Exploiting a Prioritized MAC Protocol to Efficiently Compute Interpolations

Björn Andersson, Nuno Pereira and Eduardo Tovar
IPP-HURRAY! Research Group,
Polytechnic Institute of Porto (ISEP-IPP),
Rua Dr. António Bernardino de Almeida 431,
4200-072 Porto, Portugal
{bandersson,npereira,emt}@dei.isep.ipp.pt

Abstract

Consider a network where all nodes share a single broadcast domain such as a wired broadcast network. Nodes take sensor readings but individual sensor readings are not the most important pieces of data in the system. Instead, we are interested in aggregated quantities of the sensor readings such as minimum and maximum values, the number of nodes and the median among a set of sensor readings on different nodes. In this paper we show that a prioritized medium access control (MAC) protocol may advantageously be exploited to efficiently compute aggregated quantities of sensor readings. In this context, we propose a distributed algorithm that has a very low time and message-complexity for computing certain aggregated quantities. Importantly, we show that if every sensor node knows its geographical location, then sensor data can be interpolated with our novel distributed algorithm, and the message-complexity of the algorithm is independent of the number of nodes. Such an interpolation of sensor data can be used to compute any desired function; for example the temperature gradient in a room (e.g., industrial plant) densely populated with sensor nodes, or the gas concentration gradient within a pipeline or traffic tunnel.

1. Introduction

In this paper we show that a prioritized medium access control (MAC) protocol can dramatically reduce the number of messages that needs to be transmitted, and this reduces the time-complexity for computing certain aggregated quantities in a single broadcast domain. In particular, we show that the minimum value (MIN) can be computed with a time-complexity that is O(N_{PRIOBITS}), where N_{PRIOBITS} is the number of bits used to represent the sensor data. Interestingly, the message-complexity (and thus, the time-complexity) is independent of the number of sensor nodes. The same technique can be used to compute the maximum value (MAX).

An additional, and more elaborated, application example 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 computes 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; the time-complexity of obtaining the interpolation is independent of the number of nodes as well.

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 depend on the availability of a prioritized MAC protocol that supports a very large range of priority levels and is collision-free assuming that priorities are unique and such MAC protocols are available. The CAN bus [4] is an example of that, and it is deployed in more than hundred millions of units.

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. It also overviews the system model used throughout the rest of the paper. The algorithms for efficiently computing aggregated quantities for sensor data with location-awareness are then presented in Section 3. Section 4 discusses practical aspects of the proposed novel algorithms. It also discusses the ability of previous work to solve the problem addressed in this paper. Finally, in Section 5, 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 [6]. In such protocols, messages are assigned unique priorities, and before nodes try to transmit they perform a contention resolution phase named ar-

bitration such that the node trying to transmit the highest-priority message succeeds.

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 [4] 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 1000 meters long (with no repeaters) using a bit rate of 50 Kbit/s [1]. 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 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 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.

