Using a Prioritized MAC Protocol to Efficiently Compute Aggregated Quantities in a Single Broadcast Domain

Björn Andersson

Nuno Pereira

Eduardo Tovar

TR-061102

Version: 1.0

Date: Nov 2006

Using a Prioritized MAC Protocol to Efficiently Compute Aggregated Quantities in a Single Broadcast Domain

Björn ANDERSSON, Nuno PEREIRA, 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, nap, emt}@isep.ipp.pt

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

Abstract

Consider a wireless sensor network where all nodes share a single broadcast domain. Sensor nodes take sensor readings but individual sensor readings are not very important. It is important however to compute aggregated quantities of these sensor readings. We show that a prioritized medium access control (MAC) protocol for wireless broadcast is useful for efficiently computing aggregated quantities. We present algorithms for computing aggregated quantities with a time complexity that is independent of the number of sensor nodes. We present algorithms for computing MIN and MAX and propose approximation algorithms for COUNT and MEDIAN. We show that if every sensor node knows its geographical position, then sensor data can be interpolated and the time complexity of this interpolation algorithm does not depend on the number of sensor nodes. Such an interpolation of sensor data can be used to compute any function desired; for example, the temperature gradient in a room densely populated with sensor nodes.

Using a Prioritized MAC Protocol to Efficiently Compute Aggregated Quantities in a Single Broadcast Domain

Björn Andersson
IPP-Hurray! Research Group
Polytechnic Institute Porto
Porto, Portugal
bandersson@dei.isep.ipp.pt

Nuno Pereira
IPP-Hurray! Research Group
Polytechnic Institute Porto
Porto, Portugal
nap@isep.ipp.pt

Eduardo Tovar
IPP-Hurray! Research Group
Polytechnic Institute Porto
Porto, Portugal
emt@dei.isep.ipp.pt

ABSTRACT

Consider a wireless sensor network where all nodes share a single broadcast domain. Sensor nodes take sensor readings but individual sensor readings are not very important. It is important however to compute aggregated quantities of these sensor readings. We show that a prioritized medium access control (MAC) protocol for wireless broadcast is useful for efficiently computing aggregated quantities. We present algorithms for computing aggregated quantities with a time complexity that is independent of the number of sensor nodes. We present algorithms for computing MIN and MAX and propose approximation algorithms for COUNT and MEDIAN. We show that if every sensor node knows its geographical position, then sensor data can be interpolated and the time complexity of this interpolation algorithm does not depend on the number of sensor nodes. Such an interpolation of sensor data can be used to compute any function desired; for example, the temperature gradient in a room densely populated with sensor nodes.

Categories and Subject Descriptors

C.2.1 [**Computer-Communication Networks**]: Network Architecture and Design—*Wireless communication*; C.2.5 [**Computer-Communication Networks**]: Local and Wide-Area Networks—*Access scheme*

General Terms

Algorithms, Performance, Theory

Keywords

wireless sensor networks, medium access control protocols, aggregated quantities, signal processing, sensor fusion.

1. INTRODUCTION

Sensor networks often take many sensor readings of the same type (for example, temperature readings), and instead of knowing each individual reading it is important to know aggregated quantities of these sensor readings. For example, each sensor node senses the temperature at its location and we want to know the maximum temperature among all nodes at a particular moment.

Several solutions for data aggregation have been proposed for multihop networks. Typically, nodes organize themselves into a convergecast tree with a base station at the root [1, 2]. Leaf nodes broadcast their data. All other nodes wait until they have received a broadcast from all of its children; a node aggregates the data from its children and makes a single broadcast. Such a technique has been proposed for computing useful aggregated quantities such as MIN, MAX, COUNT and MEDIAN among a set of sensor nodes. It offers good performance because it exploits the opportunities for parallel transmission and the processing enroute makes the transmitted packet typically smaller than the sum of the size of the incoming packets.

Unfortunately these advantages are lost when all nodes share a *single broadcast domain*; that is, (i) a wireless broadcast made by one sensor node reaches all other sensor nodes and (ii) if a sensor node transmits a packet then it can be received by another sensor node only if the transmission of the packet does not overlap in time with another packet transmission.

Even a small broadcast domain (covering an area <10m) may contain a few hundreds of sensors [3]. Furthermore, local group communication between nodes of geographic proximity makes possible the aggregation and compression of locally generated sensing data before being transported to remote sinks, and, in effect, local group communication among sensors is one of the basic building blocks on many WSN applications [4].

In this paper we show that a prioritized MAC protocol for wireless broadcast can significantly improve the time-complexity for computing certain aggregated quantities in a single broadcast domain. In particular we show that the minimum value can be computed with a time complexity that does not depend on the number of sensor nodes. Also the time complexity increases very slowly as the possible range of the value increases. The same technique can be used to compute the maximum value. We also show how to compute a more complex aggregated quantity: the median. This computation hinges on the ability to count the number of nodes. We propose such a technique but it only gives an estimation and hence the median function is only estimated. Simulation shows however that the estimation is accurate; simulation of the execution in Avrora [5] also shows that the computational overhead required by the estimation technique is low.

