

Chapter 2

Using a Prioritized Medium Access Control Protocol for Incrementally Obtaining an Interpolation of Sensor Readings

Björn Andersson, Nuno Pereira, Eduardo Tovar and Ricardo Gomes

2.1 Introduction

A sensor network comprises a set of computer nodes each one equipped with a processor, memory, sensors and a transceiver for communications over a (wired or wireless) channel. The sensor network must obtain an accurate image of physical phenomena and do so with a high sampling rate in both time and space. A large number of computer nodes are needed in order to obtain a high sampling rate in space. But this generates a large number of sensor readings and since these sensor readings are located on different computer nodes, a significant amount of communication may be necessary forcing a reduction in the sampling rate in time. For systems with a very large number of computer nodes, it is therefore crucial to develop techniques that make it possible to obtain a snapshot, an approximate representation of all sensor readings, and achieve this with a time-complexity (as a function of the number of nodes) that is small.

A simple approach for obtaining an approximate representation of sensor readings would be to select a subset of the computer nodes at random and let the sensor readings at those computer nodes be used for obtaining an interpolation. Although this is fast, it has the drawback that some computer nodes with extreme sensor readings may have a significant impact on the interpolation if they would be selected but they may not be selected and this causes (as illustrated in [1]) the interpolation to be a poor representation of the physical phenomenon. And this can cause a sensor network to misperceive its physical environment.

A better approach for obtaining an approximate representation of sensor readings would be to select a subset of the computer nodes, carefully selected to be the ones that represent local extreme points and let the sensor readings at those

B. Andersson (✉) · N. Pereira · E. Tovar · R. Gomes
CISTER/IPP-Hurray Research Unit, Polytechnic Institute of Porto, Porto, Portugal
e-mail: bandersson@dei.isep.ipp.pt

computer nodes be used for obtaining an interpolation. If one computer node had knowledge of all sensor readings then such a selection would be possible of course but in practice, a computer node only knows its own sensor reading (unless sensor readings are communicated) and therefore it has been non-obvious how to implement such an approach.

Recent work [1, 2] however have shown how to exploit a prioritized medium access control (MAC) protocol for selecting local extreme points and thereby it was shown how to quickly obtain an interpolation of sensor readings where sensor readings were taken by different computer nodes. This work assumes that the MAC protocol has a very large number of priority levels and that all sensor nodes know the priority of the node that was granted the channel. Such MAC protocols are common; the Controller Area Network (CAN) [3] bus is one such example for wired communication (with more than 300 million units sold) and a similar technology, WiDOM [2, 4] is available for wireless communication.

The algorithm [1, 2] which exploited a prioritized MAC protocol had a user-selectable parameter, k , which had the role that the k sensor nodes that contribute the most to the interpolation being a faithful representation of the physical reality are selected and the interpolation is based on those k sensor nodes. k is selected based on the number of local extrema of the signal as explained in [1]. With this approach it was possible to obtain the interpolation with a time-complexity that is $O(k)$, that is, the time-complexity is independent of the number of sensor nodes; yet the result of the interpolation was dependent on all sensor readings. The algorithm was implemented and tested both in wired systems (using CAN [3]) and in wireless systems (using WiDom [2, 4]).

The algorithm for obtaining an interpolation (i) had to run until completion and (ii) it was designed to have no prior knowledge of the physical environment. Unfortunately, these two facts bring two drawbacks:

1. There are situations where the delay from when the physical world changes until the computer system can react to this change is two times the duration required for obtaining the interpolation. (This situation occurs when the environment changed just after the algorithm for obtaining the interpolation had started; when this happens, the algorithm for obtaining the interpolation must finish execution and then take new sensor readings and finally obtain an interpolation of these new sensor readings.)
2. It is necessary that the sampling period of an application that uses the interpolation is $O(k)$ or greater. If the entire physical environment changes everywhere, it may really be necessary to obtain an interpolation from scratch. But one can expect that a change in the physical environment (such as a rapid fire, explosion or deformation) has only local effects initially (during the first milliseconds) and it changes the entire environment later. It would be desirable to use a sampling rate so high that the sampling period is independent of k and independent of the number of nodes, yet the system is able to detect extreme local changes with a duration of two sampling periods and obtain an image of the entire physical environment within k sampling periods.