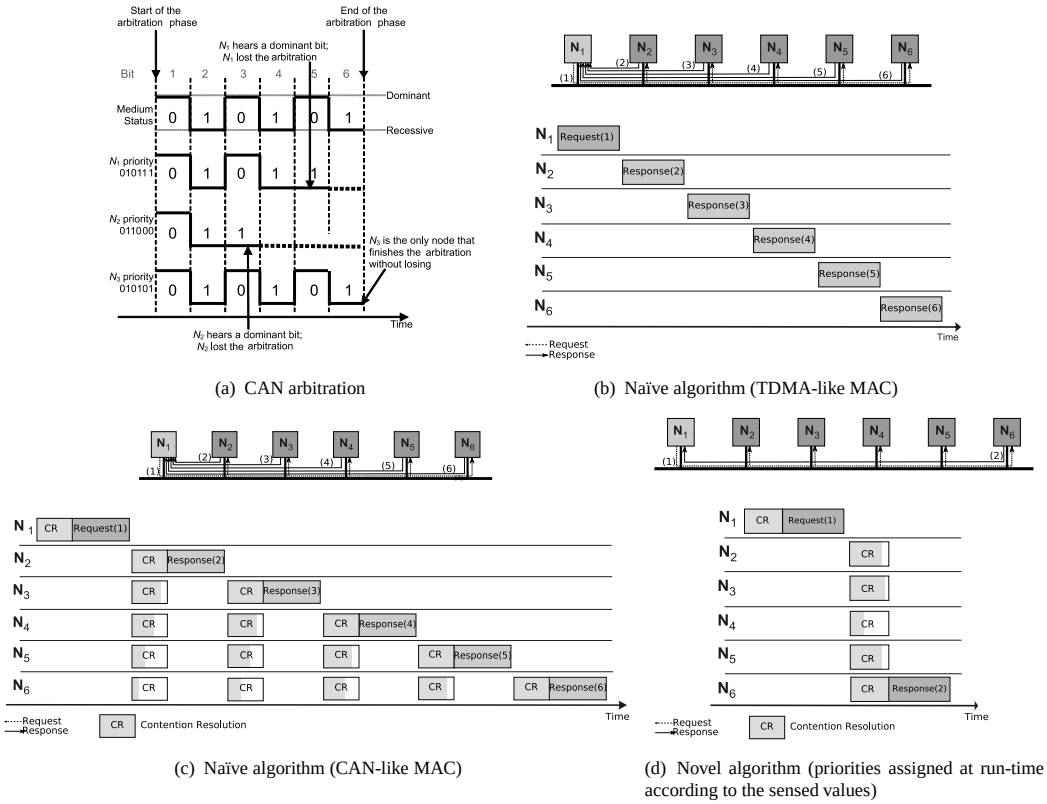
2.1 Motivation and the Main Idea

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 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.

[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.



(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.

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 will introduce a more complex aggregated quantity that can also be efficiently computed by using such a MAC approach.

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, $\mathrm{MAXP} = 2^{NPRIOBITS} - 1$.

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

3 Interpolation of Sensor Data With Location

In this section we will 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). Location-awareness brings the possibility of computing interesting aggregated quantities. For example, 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 compute 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) [8] (also used in [9, 7]). WAI is defined as follows:

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

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. There are many possible choices on how the weight could 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.

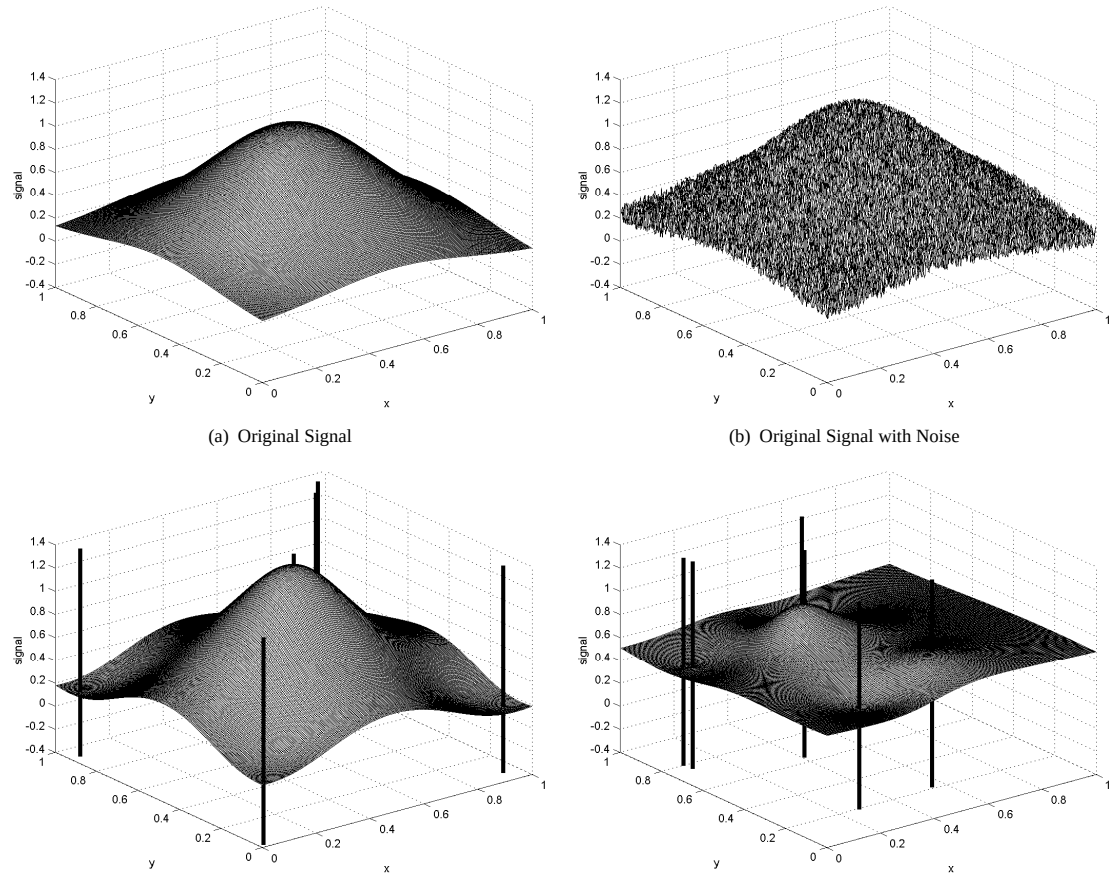
The original version [8] of weighted-average interpolation uses all points; that is $S = \{1, 2, 3, \ldots, m\}$. But this would imply that computing Equation 3 has a time-complexity of O(m). Fortunately, it is often the case [5] that sensor readings exhibit spatial locality; that is, nodes that are 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 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 [2] 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 are indicated with vertical lines.

