Technical Report

How a Cyber-Physical System can Efficiently Obtain a Snapshot of Physical Information Even in the Presence of Sensor Faults

Björn Andersson

Nuno Pereira

and Eduardo Tovar

HURRAY-TR-090201

Version: 0

Date: 18-06-2008

How a Cyber-Physical System can Efficiently Obtain a Snapshot of Physical Information Even in the Presence of Sensor Faults

Björn Andersson, Nuno Pereira, and Eduardo Tovar

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@dei.isep.ipp.pt, nap@isep.ipp.pt, emt@isep.ipp.pt

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

Abstract

We present a distributed algorithm for cyber-physical systems to obtain a snapshot of sensor data. The snapshot is an approximate representation of sensor data; it is an interpolation as a function of space coordinates. The new algorithm exploits a prioritized Medium Access Controll (MAC) protocol to efficiently transmit information of the sensor data. It scales to a very large number of sensors and it is able to operate in the presence of sensor faults.

How a Cyber-Physical System can Efficiently Obtain a Snapshot of Physical Information Even in the Presence of Sensor Faults

Björn Andersson, Nuno Pereira, and Eduardo Tovar
IPP-Hurray! Research Group, CISTER/ISEP
Polytechnic Institute of Porto, Portugal

Abstract

We present a distributed algorithm for cyber-physical systems to obtain a snapshot of sensor data. The snapshot is an approximate representation of sensor data; it is an interpolation as a function of space coordinates. The new algorithm exploits a prioritized Medium Access Controll (MAC) protocol to efficiently transmit information of the sensor data. It scales to a very large number of sensors and it is able to operate in the presence of sensor faults.

1. Introduction

Cyber-Physical systems (CPS) involve embedded computers with tight interaction to the physical dynamics. Oil rigs, refineries and car-park garages need to monitor potentially harmful gases and liquids in order to infer the state of the physical phenomenon, predict the future state and control the state. Ideally, the CPS should obtain a snapshot of the entire physical setting from all sensors and this should be done with no delay. In reality however, computers and sensors are discrete quantities so sampling must be performed; that is, sensor readings are available only at certain time instants and only at certain geographical locations. Several researchers have therefore formulated the necessary sampling frequency in time; this is often referred to as the Nyquist sampling theorem [9]. And similar reasoning applies to sampling in space.

With distributed sensing, designers of CPS are facing a dilemma. A designer can deploy a large number of sensors as a dense network and hence obtain excellent resolution in space (high sampling rate in space). But if those sensors are in a small geographical area then no two sensor nodes can broadcast their sensor readings simultaneously (on a wireless channel or a shared bus) so the time required in order to gather all sensor readings becomes large and consequently the sampling rate in time becomes low. A designer can take a diametrically opposite strategy by deploying a small number of sensor nodes and hence obtain a high sampling rate in time but not in space. It would be desirable to offer designers of CPS the best of both worlds: high sampling rate in both space and time.

The cause of this dilemma is that with a straightforward algorithm, the time complexity for a CPS to obtain a snapshot of its sensors is $O(m)$ where m is the number of sensor nodes. If an algorithm would be available that allows CPS to quickly obtain an approximate representation of sensor readings as a function in space then a high sampling rate in both time and space could be achieved. One may believe that m sensor nodes each producing 8-bit sensor readings would produce $8m$ bits of information. If sensor readings were independent random variables then this would be true indeed. But sensor readings are typically correlated in space so the amount of information is much smaller. Analogous to the Shannon's theory [11] of transferring information we are not interested in just transferring sensor readings (=channel symbols) but to transfer information. Unlike Shannon however we will not derive any channel capacity of our ability to transport sensor information but we will look at how the communication subsystem can transfer the small amount of information in a CPS where a large number of sensor reading are generated. We expect this problem to be of increasing importance as CPS are deployed in larger quantities and more pervasively in the physical environment.

Motivated by this trend, an algorithm for obtaining an interpolation of sensor readings as a function of space coordinates [4, 3] has been designed. Its time complexity grows very slowly with the number of sensor nodes. Such an interpolation of sensor readings is important since it gives a snapshot of the environment. A prioritized medium access control protocol was exploited and this made the algorithm highly efficient. Unfortunately, the algorithm was 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.