Therefore, we presented, in a workshop paper [5], a new algorithm for obtaining an interpolation of sensor readings which eliminates the two above mentioned drawbacks. This chapter is an extension of that paper.

The main idea of the algorithm is that when the system starts-up, an interpolation is obtained using the previously known algorithm [1, 2]. This step of the algorithm has the time-complexity $O(k)$, which is larger than we desire but it is done only once. Each computer node now has the k sensor readings that can be used to form an interpolation of all sensor readings. All computer nodes will now periodically take sensor readings with a small period; this period is independent of k and it is independent of the number of computer nodes. All computer nodes take their sensor readings in parallel and then each computer node computes the interpolated value at itself and compares it to its own sensor reading. The computer node whose sensor reading contributes the least to the faithfulness of the interpolation is attempted to be deselected and the computer node whose sensor reading contributes the most to the faithfulness of the interpolation is selected.

We believe this algorithm to be useful for detecting deviations from the expected behavior in the physical world very quickly, for example detecting the deformation of mechanical elements (for example in a car or aircraft) in order to enact appropriate safety actions (such as deciding which airbag to inflate or which fuel pump to be stopped or which valve to be closed).

The remainder of this paper is organized as follows. [Section 2.2](#) gives preliminaries, that is, the main idea of how a prioritized MAC protocol can be used for computations and also the system model we will use. [Section 2.3](#) discusses how to obtain an interpolation; this discussion leads to the new interpolation scheme. [Section 2.4](#) gives conclusions and future work.

2.2 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 [6]. 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 requesting to transmit the highest-priority message succeeds.

During the arbitration (depicted in [Fig. 2.1](#)), 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

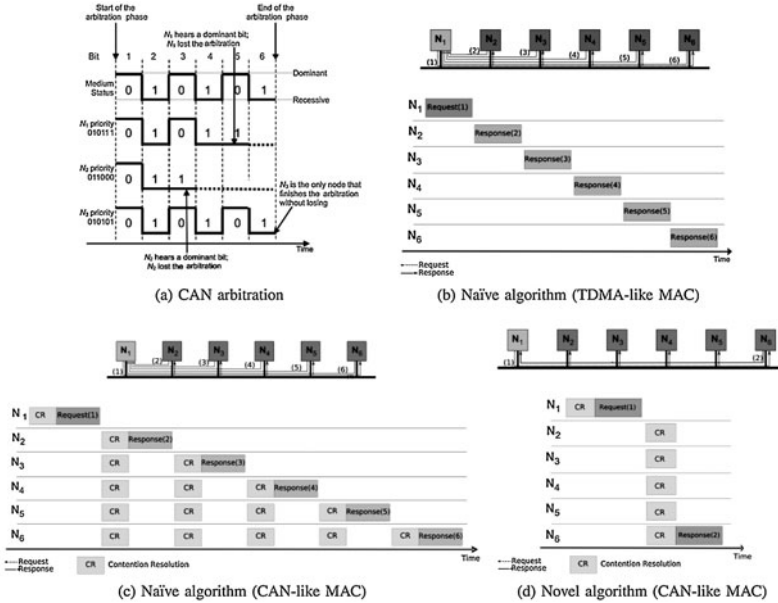


Fig. 2.1 Dominance/binary-countdown arbitration motivating examples. **a** Example of bitwise arbitration; **b** example application where N_1 needs to know the minimum (MIN) temperature reading among its neighbors (N_2 – N_6); **c** possible solution for the example application using a CAN-like MAC, 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

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 [3] 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 [7] for more examples of application fields and figures about the use of CAN technologies). 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 1 Mbit/s for a maximum bus length of 30 m, or a bus 1,000 m 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 [8]).

The focus of this paper is on exploiting a prioritized MAC protocol for efficiently obtaining an interpolation function which approximates the sensor readings in a geographical area. A key idea in the design of such an algorithm is the use of a prioritized MAC protocol for performing computations—this is explained next.

2.2.1 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 Fig. 2.1b, 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, with this approach, the execution time depends on the number of neighbor nodes (m). Figure 2.1c 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 (Fig. 2.1b).

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 Fig. 2.1d. 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.

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 [Sect. 2.3](#) we will show how this technique of using a prioritized MAC protocol for computations can be used for obtaining an interpolation of sensor readings.