It is often desired to know how physical quantities (such as temperature) vary over an area. Clearly the physical location of each sensor 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 com-

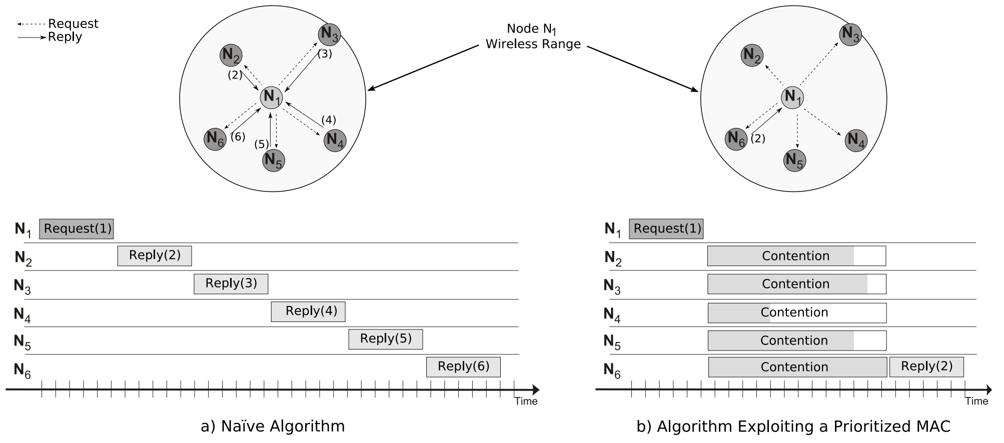


Figure 1: Motivating Example.

plexity of obtaining the interpolation is independent of the number of sensor nodes.

We consider this result to be significant because (i) sensor network are designed for 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 prioritized MAC protocols that support a very large range of priority levels and it is collision-free assuming that priorities are unique. Such a protocol has recently been proposed [6], implemented and tested [7] on a sensor network platform[1].

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 used. It also presents the system model that we use. We present algorithms for computing aggregated quantities for sensor data without location (in Section 3) and for sensor data with location (in Section 4). Section 5 discusses practical aspects of the algorithms and compares with previous work. Section 6 gives conclusions and future work.

2. PRELIMINARIES AND MOTIVATION

The basic premise for this work is the use of a prioritized MAC protocol for wireless medium. This implies that the MAC protocol assures that, of all nodes contending for the medium at a given moment, the ones with the highest priority gain access to the medium. As a result of the contention for the medium, all participating nodes will have knowledge of winner's priority. This is inspired on Dominance/Binary-Countdown [8] protocols, implemented for wired networks in the widely used CAN bus [9]. In our prioritized MAC protocol for wireless medium, lower priority values mean higher priority, which is also similar to Dominance/Binary-Countdown protocols. However, priorities are assumed to be unique in Dominance/Binary-Countdown protocols. We do not make that assumption.

The protocol in [6, 7] offers this behavior; Section 5.2 gives an overview of those protocols. The focus of this paper will be on exploiting a prioritized medium access control (MAC) protocol. We show that the availability of such a protocol enables novel distributed computations in sensor networks.

[1]The source code for the implementation can be found at `http://www.hurray.isep.ipp.pt/widom/`

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, it is required that m broadcasts are made. Considering that sensor networks are designed for large scale, dense networks [10, 3], the naïve approach can be inefficient; it causes a large delay and the long execution time wastes energy.

Let us consider a simple application scenario depicted in Figure 1a, where a node (node N_1 in Figure 1a), needs to know the minimum temperature reading among its neighbors. Let us assume no other node attempts to access the medium before this node. A naïve algorithm broadcasts a request to all its neighbors, and waits for replies from neighbor nodes. As a simplification, assume that nodes have set up a scheme to orderly access the medium in a time division multiple access (TDMA) manner, and that the initiator node knows the number of neighbor nodes, then it can compute a waiting timeout for replies based on this. Clearly, with this algorithm, the execution time depends on the number of neighbor nodes (m).

Consider now that we have a prioritized MAC protocol such as the one described in the beginning of Section 2. This scenario is depicted in Figure 1b. Assume that the range of the analog to digital converters (ADC) on the sensor nodes is known, and that the MAC protocol can, at least, represent as many priority levels. Now, to compute the minimum temperature among its neighbors, node N_1 (Figure 1b) needs to perform a request that will make all its neighbors 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[2] 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

[2]The different length of the gray bars inside the boxes depicting the contention in Figure 1b represent the amount of time that the node actively participated in the medium contention, see Section 5.2 for details.

the contention, nodes will know the priority of the winner, no more information needs to be transmitted by the winning node. If, for example, we wish that the winning node transmits information (like its location) in the data packet, then one can code the priority of the nodes with more information (for example, the node ID), 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 could be used to compute the maxium temperature reading. Instead of directly coding the priority with the temperature reading, nodes use the bitwise negation of the temperature reading as the priority. Upon completion of the medium access contention, given the winning priority, nodes may reverse the operation to know the maximum temperature value.

These are just two examples of efficiently computing aggregate quantities (in this case, MIN and MAX) using an algorithm that exploits a prioritized MAC. Another motivating application example could be an application were a node N_1 (Figure 1), upon detection of a loud sound, needs to know how many neighbors have also detected the loud sound. Based on the same basic principle we will (in Section 3) also present an algorithm to efficiently perform this. But first we need to specify the system model and notations.

2.2 System model

The sensor network consists of m sensor nodes (called *nodes* for short) where a node is given a unique identifier in the range $1..m$. Nodes do not have a shared memory; all data variables are local to each node.

Each node has a wireless transceiver and is able to transmit to or receive from a single wireless channel. Every node has an implementation of a prioritized MAC protocol with the characteristics described earlier. Nodes perform requests to transmit. Each transmission request has an associated priority. Priorities are integers in the range [0, $MAXV$], where a low number signifies a high priority. Let $NPRIOBITS$ denote the number of priority bits. It is the same for all nodes. Since $NPRIOBITS$ are used to denote the priority, the priority represents a number from 0 to $2^{NPRIOBITS} - 1$. Clearly, $MAXV=2^{NPRIOBITS} - 1$.

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 are in a single broadcast domain. This implies that there are no hidden nodes and the network provides reliable broadcast.