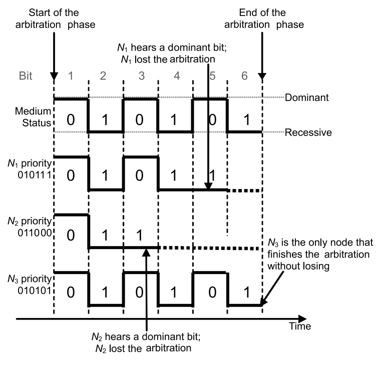
Therefore, in this paper, we extend the previously proposed algorithm [4, 3] to be able to operate in the presence of sensor faults.

Our new algorithm assumes that all nodes are positioned such that they are all in a single broadcast domain. This is not a fundamental limitation; it can be removed by using a technique in [5]. Our new algorithm also assumes that each node runs a prioritized medium access control protocol. Such protocols are common. The CAN bus [6], deployed in more than 400 million units, offers that behavior and it is one of the most common communication technologies used in practice today. In fact, the CAN bus was used in previous work that implemented an interpolation algorithm [4, 3]; see [1]. A similar behavior can be achieved for wireless channels as well [10].

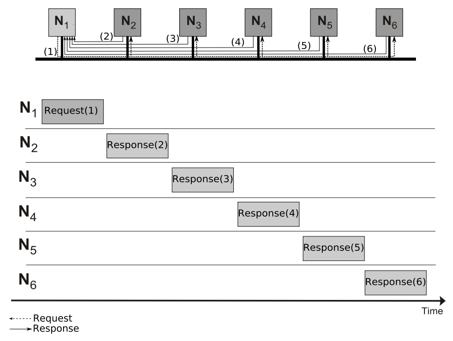
The remainder of this paper is structured as follows. Section 2 gives an application background and the main idea of how a prioritized MAC protocol can be exploited for computing aggregated quantities. The algorithms for efficiently obtaining an interpolation is presented in Section 3. It also presents how to make it able to operate even in the presence of sensor faults. Finally, in Section 4, conclusions are drawn.

2. Preliminaries and Motivation

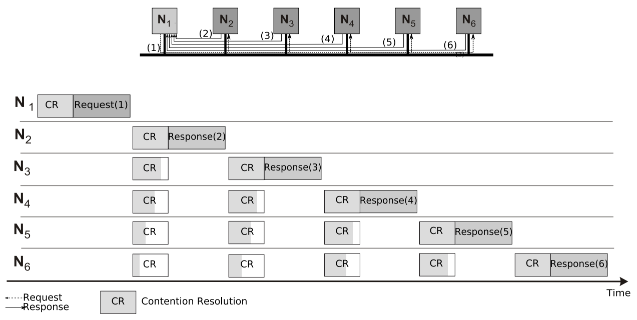
The basic premise for this work is the use of a prioritized MAC protocol. This implies that the MAC protocol assures that out of all nodes contending for the medium at a given moment, the one(s) with the highest priority gain access to it. This is inspired by Dominance/Binary-Countdown protocols [8]. 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.


Start of the arbitration phase
End of the arbitration phase
N1 hears a dominant bit; N1 lost the arbitration
Bit 1 2 3 4 5 6
Medium Status
Dominant
Recessive
N1 priority 010111
N2 priority 011000
N3 priority 010101
N3 is the only node that finishes the arbitration without losing
N2 hears a dominant bit; N2 lost the arbitration
Time

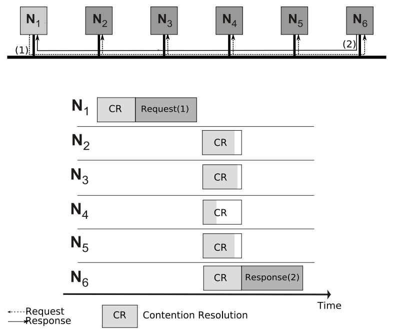
(a) CAN arbitration



(b) Naïve algorithm (TDMA-like MAC)



(c) Naïve algorithm (CAN-like MAC)



(d) Novel algorithm (priorities assigned at run-time according to the sensed values)

Figure 1. Dominance/Binary-Countdown Arbitration and Motivating Examples.