2.2.2 System Model

The network consists of m nodes that take sensor readings where a node is given a unique identifier in the range $1 \dots m$. 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 $[0, \text{MAXP}]$, where lower numbers correspond to higher priorities. Let NPRIOBITS denote the number of priority bits. This parameter has the same value for all nodes. Since NPRIOBITS is used to denote the number of bits used to represent the priority, the priority is a number in the range of $0 - 2^{\text{NPRIOBITS}} - 1$. Clearly, $\text{MAXP} = 2^{\text{NPRIOBITS}} - 1$.

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 $[0, \text{MAXS}]$ and it is assumed that $\text{MAXS} \leq \text{MAXP}$.

2.3 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 position to present our approach for obtaining an interpolation of sensor readings. We will do so formally with pseudo-code; this pseudo-code returns an upper bound on the error of the interpolation. We will first (in [Sect. 2.3.1](#)) present the main idea of the previously known interpolation scheme.

This will lead us (in [Sect. 2.3.2](#)) to the new incremental interpolation scheme. This new incremental interpolation scheme needs to, as an intermediate result, evaluate the interpolation at certain geographical points. Therefore, we will (in [Sect. 2.3.3](#)) modify this algorithm to perform calculations at those specific points with additional speed and this results in an improvement of the new incremental interpolation scheme.

2.3.1 Previously Known Algorithm

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_j denote the magnitude of the error at node N_j ; that is:

$$e_j = |s_j - f(x_j, y_j)| \quad (2.1)$$

and let e denote the global error; that is:

$$e = \max_{j=1 \dots m} e_j \quad (2.2)$$

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 sensor readings 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) [9–11]. WAI is defined as follows:

$$f(x, y) = \begin{cases} 0 & \text{if } S = \emptyset \\ s_j & \text{if } \exists N_j \in S : x_j = x \wedge y_j = y \\ \frac{\sum_{j \in S} s_j * w_j(x, y)}{\sum_{j \in S} w_j(x, y)} & \text{otherwise} \end{cases} \quad (2.3)$$

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

$$w_j(x, y) = \frac{1}{(x_j - x)^2 + (y_j - y)^2} \quad (2.4)$$

Algorithm 1 Finding a subset of nodes to be used inWAI

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: The code below is executed by every node. A node can read the variable i and obtain its node index.
Require: $(\text{MAXS}+1) \times (\text{MAXNNODES}+1) + \text{MAXNNODES} \leq \text{MAXP}$.
1: **function** find_nodes() **return** a set of packets
2: $S \leftarrow \emptyset$
3: **for** $q \leftarrow 1$ to k **do**
4: Calculate $f(x_i, y_i)$ in Equation 3 and assign
 it to the variable "myinterpolatedvalue"
5: $\text{error} \leftarrow \text{abs}(s_i - \text{to_integer}(\text{myinterpolatedvalue}))$
6: $\text{temp_prio} \leftarrow \text{error} \times (\text{MAXNNODES} + 1) + i$
7: $\text{prio} \leftarrow (\text{MAXP}+1) - \text{temp_prio}$
8: $\text{snd_pack} \leftarrow \langle s_i, x_i, y_i \rangle$
9: $\langle \text{winning_prio}, \text{rcv_pack} \rangle \leftarrow \text{send_and_rcv}(\text{prio}, \text{snd_pack})$
10: $S \leftarrow S \cup \{ \text{rcv_pack} \}$
11: **end for**
12: **return** S
13: **end function**

Intuitively, Eqs. 2.3 and 2.4 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 [12–14].

The original version [9] of weighted-average interpolation uses all available sensor readings for interpolation. But this would imply that computing Eq. 2.3 from sensor readings has a time complexity of $O(m)$. Fortunately, it is often the case [15] 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 Eq. 2.3. We select a number of k nodes that contribute to lowering the error of 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

