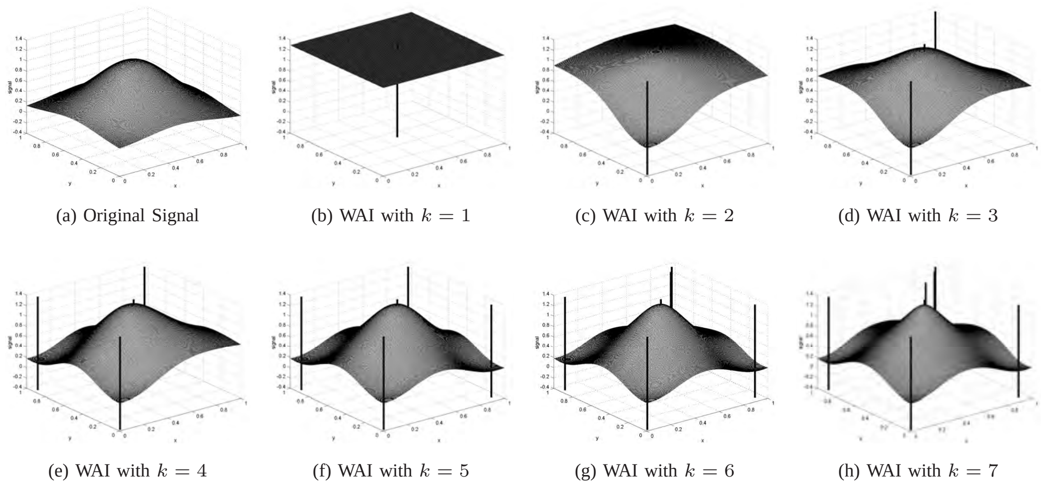
(a) Original Signal (b) Original Signal with Noise (c) WAI with Carefully Selected Data Points (d) WAI with Randomly Selected Data Points

Fig. 12. Interpolation of a plot with a shape of a swimming pool with 2 different levels of depth.



(a) Original Signal (b) Original Signal with Noise (c) WAI with Carefully Selected Data Points (d) WAI with Randomly Selected Data Points

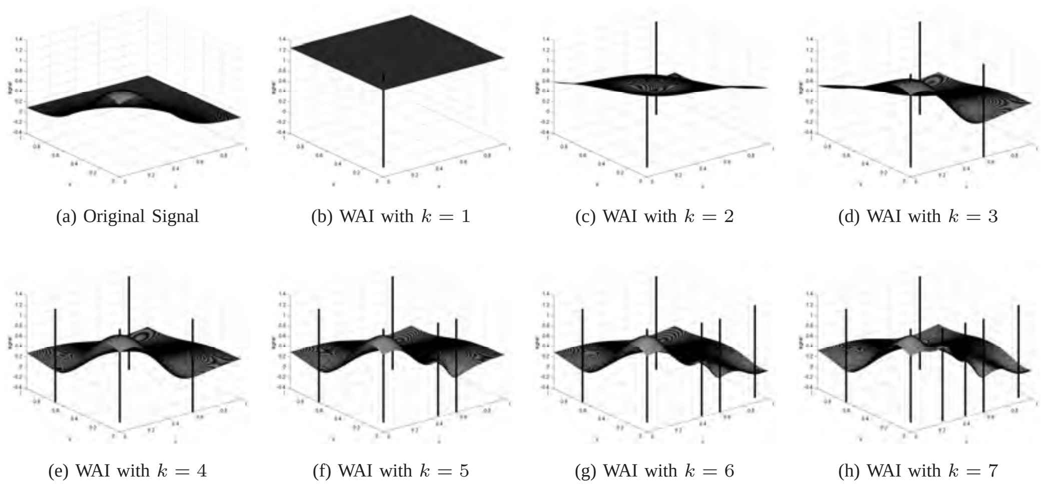
Fig. 13. Interpolation of a plot with a shape of a hot tub.



(a) Original Signal (b) WAI with $k = 1$ (c) WAI with $k = 2$ (d) WAI with $k = 3$

(e) WAI with $k = 4$ (f) WAI with $k = 5$ (g) WAI with $k = 6$ (h) WAI with $k = 7$

Fig. 14. Interpolation of a plot with one peak using k=1 to 7.