During the arbitration (depicted in Figure 1(a)), 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, only one node reaches the end of the collision resolution phase, and this node (the winning node) proceeds with transmitting the data bits 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 [6] 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. 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 5000 meters long (with no repeaters) using a bit rate of 10 Kbit/s [2]. 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 above this value (which is often also imposed by the software of the CAN transceivers).

The focus of this paper will be on exploiting a prioritized MAC protocol for efficient distributed computation of aggregated quantities. We present 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 run-time 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.

2.1. Motivation and the Main Idea

The problem of computing aggregated quantities in a single broadcast domain can be solved with a naïve algorithm: every node broadcasts its sensor reading. Hence, all nodes know all sensor readings and then they can compute 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 1(b), 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 compute a waiting timeout for replies based on this knowledge. Clearly, with this approach, the execution time depends on the number of neighbor nodes (m). Figure 1(c) depicts another naïve approach, but this using a CAN-like MAC protocol (The different length of the gray bars inside the boxes depicting the contention in Figure 1(c) represent the amount of time that the node actively participated in the medium contention). 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 run-time. Note that to send a message, nodes have to perform arbitration before accessing the medium. When a node wins access to the medium, it sends its response and stops trying to access the medium. It is clear that using a naïve approach with CAN brings no advantages as compared to the other naïve solution (Figure 1(b)).

Consider now that instead of using their priorities to access the medium, nodes use as priority the value of its sensor reading. 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[1]. This alternative would allow an approach as depicted in Figure 1(d). With such an approach, to compute 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

[1]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.

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 as a result of the contention nodes will know the priority of the winner, no more information needs to be transmitted by the winning node. 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 with more information (for example, the node ID) in the least significant bits, such that priorities will be unique. In this scenario, the time to compute the minimum temperature reading only depends on the time to perform the contention for the medium, not on m.

A similar approach can be used to compute 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 two examples of how simple aggregate quantities can be computed with a minimum message-complexity (and therefore time complexity) if message priorities are dynamically assigned at run-time upon the values of the sensed quantity. In Section 3 we introduce an algorithm for obtaining an interpolation.

2.2. 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 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 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 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 winner has transmitted its packet (assuming that the winner did not send an empty packet). Then, the node contends for the channel again. `send` blocks until it has won the contention and transmitted a packet. `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), `send_empty` gives the control back to the application and returns the priority of the winner.