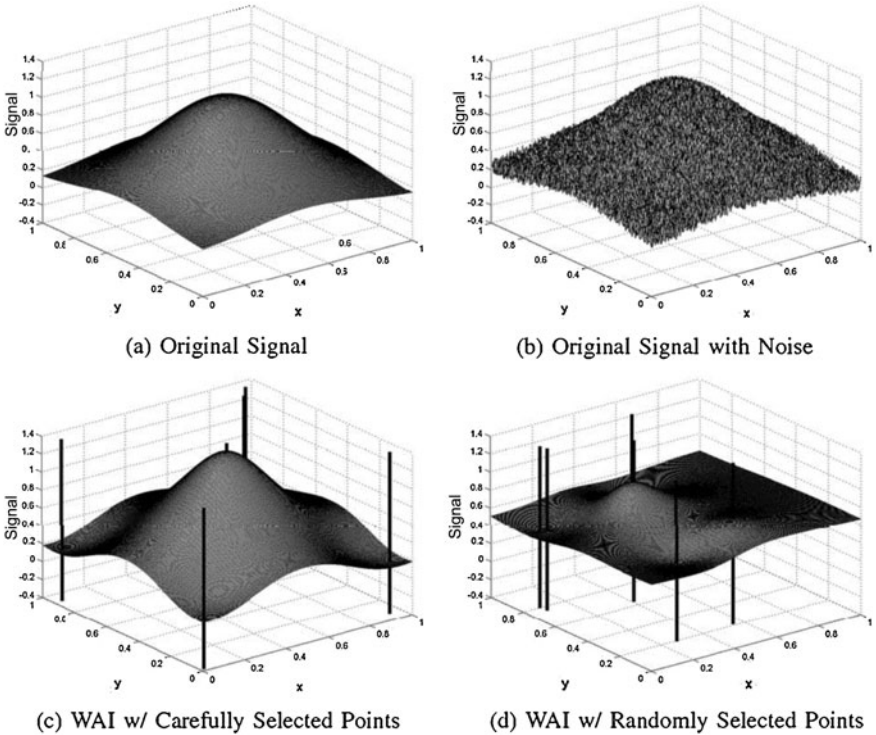


Fig. 2.2 Interpolation Example 1. **a** The original signal that varies over space; **b** noise added to the original signal; **c** the result of the interpolation given by our algorithm. The location of the subset of $k = 6$ nodes that were selected to be used in the interpolation is indicated with vertical lines; **d** example result of the interpolation when nodes are selected randomly

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 and it is previously published [1]. 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 Sect. 2.2.2 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 2.2 and 2.3 illustrate the operation of this scheme. It can be seen that the interpolation result is smooth and that it tracks well the original signal. However, performing weighted-average interpolation with six randomly selected nodes gives poor interpolation. This is illustrated in Fig. 2.2d.

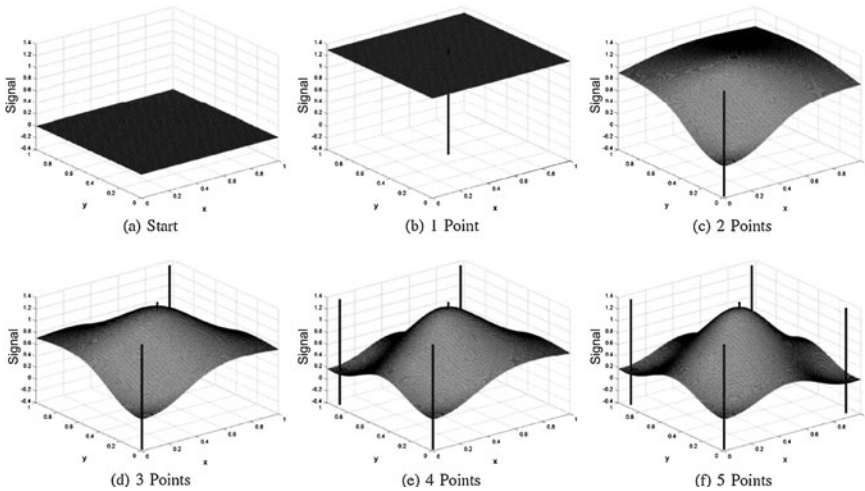


Fig. 2.3 Iterations concerning interpolation Example 1. **a** At the beginning, there is no point included in S . Therefore, from the definition of $f(x, y)$ in Eq. 2.3 it results that $f(x, y) = 0$. This gives a plane surface; **b** then, each node calculates the error between its sensor reading and 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 $k = 6$

2.3.2 New Algorithm

The previously proposed algorithm [1] (which was presented in previous section and stated in Algorithm 1) obtained an interpolation from scratch every time it was run. As mentioned in the introduction of this paper, this brings two drawbacks. It is worthwhile to counter those drawbacks and therefore we need to devise a new algorithm.