Figure 3 provides further intuition on the behavior of



(a) Original Signal

(b) Original Signal with Noise

(c) WAI w/ Carefully Selected Points

(d) WAI w/ Randomly Selected Points

Figure 2. Interpolation Example 1.

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

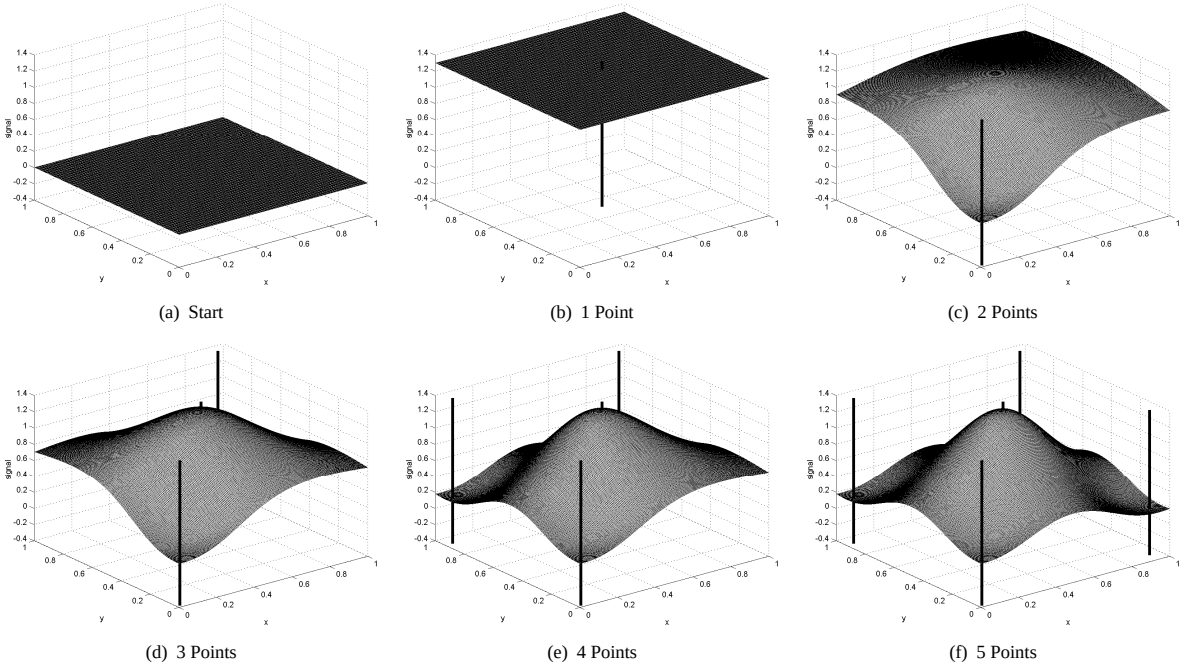
Require: All nodes start Algorithm 1 simultaneously.
Require: k denotes the desired number of interpolation points.
Require: A node N_i knows x_i,y_i and s_i.
Require: MAXNNODES denotes an upper bound on m.
Require: (MAXS+1)×(MAXNNODES+1)+MAXNNODES≤MAXP.

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 $\bigcup$ recv_pack
  end for
  return S
end function

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 $k = 6$, will be as it is depicted in Figure 2(c).

It can be seen that the interpolation result is smooth and that it tracks well the original signal. But performing weighted-average interpolation with 6 nodes selected randomly gives poor interpolation. This is illustrated in Figure 2(d).