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: MAXNNODES denotes an upper bound on $m$.
Require: (MAXS+1)×(MAXNNODES+1)+MAXNNODES≤MAXP.
function find_nodes() return a set of packets
    myinterpolatedvalue ← 0
    num ← 0.0
    denom ← 0.0
    S ← ∅
    update_myinterpolation ← TRUE
    for j ← 1 to k do
        error ← abs( $s_i$ - to_integer(myinterpolatedvalue) )
        prio ← MAXP - (error × (MAXNNODES + 1) + i)
        snd_pack ← <$s_i$,$x_i$,$y_i$>
        <win_prio, rcv_pack> ← send_and_rcv( prio, snd_pack)
        if win_prio = prio then
            update_myinterpolation ← FALSE
            myinterpolatedvalue ← $s_i$
        end if
        if update_myinterpolation = TRUE then
            dx ← $x_i$ - recv_pack.x
            dy ← $y_i$ - recv_pack.y
            weight ← 1.0 / ( dx × dx + dy × dy )
            num ← num + recv_pack.value × weight
            denom ← denom + weight
            myinterpolatedvalue ← num/denom
        end if
        S ← S ⋃ recv_pack
    end for
    return $S$
end function
```

`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 [$MINS$, $MAXS$] and it is assumed that $MINS$=0.

We assume that sensor readings might be faulty, but correct sensor readings are always in majority within a set of neighbor nodes. Sensor readings are assumed to to exhibit spatial locality [7]. This means that nodes that are close in space have similar sensor readings, and, based on the designer's knowledge, it is possible to define a parameter that bounds the gradient of the signal amongst non-faulty neighboring nodes.

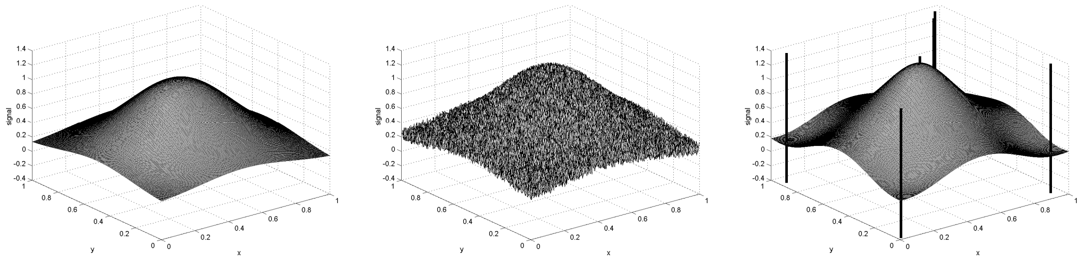
3. Interpolation of Sensor Data With Location

In this section we assume that a node N_i knows its location given by two coordinates (x_i,y_i). Location-awareness brings the possibility 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. Section 3.1 presents the main idea of the interpolation and Section 3.2 extends it to deal with sensor faults.

3.1. Assume no Sensor Faults

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:

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



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

Figure 2. Interpolation Example 1.

and let e denote the global error; that is:

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

We will use weighted-average interpolation (WAI) [13] (also used in [15, 12]). WAI is defined as follows:

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

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

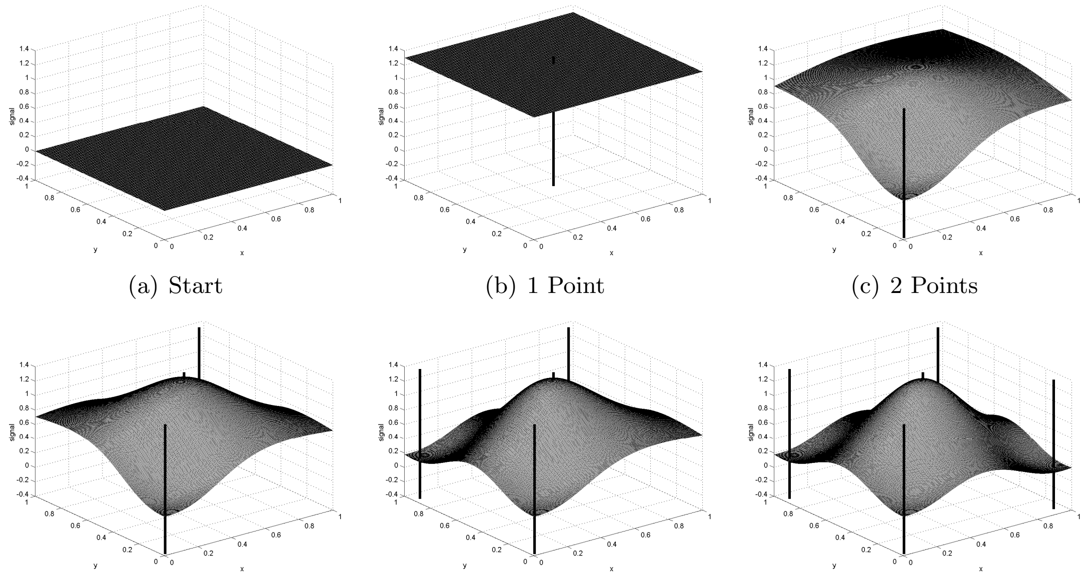
$$w_i(x,y) = \frac{1}{(x_i - x)^2 + (y_i - y)^2} \tag{4}$$

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.

The original version [13] of weighted-average interpolation uses all points; that is $S = \{1, 2, 3, \ldots, m\}$. But this would imply that computing Equation 3 has a time complexity of O(m). Fortunately, it is often the case [7] 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.

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 contribute to 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 $f(x,y)$ 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 8) the error. This error is concatenated with the identifier of the node (together this forms the priority of the message). This ensures that all priorities are unique. All nodes send their messages in



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

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

Figure 3. Iterations Concerning Interpolation Example 1.

parallel (on line 11) and one packet will win the contention. Recall from Section 2.2 that when nodes call `send_and_rcv`, then both priority of the winner and the packet transmitted by the winner is returned to the application on every node. This packet is added (on line 24) to the set S, which keeps track of all received packets related to the problem of creating an interpolation. If node N_i did not win the contention, then it updates (on lines 17-22) the interpolated value at its position.

Figures 2 and 3 illustrate the operation of our interpolation scheme. Figure 2(a) illustrates a signal that varies over space. We add noise and obtain the signal in Figure 2(b). Our algorithm is used to find a subset of $k = 6$ nodes that are to be used in the interpolation. The result of this interpolation is shown in Figure 2(c). The location of the nodes in S are indicated with vertical lines.

Figure 3 provides further intuition on the behavior of the proposed algorithm. 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 as depicted in Figure 3(a). 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, leading to the surface as depicted in Figure 3(b). Then 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. The iterations until $k = 5$ are depicted in Figure 3. The result of the final iteration, for $k = 6$, is depicted in Figure 2(c). It can be seen that the interpolation tracks well the original signal.

3.2. Assume that Sensor Faults Can Occur

Algorithm 2 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

Algorithm 2 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
```