(a) Original Signal (b) WAI with $k = 1$ (c) WAI with $k = 2$ (d) WAI with $k = 3$

(e) WAI with $k = 4$ (f) WAI with $k = 5$ (g) WAI with $k = 6$ (h) WAI with $k = 7$

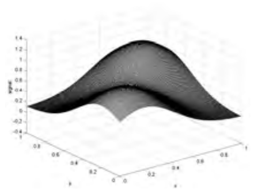
Fig. 15. Interpolation of a plot with one peak close to the origin using k=1 to 7.

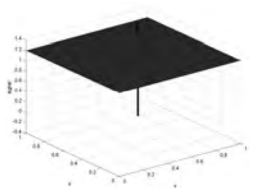
B. Performance of the Interpolation for Several Values of k

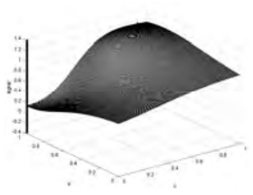
Let us now study how our scheme behaves for different values of k. Figure 14 presents how the interpolation scheme behaves for different values of k, in the presence of a signal with a single peak.

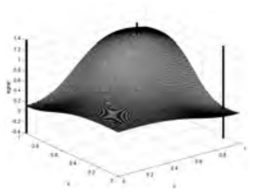
Figure 15 presents how the interpolation scheme behaves for different values of k, in the presence of another signal with a single peak, but close to the origin. Figure 16 presents a signal with two peaks.

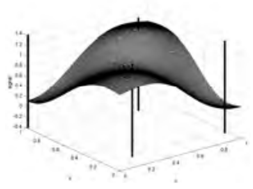
Figure 17 presents how the interpolation scheme behaves for different values of k, in the presence of a signal with three peaks. In order to represent this signal we would need 7 data points (one on each peak and one in each

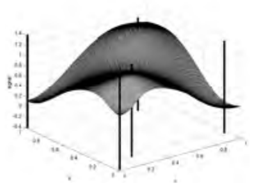

(a) Original Signal

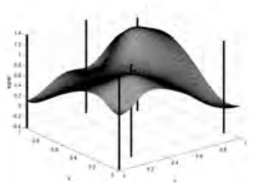

(b) WAI with $k = 1$


(c) WAI with $k = 2$


(d) WAI with $k = 3$


(e) WAI with $k = 4$


(f) WAI with $k = 5$


(g) WAI with $k = 6$

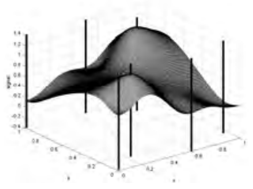
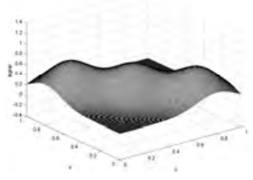
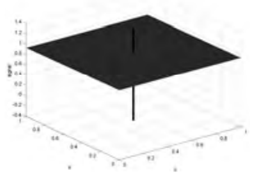
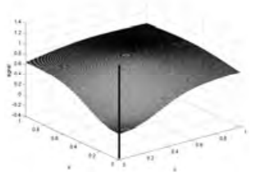
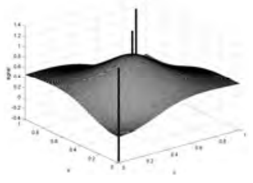

(h) WAI with $k = 7$

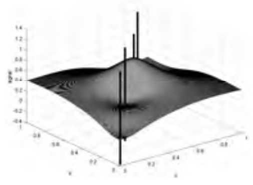
Fig. 16. Interpolation of a plot with two peaks using k=1 to 7.

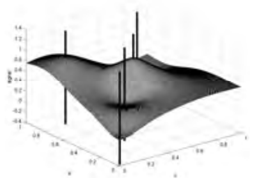

(a) Original Signal

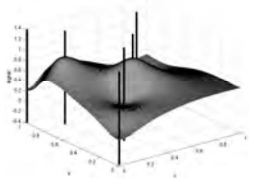

(b) WAI with $k = 1$


(c) WAI with $k = 2$


(d) WAI with $k = 3$


(e) WAI with $k = 4$


(f) WAI with $k = 5$


(g) WAI with $k = 6$

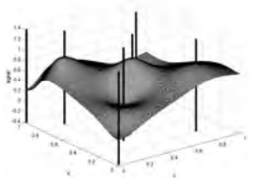

(h) WAI with $k = 7$

Fig. 17. Interpolation of a plot with three peaks using k=1 to 7.

corner). Our interpolation scheme still gives a fair approximation to the signal for only $k = 6$ data points. The case for $k = 7$ is also illustrated in Figure 17 and here our interpolation scheme does a good job. We can conclude that it is important that k is at least as large as the dimensionality of the signal in order for our scheme to do a good job. More details on choosing k are given in Appendix B.