Another example, with two peaks, is illustrated in Figure 4. It can be seen that our interpolation scheme still performs well and it shows that the idea is promising. With further experimentation (see Appendix B in our technical report [3]) we have found that the interpolation technique performs well as long as the signal does not change too



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

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

Figure 3. Iterations Concerning Interpolation Example 1.

abruptly when the location changes.

The time-complexity of our interpolation scheme is low. It is $O(k \cdot NPRIOBITS)$ for obtaining an interpolation and it is $O(k)$ for computing the value of the interpolation at a specific point, assuming that an interpolation is available.

4 Discussion and Previous Work

The problem of obtaining an interpolation to extract data from a set of nodes that take sensor readings is attracting increasing attention in the research community [9, 7, 5]. But unfortunately, these algorithms are designed for wireless multihop networks, and if those algorithms would be applied in a single broadcast domain then their time-complexity would heavily depend on the number of nodes.

In this work we have assumed that all nodes start the execution of the protocol simultaneously. In practice, this can be handled easily by letting a node broadcast a message containing a request to compute the aggregated quantity. This request would provide a time reference for all nodes. By analyzing the worst-case execution time of the interpolation algorithm, it would then be possible to setup 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.

5 Conclusions

We have shown how to use a prioritized MAC protocol to compute aggregated quantities efficiently. The design of algorithms that exploit such MAC protocol, can dramatically reduce their time-complexity. This is clearly important for applications that operate under real-time constraints.

Acknowledgements

This work was partially funded by the Portuguese Science and Technology Foundation (Fundação para Ciência e Tecnologia - FCT) and the ARTIST2 Network of Excellence on Embedded Systems Design.

References

[1] http://www.can-cia.org/products/canpg2006/html/about_can1.html.

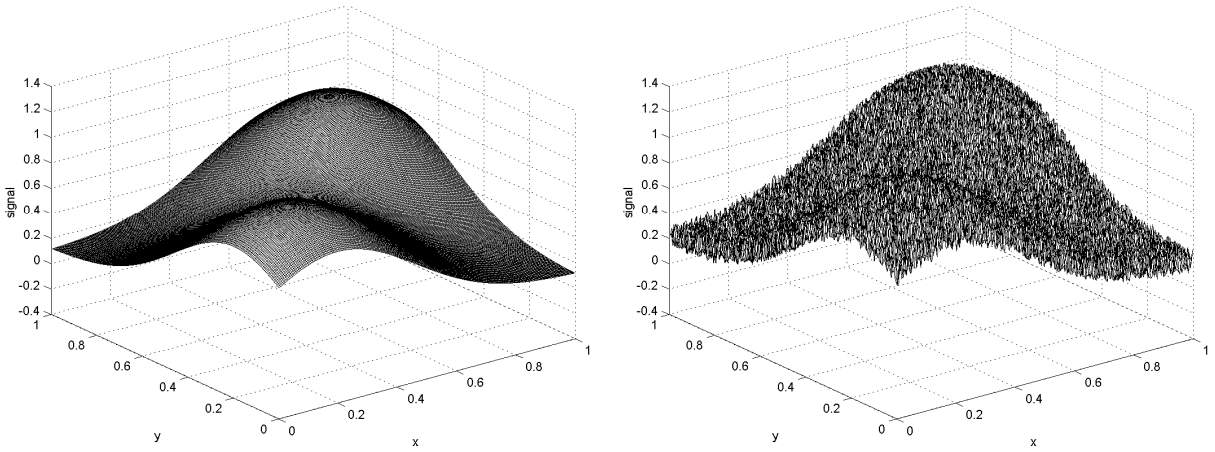
[2] http://www.hurray.isep.ipp.pt/activities/widom/getfile.aspx?file=interpolation.zip.

[3] B. Andersson, N. Pereira, and E. Tovar. Using a prioritized MAC protocol to efficiently compute aggregated quantities in a single broadcast domains. In *IPP-HURRAY Technical Report - TR-061102*, 2006. Available at http://www.hurray.isep.ipp.pt/activities/widom/GetFile.aspx?File=tr-hurray-0601102.pdf.

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

[5] 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.

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



(a) Original Signal

(b) Original Signal with Noise

(c) WAI w/ Carefully Selected Points

(d) WAI w/ Randomly Selected Points

Figure 4. Interpolation Example 2.

[7] 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.

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

[9] 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*, pages 123– 131, 2005.