The operating system offers systems calls for interacting with other nodes. The `send` system call takes two parameters, one describing the priority of the packet and one describing the data bits 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` only unblocks when it has won a tournament and transmitted a packet. The function `send_empty` takes only one parameter and it is a priority. Interestingly, `send_empty` does not take any parameter describing the data packet. The system call `send_empty` also results in the MAC protocol performing the contention for the medium. But the behavior of `send_empty` is different when the node wins, it does not send anything. In addition, when the tournament 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 `send_empty` system call will be used in environments where two nodes may have the same priority and hence there may be more than one node that declares itself as a winner. This is acceptable since they do not send any data, so there is no collision of the data.

Algorithm 1 Estimating COUNT (the number of nodes)

```
Require: All nodes start Algorithm 1 simultaneously.
Input: *active* - a global boolean variable indicating if the
    node is considered in the COUNT
 1: function $nnodes(j$ : integer, $x$ : array[1..$k$] of integer)
    return a real
 2: $r$ : array[1..$k$] of integer
 3: $x$ : array[1..$k$] of integer
 4: $q$ : integer
 5: for $q \leftarrow 1$ to $k$
 6:     if ($active$ = TRUE) then
 7:         $r[q] \leftarrow random(0, MAXV)$
 8:     else
 9:         $r[q] \leftarrow MAXV$
10:     end if
11:     $x[q] \leftarrow send\_empty(r[q])$
12: end for
13: $est\_nodes \leftarrow ML\_estimation(x[1], x[2], ..., x[k])$
14: return $est\_nodes$ // the estimation of COUNT
```

The system call `send_and_rcv` is similar to `send_empty` in the sense that it returns the priority of the winner and it gives control back to the application even if the node lost contention. However, `send_and_rcv` sends a data packet if it wins and it always receives the packet that the winner transmitted; this packet is returned to the application calling the `send_and_rcv` function.

We also assume that each node has a function which generates a uniformly distributed random integer variable in the range [0, $MAXV$], denoted by `random`(0, $MAXV$).

Each node takes sensor readings. These readings are in the range $[MINV, MAXV]$ and it is assumed that $MINV = 0$.

3. SENSOR DATA WITHOUT LOCATION

In this section, we will assume that nodes take sensor readings but a sensor node does not know its location. Consequently, the aggregated quantity depends on the sensor data but not on the location of the node that took the sensor data. Let v_i denote the sensor reading on node N_i. Clearly, we wish to compute a function $f(v_1, v_2, \ldots, v_m)$.

Using the basic idea for computing MIN presented in Section 2, we will first (in Section 3.1) show how to estimate COUNT, that is the number of nodes. It can also be used to count the number of nodes with a certain attribute. We will use this (in Section 3.2) and show that it can be used to compute the MEDIAN.

3.1 Estimating COUNT

Section 3.3.1 presents an algorithm that estimates the

Algorithm 2 Function `ML_estimation`

```
Require: The division of two integers (as is done in line 6) returns a real number.
 1: function $ML\_estimation(x$ : array[1..$k$] of integer) return an integer
 2:     $v$ : array[1..$k$] of real
 3:     $sumv$, $q$ : integer
 4:     $sumv \leftarrow 0$
 5:     for $q \leftarrow 1$ to $k$
 6:         $v[q] \leftarrow ln\left(\frac{1}{1-\frac{x[q]}{MAXV}}\right)$
 7:         $sumv \leftarrow sumv + v[q]$
 8:     end for
 9:     return $\lceil \frac{k}{sumv} \rceil$
10: end function
```

number of nodes. Section 3.3.2 considers an interval and computes the a posteriori probability that the number of nodes is in this interval. Section 3.3.3 presents an experimental evaluation of the algorithm.

3.1.1 Estimating a Single Value

A prioritized MAC protocol lets all nodes know the priority of the winner but it is unknown how many nodes had this priority. For this reason, we cannot compute the number of nodes exactly; we can only estimate it.

Let us consider nodes that use random priorities. If the number of nodes is sufficiently large then the probability approaches 100% for the event that the minimum priority is 0. But if there is only one node, it is highly unlikely that the minimum among the random priorities is 0. From this observation, we can see that one can estimate the number of nodes from the minimum of random numbers; this is the approach we will take.

The pseudo code of the algorithm for estimating the number of nodes is shown in Algorithms 1 and 2. The main algorithm (Algorithm 1) assumes that all computer nodes start their execution simultaneously and uses a global boolean variable *active* as input, indicating if the node should be considered in the COUNT operation. When performing Algorithm 1, all nodes have *active* equal to TRUE and proceed in following way. First, on line 7, the algorithm generates a random number in the range [0, $MAXV$], then all nodes send their random number and find the minimum random number (line 11). This is performed k times. The line 13 computes the estimation of the number of nodes based on the minimum numbers obtained on line 11. Line 13 uses a function, shown in Algorithm 2. The design of the function in Algorithm 2 can be explained in terms of maximum-likelihood estimation. We omit those details in this conference version of the paper; see our Technical Report [11] for details.

3.1.2 Estimating an Interval

It is sometimes necessary to know the probability that the number of nodes is less than or equal to j_2. On the other hand, it is sometimes necessary to know if the number of nodes is greater than or equal to j_1. And we want to know this with a certain confidence. Since there is diversity in what application developers want, we propose a simple generic function that computes the probability that: $j_1 \le m \le j_2$, where j_1 and j_2 are parameters selected by the designer. If the probability is not large enough, then it is up to the application program to decrease j_1 or increase j_2, or perform an estimation with a larger k.