Algorithm 2 shows the new interpolation scheme as pseudo-code. The algorithm works as follows. First, Algorithm 1 is called and this gives us a set S with the selected data points. Then the algorithm executes lines 4–17 periodically; it is assumed that the execution of lines 4–17 is initiated periodically. The execution of lines 4–17 differs from the one in Algorithm 1 in only two respects. First, only one data point is selected instead of k data points. Second, the computation of lines 4–17 begins by removing one element in S (done at lines 5–6) and then a new element is added (done at line 17). The rationales for these adding and deleting rules are as follows. We desire to find the computer node whose sensor reading contributes the least to a faithful representation of the physical world. It would be possible to find that out using the prioritized MAC protocol but this would require some communication. Instead, we simply remove the node which was added to S least recently (done at lines 5–6). Then line 17 adds the element that contributes the most to a faithful representation of the physical world.

Algorithm 2 New Algorithm for finding a subset of nodes to be used in WAI

Require: All nodes start Algorithm 2 simultaneously.

Require: k denotes the desired number of interpolation points.

Require: A node N_i knows x_i, y_i and s_i .

Require: The code below is executed by every node. A node can read the variable i and obtain its node index.

Require: $(\text{MAXS}+1) \times (\text{MAXNNODES}+1) + \text{MAXNNODES} \leq \text{MAXP}$.

- 1: all nodes take sensor readings; the sensor reading at computer node N_j is s_j .
 - 2: call find_nodes (in Algorithm 1) and let S denote the set that is returned
 - 3: **while** (true) **do begin**
 - 4: all nodes take sensor readings; the sensor reading at computer node N_j is s_j .
 - 5: for each element in S , there is a time when it most recently became a member in S , pick the element with the earliest such time and call it OLDNODE
 - 6: $S \leftarrow S \setminus \text{OLDNODE}$
 - 7: **if** $N_i \in S$ **then**
 - 8: Calculate $f(x_i, y_i)$ in Equations 3 and 4 based on $S \setminus \{N_i\}$ and assign it to the variable "myinterpolatedvalue".
 - 9: **else**
 - 10: Calculate $f(x_i, y_i)$ in Equations 3 and 4 based on S and assign it to the variable "myinterpolatedvalue".
 - 11: **end if**
 - 12: $\text{error} \leftarrow \text{abs}(s_i - \text{to_integer}(\text{myinterpolatedvalue}))$
 - 13: $\text{temp_prio} \leftarrow \text{error} \times (\text{MAXNNODES} + 1) + i$
 - 14: $\text{prio} \leftarrow (\text{MAXP}+1) - \text{temp_prio}$
 - 15: $\text{snd_pack} \leftarrow \langle s_i, x_i, y_i \rangle$
 - 16: $\langle \text{winning_prio}, \text{rcv_pack} \rangle \leftarrow \text{send_and_rcv}(\text{prio}, \text{snd_pack})$
 - 17: $S \leftarrow S \cup \{ \text{rcv_pack} \}$
 - 18: **end while**
-

We can distinguish between two cases. One case is that the element removed (at lines 5–6) and the element added (at line 17) are the same. This occurs when the new sensor readings obtained at line 4 changed very little.

Another case is that the element removed (at lines 5–6) and the element added (at line 17) are not the same. This occurs when the new sensor readings obtained at line 4 changed a lot and therefore it is necessary that the set S is modified to reflect the changes in the physical environment.

Note that the lines 4–17 can be executed very quickly; only one transmission (line 16) is needed. Executing the lines 8 and 10 have time-complexity $O(k)$ and this is undesirable though. Therefore, the next section will present an improved version of Algorithm 2 which avoids this potential performance bottleneck.