between the sensor readings relative to the distance is computed; if this value is greater than what is possible by the physical dynamics (this is knowledge specified by the designer, using variable $THRESHOLD$ and defines the upper bound for the square of the difference between two pairs of sensor readings) 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.

4. Conclusions

We have previously shown how to use a prioritized MAC protocol to obtain an interpolation efficiently. In this paper we have extended this algorithm to be able to operate even in the presence of sensor faults. For future work, we consider (i) the use of robust estimation techniques for outlier detection and (ii) the use of model-based diagnosis techniques [14] that use physical dynamics to create hypotheses of the set of faulty node and attempts to find the minimum set.

References

[1] http://www.hurray.isep.ipp.pt/activities/data-ag/.

[2] Softing AG. CAN bus practical bus length. http://www.softing.com/home/en/industrial-automation/products/can-bus/more-can-bus/bit-timing/practical-bus-length.php?navanchor=3010538.

[3] B. Andersson, N. Pereira, W. Elmenreich, E. Tovar, F. Pacheco, and N. Cruz. A scalable and efficient approach to obtain measurements in CAN-based control systems. *in IEEE Transactions on Industrial Informatics*, 4(2), May, 2008.

[4] B. Andersson, N. Pereira, and E. Tovar. Exploiting a prioritized MAC protocol to efficiently compute interpolations. In *12th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA'07)*, Patras, Greece, 2007.

[5] B. Andersson, N. Pereira, and E. Tovar. Exploiting a prioritized MAC protocol to efficiently compute MIN and MAX in multihop networks. In *5th IEEE International Workshop on Intelligent Solutions in Embedded Systems (WISES'07)*, Madrid, Spain, 2007.

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

[7] 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 3rd International Symposium on Information Processing in Sensor Networks (IPSN'04)*, 2004.

[8] A. K. Mok and S. Ward. Distributed broadcast channel access. *Computer Networks*, 3:327–335, 1979.

[9] H. Nyquist. Certain topics in telegraph transmission theory. *Transactions of AIEE*, 47:617–644, 1928. Reprinted in Proc. of the IEEE, Vol 90, No 2, Feb 2002.

[10] N. Pereira, B. Andersson, and E. Tovar. WiDom: a dominance protocol for wireless medium access. *IEEE Transactions on Industrial Informatics*, 3:120–130, 2007.

[11] C. Shannon. A mathematical theory of communication. *Bell System Technical Journal*, 27:379–423 and 623–656, 1948. Available at http://cm.bell-labs.com/cm/ms/what/shannonday/paper.html.

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

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

[14] G. Steinbauer and F. Wotawa. Combining quantitative and qualitative models with active observations for better diagnoses of autonomous mobile robots. In *5th IEEE International Workshop on Intelligent Solutions in Embedded Systems (WISES'07)*, Madrid, Spain, 2007.

[15] R. Tynan, G.M.P. OHare, D. Marsh, and D. OKane. Interpolation for Wireless Sensor Network Coverage. In *Proceedings of the the Second IEEE Workshop on Embedded Networked Sensors*, pages 123– 131, 2005.