Algorithm 3 Computing MEDIAN

```
Require: All nodes start Algorithm 3 simultaneously.
 1: function $calcmedian(vi$ : integer) return an integer
 2:     $LB \leftarrow MINV$
 3:     $UB \leftarrow MAXV$
 4:     for $j \leftarrow 1$ to $log2(MAXV - MINV)$ do
 5:         $mid \leftarrow (LB + UB)/2$
 6:         $active \leftarrow vi \le mid$
 7:         $nVless \leftarrow$ call Algorithm 1
 8:         $active \leftarrow vi \ge mid$
 9:         $nVgreater \leftarrow$ call Algorithm 1
10:         if $nVless \le nVgreater$ then
11:             $LB \leftarrow mid$
12:         else
13:             $UB \leftarrow mid$
14:         end if
15:     end for
16:     return $mid$
17: end function
```

Such algorithm has been developed, based on the same principle as the number of nodes estimation presented in the previous section. Due to space considerations, the reader is referred to [11] for the exact description of such algorithm as well as its rationale.

3.1.3 Performance Evaluation

Let us find out how the error of the protocol varies as a function of m and k. A simulation of the algorithm proposed in this paper[3] was run with $k = 5$ and $k = 20$, for different numbers of nodes (m = 1, 4, 16, ..., 2^{16}). The boxplots in Figure 2 are presented in a logarithmic scale and depict the distribution of 1000 estimations for the different numbers of nodes. In these boxplots, the box stretches from 25th percentile to the 75th percentile. The value of the median is shown and depicted as a line across the box. The minimum values are depicted below the box and the maximum is above.

From the box plots in Figure 2, it is possible to observe that the quality of the estimation improves significantly by increasing k and the error is often acceptable for k=20.

3.2 Computing MEDIAN

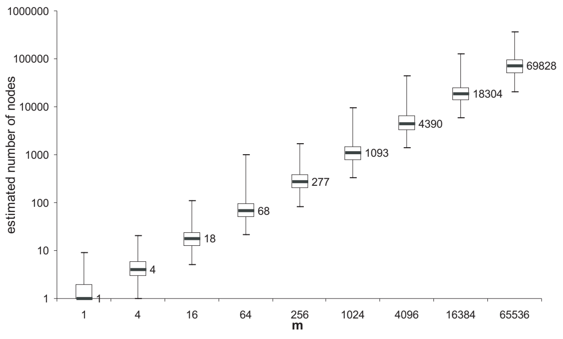
We now consider the case where the function that we want to compute is the median of v_1,v_2,...,v_m. Let us define $V_{less}(q)$ and $V_{greater}(q)$ as:

$$V_{less}(q) = \{v_j : v_j \le q\} \tag{1}$$

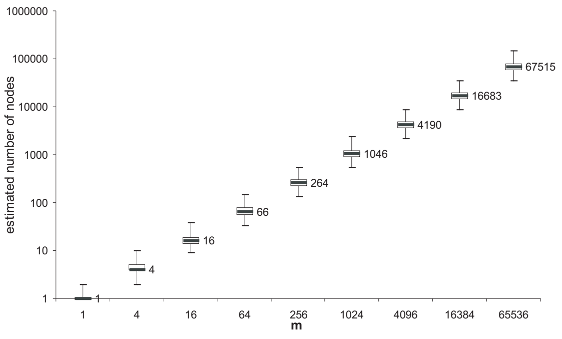
$$V_{greater}(q) = \{v_j : v_j \ge q\} \tag{2}$$

With these definitions our goal is to find q such that $||V_{greater}(q)| - |V_{less}(q)||$ is minimized. We can do this as follows. We know the that $0 \le q \le MAXV$. For this reason we know that a lower bound (LB) of q is 0 and an upper bound (UB) of q is $MAXV$. After that, we compute the midpoint (mid) between LB and UB. We count the number

[3]The source code for the simulation can be found at `http://www.hurray.isep.ipp.pt/widom/`



(a) $k = 5$



(b) $k = 20$

Figure 2: Estimation of the number of nodes for different values of m and k.

of nodes with a sensor reading in $[LB,mid)$ and the number of sensor readings in $[mid,UB]$. If there are more sensor readings in the former then we know that $q \in [LB,mid)$ and hence we set UB=mid and repeat the argument. If it is the latter then we know that $q \in [mid,UB]$ and hence we set LB=mid and repeat the argument. By repeating this process, we end up with LB=UB and then we know the median. We do not count the number of nodes exactly; we use Algorithm 1. Algorithm 3 presents how to compute the MEDIAN.

4. SENSOR DATA WITH LOCATION

In this section, we will assume that nodes take sensor readings just like we did in Section 3 but we will also assume that a sensor node knows its location. Naturally all techniques discussed in Section 3 can be used when a node know its location. But location-awareness also brings the possibility of computing new, more advanced aggregated quantities. It is possible to compute how the physical quantity varies over space. We will 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:

$$e_i = |v_i - f(x_i, y_i)| \tag{3}$$

and let e denote the global error; that is:

$$e = \max_{i=1..m} e_i \tag{4}$$

Our goal is to find $f(x,y)$ that minimizes e subject to the constraints that (i) the time required for computing f at a specific point is low and (ii) the time required to find the function $f(x,y)$ is low. The latter is motivated by the fact that we are interested in tracking physical quantities that change quickly; it may be necessary to recompute the interpolation every second in order to track how the concentration of hazardous gases move. For this reason, we will use weighted-average interplation (WAI) [12] (also used in [13, 14]). It interpolates as follows:

$$f(x,y) = \begin{cases} v_i & \text{if } \exists N_i \text{ with } x_i{=}x \text{ and } y_i{=}y; \\ \frac{\sum_{i \in S} v_i \cdot w_i(x,y)}{\sum_{i \in S} w_i(x,y)} & \text{otherwise.} \end{cases} \tag{5}$$

where $w_i(x, y)$ is given by:

$$w_i(x,y) = \frac{1}{(x_i - x)^2 + (y_i - y)^2} \tag{6}$$

Intuitively, the Equations 5 and 6 states that the interpolated value is a weighted sum 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 hardward support for floating point calculations. This is the case for typical sensor network platforms.