2.3.3 An Improved Version of the New Algorithm

Evaluating $f(x_i, y_i)$ can be performed quickly and easily if $N_i \in S$. The result is simply s_i as can be seen from Eq. 2.3. Evaluating $f(x_i, y_i)$ quickly for the case $N_i \notin S$ requires additional improvements of the algorithm though. We can note that each iteration of the lines 4–17 in Algorithm 2 evaluates $f(x_i, y_i)$ based on elements in the set S . Since this set has k elements, each of these evaluations has time complexity $O(k)$. Recall (from Eq. 2.3) that for this more complex case that we are discussing now, $f(x_i, y_i)$ is defined as:

$$f(x, y) = \frac{\sum_{j \in S} s_j * w_j(x, y)}{\sum_{j \in S} w_j(x, y)} \quad (2.5)$$

Let us define num_i and $denom_i$ as:

$$num_i = \sum_{j \in S} s_j * w_j(x, y) \quad (2.6)$$

and

$$denom_i = \sum_{j \in S} w_j(x, y) \quad (2.7)$$

Hence, we can clearly (for this more complex case) write $f(x_i, y_i)$ as:

$$f(x, y) = \frac{num_i}{denom_i} \quad (2.8)$$

We can clearly evaluate $f(x_i, y_i)$ on line 8 or 10 in Algorithm 2 by calculating $f(x_i, y_i)$ from Eq. 2.8. And we can update num_i and $denom_i$ whenever the set S changes. Therefore, when we add a new node to S , we simply have to also add an extra term to num_i and $denom_i$. Analogously, removing a node from S requires that we subtract a term. Based on this observation, we can reformulate Algorithm 2.

Algorithm 3 Improved Version of the new Algorithm for finding a subset of nodes to be used in WAI

Require: All nodes start Algorithm 3 simultaneously.
Require: k denotes the desired number of interpolation points.
Require: A node N_i knows x_i, y_i and s_i .
Require: The code below is executed by every node. A node can read the variable i and obtain its node index.
Require: $(MAXS+1) \times (MAXNNODES+1) + MAXNNODES \leq MAXP$.

- 1: all nodes take sensor readings; the sensor reading at computer node N_j is s_j .
- 2: call find_nodes (in Algorithm 1) and let S denote the set that is returned
- 3: **if** $N_i \in S$ **then**
- 4: $I_am_in_S \leftarrow \text{true}$
- 5: **else**
- 6: $I_am_in_S \leftarrow \text{false}$
- 7: **end if**
- 8: $num \leftarrow 0$
- 9: $denom \leftarrow 0$
- 10: **for each** $N_j \in S \setminus \{N_i\}$ **do**
- 11: $num \leftarrow num + s_j \cdot w_j(x_i, y_i)$
- 12: **end for**
- 13: **for each** $N_j \in S \setminus \{N_i\}$ **do**
- 14: $denom \leftarrow denom + w_j(x_i, y_i)$
- 15: **end for**
- 16: **while** (true) **do begin**
- 17: all nodes take sensor readings; the sensor reading at computer node N_j is s_j .
- 18: for each element in S , there is a time when it most recently became a member in S , pick the element with the earliest such time and call it OLDNODE
- 19: $S \leftarrow S \setminus \text{OLDNODE}$
- 20: **if** $N_i = \text{OLDNODE}$ **then**
- 21: $I_am_in_S \leftarrow \text{false}$
- 22: **else**
- 23: $j \leftarrow \text{OLDNODE}$
- 24: $num \leftarrow num + s_j \cdot w_j(x_i, y_i)$
- 25: $denom \leftarrow denom + w_j(x_i, y_i)$
- 26: **end if**
- 27: **if** $I_am_in_S$ **then**
- 28: $myinterpolatedvalue \leftarrow s_i$
- 29: **else**
- 30: $myinterpolatedvalue \leftarrow num_i / denom_i$
- 31: **end if**
- 32: $error \leftarrow \text{abs}(s_i - \text{to_integer}(myinterpolatedvalue))$
- 33: $temp_prio \leftarrow error \times (MAXNNODES + 1) + i$
- 34: $prio \leftarrow (MAXP+1) - temp_prio$
- 35: $snd_pack \leftarrow \langle s_i, x_i, y_i \rangle$
- 36: $\langle \text{winning_prio}, rcv_pack \rangle \leftarrow \text{send_and_rcv}(prio, snd_pack)$
- 37: $S \leftarrow S \cup \{rcv_pack\}$
- 38: **if** $\text{winning_prio} = prio$ **then**
- 39: $I_am_in_S \leftarrow \text{true}$
- 40: **else**
- 41: $j \leftarrow \text{the node in } rcv_pack$
- 42: $num \leftarrow num + s_j \cdot w_j(x_i, y_i)$
- 43: $denom \leftarrow denom + w_j(x_i, y_i)$
- 44: **end if**
- 45: **end while**