APPENDIX B
SELECTING k

The choice of the value of k clearly depends on the distribution of the signal in space. In general, a higher value of k offers a smaller error. In this Appendix, we discuss the selection of k. Based on some knowledge of the signal, we can define that, as a guideline, k should be selected as:

$$k = N_{min} + N_{max} + N_{saddles} + 2^{dim} \tag{9}$$

where N_{min} is an upper bound on the number of *interesting* points that are local minimizers in the signal; $N_{max}$is an upper bound on the number of *interesting* points that are local maximizers in the signal; $N_{saddles}$ is an upper bound on the number of *interesting* points that are saddle points in the signal and dim is the dimension of the space where the signal is distributed in.

A point is an *interesting local maximizer* if it is a point that the application developer cares about. For example, a signal may have two local maximizers very close to each other. It may be that the physical phenomena is expected to have only one maximizer but because of measurement noise or other physical phenomena that are not considered, two local maximizers exist close to each other in space.

Interesting local minimizers and *saddle points* are defined in the same way.

Consider for example the signal Figure 3a. We can obtain $N_{min} = 0$, $N_{max} = 1$, $N_{saddles} = 0$ and $dim = 2$. This results in $k = 5$, which is enough to get a rough picture of the signal; it tracks all minimizers and maximizers and gives the right picture of the slopes. It can be seen that choosing a larger k gives a smaller error but the benefit is small.

APPENDIX C
DEALING WITH SENSOR FAULTS

Algorithm 3 Finding a subset of nodes to be used in WAI. This algorithm tolerates sensor faults.

```
Require: The same requirements as in Algorithm 1
 1: All nodes run Algorithm 1
 2: for each $N_j \in S$ do
 3:     $SILENT_j \leftarrow$ FALSE
 4: end if
 5: Let $T$ denote an ordered set containing the elements in $S$
 6:     Any order is fine as long as $T$ is the same on all nodes.
 7:     Note that $S$ is the same on all nodes.
 8: for each $N_j \in T$ do
 9:     for each $N_k \in T$ where $N_j$ is earlier than $N_k$ in $T$ do
10:         sqrds := $(s_j - s_k)^2$
11:         sqrdist := $(x_j - x_k)^2 + (y_j - y_k)^2$
12:         if $SILENT_j$ =FALSE and $SILENT_k$ =FALSE and
13:             sqrds $>$ THRESHOLD $\cdot$ sqrdist then
14:                 $SILENT_j \leftarrow$ TRUE
15:                 $SILENT_k \leftarrow$ TRUE
16:         end if
17:     end for
18: end for
19: All nodes $N_j$ with $SILENT_j$=FALSE run Algorithm 1
```

The algorithm presented in this report is sensitive to faulty sensors; a single faulty sensor can have a significant impact and cause all sensor nodes to misperceive the physical environment. This risk increases as systems become larger and hence it could offsets the advantage of the scalability of the algorithm.

In this section we present a simple extension of the previously proposed algorithm to be able to operate in the presence of sensor faults.

Algorithm 3 obtains an interpolation of sensor readings even in the presence of sensor faults. The idea is simple and it is as follows. An interpolation is created as was done in Algorithm 1. Then each pair of nodes that were selected are inspected and the difference between the sensor readings relative to the distance is computed; if this value is greater than what is possible by the physical dynamics (knowledge specified by the designer) then both nodes are declared as SILENT; meaning that it has detected that at least one of the nodes in the pair has a sensor that was faulty. After that all nodes that are not SILENT run Algorithm 1 again and then the faulty nodes have been eliminated.

Observe that the set of nodes that are selected in line 1 is the same for all nodes. And all nodes agree on which nodes are silent. Also observe that during the execution of lines 8-18, our new algorithm may cause a small number of sensor nodes that are non-faulty to be SILENT. This is acceptable since we consider dense networks.