The original version of weighted-sum interpolation used all points; that is $S = \{1, 2, 3, \ldots, m\}$. But this would imply that computing Equations 5 and 6 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, only a small number of sensor readings are necessary in Equations 5. Hence, our goal is now to find those sensor nodes that contribute the most to the interpolation given by Equations 5. Recall that a prioritized MAC protocol can find the maximum among sensor readings. We can exploit this feature to find the node for which the magnitude of the error between the sensor reading and the interpolated value at the node is maximum. Such a node has a big impact on the interpolation.

Algorithm 4 is designed based on this principle. It computes (on line 9) the error. This error is used to compute a number and and the identifier of the node is concatenated; together this forms the priority of the message. This ensures that all priorities are unique. All nodes send their messages in parallel (on line 13), one packet will win the tournament and all nodes receive this packet. This packet is added (on line 26) to a set S which keeps track of all received packets related to the problem of creating an interpolation. If the node N_i did not win the tournament then it updates (on lines 19-24) the interpolated value at its position.

Algorithm 4 Finding a subset of nodes to be used in distance-weighted interpolationg

```
Require: All nodes start Algorithm 4 simultaneously.
 1: function find_nodes() return a set of packets
 2:     $MAXVerr \leftarrow MAXV/(MAXNNODES + 1)$
 3:     $myinterpolatedvalue \leftarrow 0$
 4:     $num \leftarrow 0.0$
 5:     $den \leftarrow 0.0$
 6:     $S \leftarrow \emptyset$
 7:     $update\_myinterpolation \leftarrow TRUE$
 8:     for $j \leftarrow 1$ to $k$ do
 9:         calculate $e_i$ according to Equation 3
10:         $prio\_error \leftarrow \frac{MAXVerr \times (MAXV - error)}{MAXV}$
11:         $prio \leftarrow prio\_error \times (MAXNNODES + 1) + i$
12:         $packet\_to\_send \leftarrow <v_i, x_i, y_i>$
13:         $< winning\_prio, rcv\_pack > \leftarrow send\_and\_rcv(prio, packet\_to\_send)$
14:         if $winning\_prio = prio$ then
15:             $update\_myinterpolation \leftarrow FALSE$
16:             $myinterpolatedvalue \leftarrow v_i$
17:         end if
18:         if $update\_myinterpolation = TRUE$ then
19:             $dx \leftarrow x_i - recv\_pack.x$
20:             $dy \leftarrow y_i - recv\_pack.y$
21:             $weight \leftarrow 1.0/(dx \times dx + dy \times dy)$
22:             $num \leftarrow num + recv\_pack.value \times weight$
23:             $denom \leftarrow denom + weight$
24:             $myinterpolatedvalue \leftarrow num/denom$
25:         end if
26:         $S \leftarrow S \bigcup rec\_packet$
27:     end for
28:     return $S$
29: end function
```

After nodes have found a subset of nodes that are to be used in the interpolation, it is possible to compute the value at any point. The set S that is used in Equation 5 is taken from Algorithm 4. It is straightforward to see that our interpolation algorithm has the time complexity O(k), where k is the number of nodes that are selected. Due to spatial locality, it is typically the case that k can be selected as $k << m$.

Figure 3 illustrates the operation of our new interpolation scheme. Figure 3(a) illustrates a signal that varies in space. We add noise and obtain the signal in Figure 3(b). Our algorithm[4] is used for selecting k=6 nodes and we interpolate between these nodes. The result is shown in Figure 3(c). The location of the nodes are indicated with vertical lines. We can see that the interpolation is smooth and it tracks the original signal well. However, performing weighted sum-interpolation with 6 sensor nodes selected randomly gives poor interpolation. This is illustrated in Figure 3(d).

Another example is given in Figure 11 with two peaks. We can see 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 [16]) we have found that the interpolation technique performs well as long as the signal does not change too abruptly when the location changes.

[4] The source code for the implementation of our interpolation algorithm can be found at `http://www.hurray.isep.ipp.pt/widom/`