Algorithm 3 shows this improved version of Algorithm 2. It can be seen that the execution of the lines 17–44 is independent of m and it is also independent of k . Note also that each line can execute very quickly. In particular, note that line 36, `send_and_rcv`, can be executed within 3 ms if the hardware platform from [2] is used. Because the execution of the lines 17–44 can be performed at such a high speed we see that it is possible to build a sensor network that monitors its environment by obtaining an interpolation as a representation of the physical world and obtain an update of that interpolation very quickly. Any extreme localized change in the physical environment will be detected and its position will be found within just 3 ms. Future hardware developments may make it even faster.

2.4 Conclusions

We have presented a new approach for obtaining an interpolation based on sensor readings. We left open the problem of analyzing the error of the interpolation and also finding out whether smaller errors can be achieved if knowledge about the physical phenomenon is known.

Acknowledgements This work was partially funded by CONET, the Cooperating Objects Network of Excellence, funded by the European Commission under FP7 with contract number FP7-2007-2-224053, the ARTISTDesign Network of Excellence on Embedded Systems Design ICT-NoE- 214373 and by the Portuguese Science and Technology Foundation (Fundação para Ciência e Tecnologia—FCT) and the project SmartSkin supported by ISEP.

References

1. Andersson B, Pereira N, Elmenreich W, Tovar E, Pacheco F, Cruz N (2008) A scalable and efficient approach to obtain measurements in CAN-based control systems. *IEEE Trans Ind Inform* 4(2):80–91
2. Pereira N, Gomes R, Andersson B, Tovar E (2009) Efficient aggregate computations in large-scale dense WSN. In: 15th IEEE real-time and embedded technology and applications symposium (RTAS'09), San Francisco, CA, USA
3. CAN Specification, ver: 2.0 (1991) Bosch GmbH, Stuttgart
4. Pereira N, Andersson B, Tovar E (2007) Widom: a dominance protocol for wireless medium access. *IEEE Trans Ind Inform* 3(2):120–130
5. Andersson B, Pereira N, Tovar E, Gomes R (2009) Using a prioritized medium access control protocol for incrementally obtaining an interpolation of sensor readings. In: Proceedings of the 7th workshop on intelligent solutions in embedded systems (WISES'09), Ancona, Italy
6. Mok AK, Ward S (1979) Distributed broadcast channel access. *Comput Netw* 3:327–335
7. (CiA), CAN in automation website [Online]. <http://www.can-cia.org>
8. Kimaldi, network of readers website section [Online]. http://www.kimaldi.com/kimaldi_eng/pro ductos/lectores_de_tarjetas/red_de_lectores_can/red_de_lectores_ampliacion_de_informacion
9. Shepard D (1968) A two-dimensional interpolation function for irregularly-spaced data. In: Proceedings of the 1968 ACM national conference, pp 517–524

10. Tynan R, O'Hare G, Marsh D, O'Kane D (2005) Interpolation for wireless sensor network coverage. In: Proceedings of the second IEEE workshop on embedded networked sensors, pp 123–131
11. Sharifzadeh M, Shahabi C (2004) Supporting spatial aggregation in sensor network databases. In: Proceedings of the 12th annual ACM international workshop on geographic information, pp 166–175
12. Crossbow, MICA2—wireless measurement system product datasheet. http://www.xbow.com/products/Product_pdf_files/Wireless_pdf/MICA2_Datasheet.pdf
13. Crossbow, MicaZ—wireless measurement system product datasheet. http://www.xbow.com/products/product_pdf_files/Wireless_pdf/MICAZ_Datasheet.pdf
14. Polastre J, Szewczyk R, Culler D (2005) 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, New York, pp 364–369
15. Guestrin C, Bodík P, Thibaux R, Paskin M, Madden S (2004) Distributed regression: an efficient framework for modeling sensor network data. In: Proceedings of the third international conference on information processing in sensor networks (IPSN04)

Solutions on Embedded Systems

Conti, M.; Orcioni, S.; Martínez Madrid, N.; Seepold, R.E.D. (Eds.)

2011, X, 314 p., Hardcover

ISBN: 978-94-007-0637-8