5. DISCUSSION AND PREVIOUS WORK

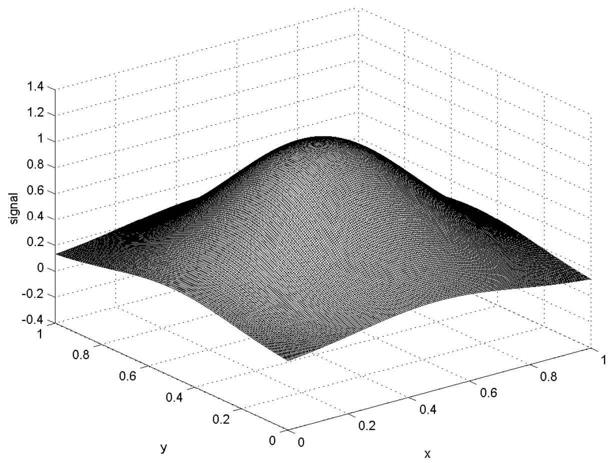
5.1 Previous work

5.1.1 MIN, MAX, General

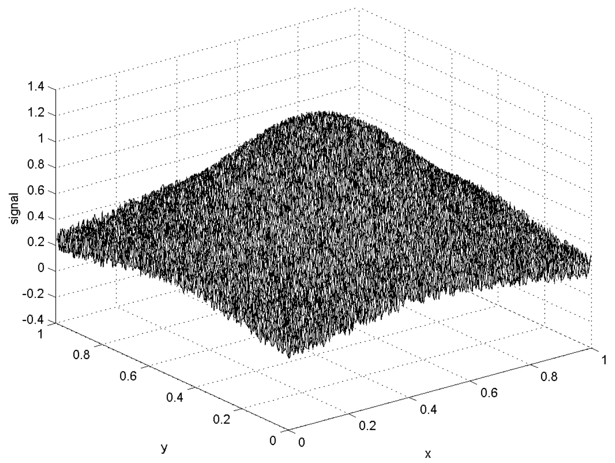
A prioritized MAC protocol is useful to schedule real-time traffic [6, 7] and it can support data dissemination when topology is unknown [6]. In this paper we have discussed how to efficiently compute aggregated quantities using a prioritized MAC protocol. Distributed calculations have been performed in previous research. It has been observed that nodes often [4, 17] detect an event and then needs to spread the knowledge of this event to its neighbors [4]. This is called one-to-k communication [4] because only k neighbors need to receive the message. After that, the neighbor nodes perform local computations and reports back to the node that made the request for 1-to-k communication. This reporting back is called k-to-1 communication. Algorithms for both 1-to-k and k-to-1 communication are shown to be faster than a naïve algorithm but unfortunately, the time-complexity increases as k increases. Our algorithms compute a function f and takes parameters from different nodes; this is similar to the average calculations in [18]. However our algorithms are different from [4, 17]; our algorithms have a time-complexity that does not depend on the number of nodes. We think our new algorithms are also useful building blocks for leader election and clock synchronization.

One way to use these algorithms is to encapsulate them in a query processor for database queries. Query processors for sensor networks have been studied in previous work [1, 2] but they are different in that they operate in multhop environment, and do not compute aggregated quantities as efficiently as we do. They assume one single sink node and that the other nodes should report an aggregated quantity to this sink node. The sink node floods its interest in the data it wants into the network and this also causes nodes to discover the topology. When a node has new data it, broadcasts this data; other nodes hear it, then it is routed and combined so that the sink node receives the aggregated. These works exploit the broadcast characteristics of the wireless medium (like we do) but they do not make any assumption on the MAC protocol (and hence they do not take advantage of the MAC protocol). One important aspect of these protocols is to create a spanning tree. It is known that computing an optimal spanning tree for the case when only a subset of nodes can generate data is equivalent to finding a Steiner-tree, a problem known to be NP-hard (the decision problem is NP-complete, see page 208 in [19]). For this reason, approximation algorithms have been proposed [20, 21]. However, in the average case, very simple randomized algorithms perform well [22]. Since a node will forward its data to the sink using a path which is not necessarily the shortest path to the sink, these protocols cause an extra delay. Hence, there is a trade-off between delay and energy-efficiency. To make this trade-off, a framework based on feedback was developed [23] for computing aggregated quantities. Techniques to aggregate data in the network such that the user at the base station can detect whether one node gives faked data has been addressed as well [24].

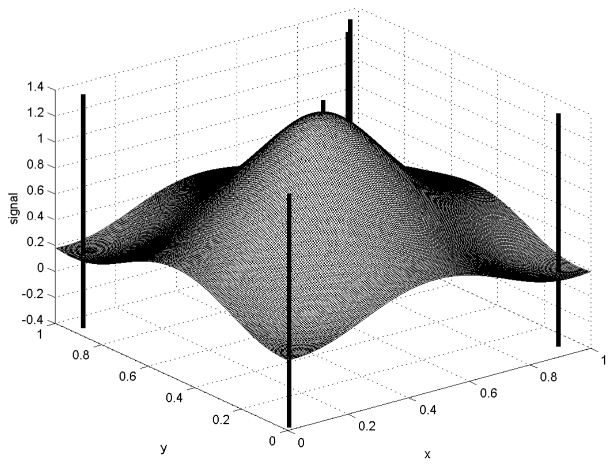
It has been observed that computing the median is especially difficult in multihop networks because combining two medians from different subnetworks requires a large amount of memory. Researchers in [24] observed that it is necessary



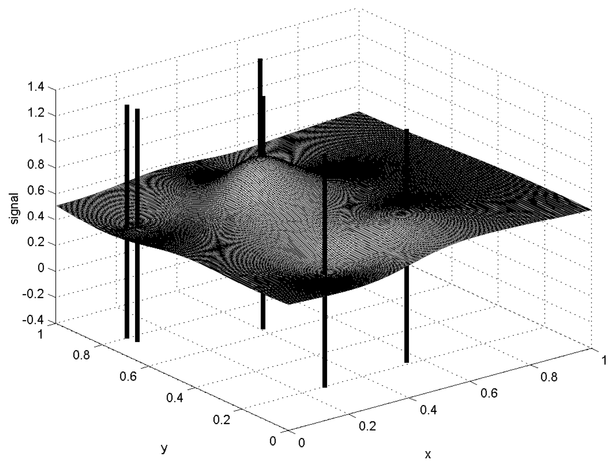
(a) Original Signal



(b) Original Signal with Noise



(c) WAI with Carefully Selected Data Points



(d) WAI with Randomly Selected Data Points

Figure 3: Interpolation.

for packets forwarded to be bigger and bigger the closer they get to the base station. Several algorithms for computing the exact median in $O(m)$ time complexity are available (the earliest one is [25]). Our algorithm is faster; it has the time complexity $O(log(MAXV - MINV))$ but at the expensive of the accuracy of the result.

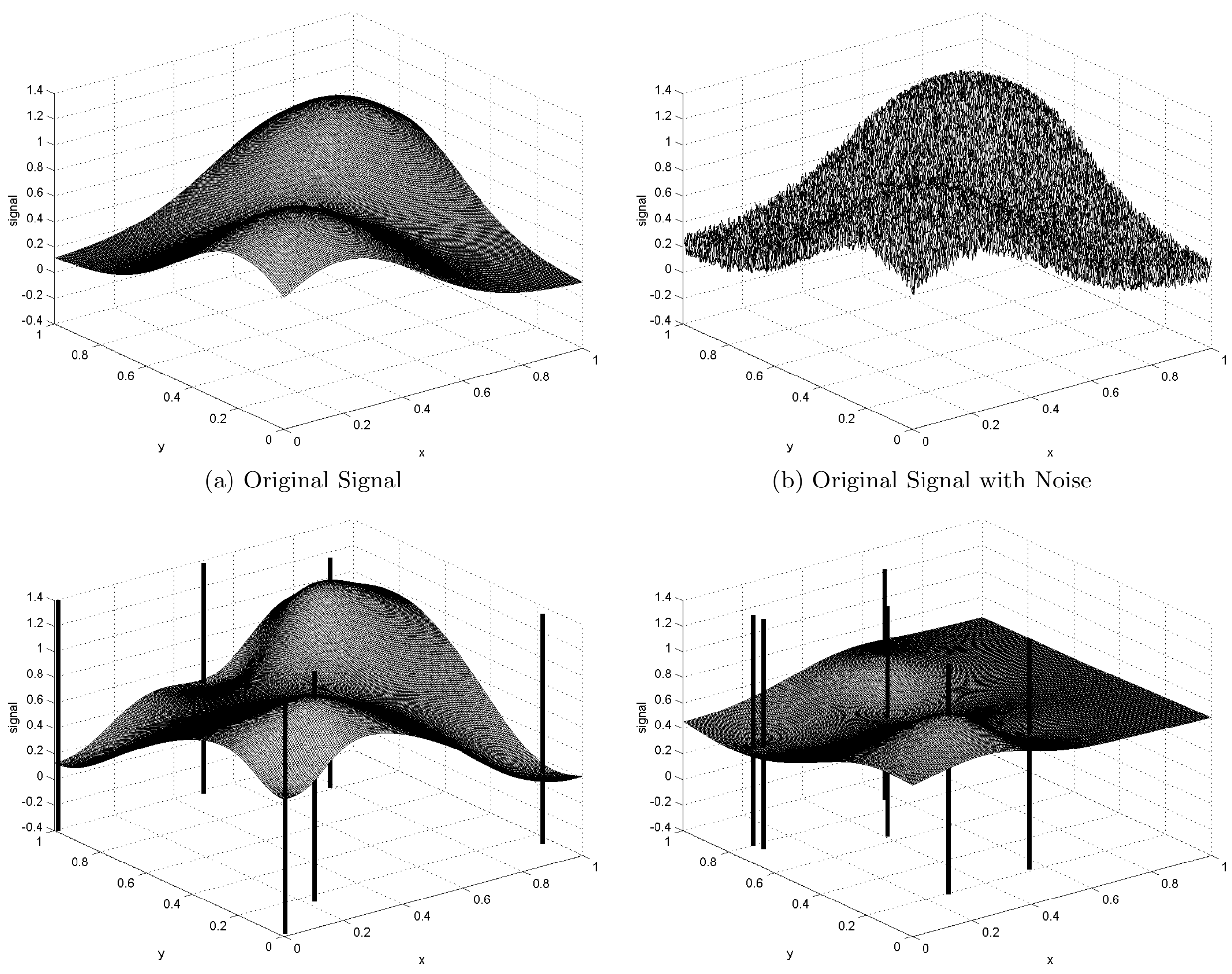
Computing averages has been done under the assumption that an adversary generates faults [26]. Unfortunately, it has a time-complexity which is larger than our algorithm and also larger than the algorithm proposed by [27].

5.1.2 COUNT

The problem of estimating the number of nodes in a network can be viewed from different perspectives. Gossip, rumour spreading and infectious algorithms, all have in common that they use randomized local computations repeatedly to achieve a global computation. Originally these algorithms were developed to propagate data, but recently they have been reworked to calculate aggregated quantities. These algorithms are robust in face of node and link failures and they can operate in multihop networks. Such algorithms are available for a large number of distributed calculations, such as MIN, MAX, SUM, AVERAGE (see for example [28, 29]). These calculations can be used to calculate/estimate the number of nodes as well. Two techniques are known. The first technique populates the value of zero on all nodes but one; this special node is populated the value of one. Then the protocol calculates the average and taking the inverse of the average gives the number of nodes. The second technique calculates the minimum value and applies maximum likelihood estimation to obtain the number of nodes.

Deterministic algorithms for unstructured environments have also been proposed. The algorithm in [18] performs repeated local operations to compute an average and it works in multihop environments. The algorithm in [30] computes the average in a single-hop network. It is designed to perform well against an adversary that injects faults but unfortunately, its time complexity is high. These techniques that compute averages could (as mentioned above) be used to compute the number of nodes.

Data aggregation protocols for WSN can compute the number of nodes, typically using a convergecast tree [1, 2]. The same problem has been addressed by researchers in data communications with the goal of estimating the size of the



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

Figure 4: Interpolation.

audience of a multicast [31, 32].

Common to all these works [28, 29, 18, 30, 1, 1, 31, 32] is that their time complexity is O(m) or more whereas our techique has a time complexity which is independent of m.

5.1.3 Interpolation

Formally speaking, interpolation requires that the function crosses all data points; if it only approximates the data points then it is called curve-fitting or regression. But previous work [13] in sensor networks use the word interpolation even if the function does not cross all data points and this is the reason why we call our algorithm an *interpolation algorithm*.

The problem of obtaining an interpolation to extract data from a sensor network is attracting increasing attention in the sensor network community [13, 14, 15]. But unfortunately, if those algorithms would be applied in a single broadcast domain then their time complexity would depend on the number of nodes.

5.2 Practical issues

We have assumed that all nodes start the execution of the protocol simultaneously. This can be dealt easily by letting a node broadcast a message containing a request to compute the number of nodes. All nodes receive this at approximately the same time. There are small differences in time when nodes start the protocol, but the MAC protocol (see [7]) synchronizes so that the tournament on all nodes executes simultaneously, so this poses no problem.

The MAC protocol exploited in this work was idealized based on an existing family of MAC protocols for wired networks. This family is named Dominace/Binary-Countdown [8] protocols.

In Dominace/Binary-Countdown [8], nodes perform a tournament as depicted in Figure 5 to access the medium. The nodes start by agreeing on an instant when the tournament starts. Then nodes transmit the priority bits starting with the most significant bit. A bit is assigned a time interval. If a node contends with a dominant bit ("0"), then a carrier wave is transmitted in this time interval; if the node contends with a recessive bit ("1"), it transmits nothing but listens. At the beginning of the tournament, all nodes have the potential to win, but if a node contends with a recessive bit and perceives a dominant bit then it withdraws from the

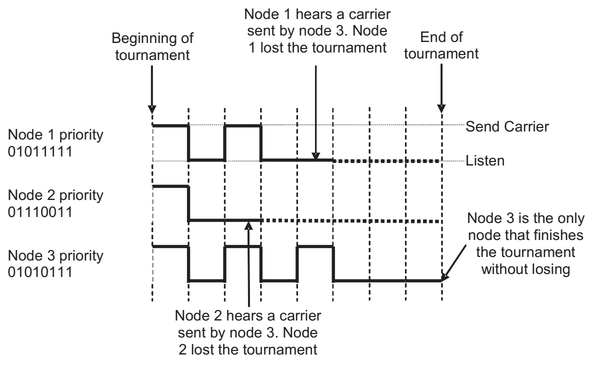

Beginning of tournament
Node 1 hears a carrier sent by node 3. Node 1 lost the tournament
End of tournament
Node 1 priority 01011111
Send Carrier
Listen
Node 2 priority 01110011
Node 3 is the only node that finishes the tournament without losing
Node 3 priority 01010111
Node 2 hears a carrier sent by node 3. Node 2 lost the tournament

Figure 5: The MAC protocol tournament.

tournament and cannot win. If a node has lost the tournament then it continues to listen in order to know the priority of the winner. When a node finishes sending all priority bits without hearing a dominant bit, then it has won the tournament and clearly knows the priority of the winner. Hence, lower numbers represent higher priorities.

To support the hipothesis of implementing a protocol with similar properties for wireless networks, we have referenced the reader to [7]. So far, the implementation of this prioritized MAC protocol for wireless networks introduces a significant amount of overhead. This overhead is to a large extent due to the transition time between transmission and reception. The platform used to implement the MAC protocol in [7] had a switching time of 192μs. But this is a technological parameter that can be improved with better radio hardware, as witnessed by the fact that the Hiperlan standard [33] required a switching time of 2μs.

But, despite the previous consideration, to show how the execution time of the algorithm compares with a naïve algorithm (as mentioned in Section 2), the time to execute the algorithm was acquired by using data from previous research [7], and running the proposed algorithm on a cycle accurate simulator for a mote platform, called Avrora [5]. From [7], it is possible to know that the time to run the tournament C_{trt} is 45 ms. Running the algorithm in Avrora, provided a measurement on the time to generate a random number ($\approx$ 0.003 ms, which together with all computations other than the estimation itself, was considered negligible) and the time to compute the function `ML_estimation`, as depicted in Algorithm 2 ($C_{est} \approx 86$ ms, for $k = 5$). Therefore, for $k = 5$, the time to perform the algorithm is $k \times C_{trt} + C_{est} = 5 \times 45 + 86 = 316$ ms.

Let us assume a very simplified model for assessing the overhead of the naïve algorithm. Only the time to transmit messages was considered and everything else is regarded as negligible. Considering a radio transmitting at 38.4 Kbps (a typical value for mote platforms [34]), a message with 2 bytes of data and 3 bytes for header/preamble would take $C_{msg} = \frac{(2+3)\times 8}{38400} \approx 1$ ms. The time to run a naïve protocol is then $m \times C_{msg}$.

From this, it is obvious that, with $k = 5$, our algorithm always runs faster when $m > 316$, and, more importantly, our protocol has a time complexity which only depends on k.

6. CONCLUSIONS

We have shown how to use a prioritized protocol to compute aggregated quantities efficiently. This is clearly important for applications that operate under real-time constraints. But, since the high speed makes it possible for nodes to stay awake for only a short time and they can then sleep, it is also very useful to reduce energy-consumption. This saves energy and contributes to a longer life-time. Also, our interpolation can be used to find areas that are "interesting"' and the interpolation can be used to "'zoom-in"' on only those areas. We left open the question on how to use a prioritized MAC protocol to compute aggregated quantities when the network is not a single broadcast domain.

7. REFERENCES

[1] Y. Yao and J. Gehrke. Query processing in sensor networks. In *Proceedings of the 1st Biennial Conference on Innovative Data Systems Research (CIDR'03)*, 2003.

[2] S. Madden, M. J. Franklin, J.M. Hellerstein, and W. Hong. TAG: a tiny aggregation service for ad-hoc sensor networks. In *Proceedings of the 5th symposium on Operating systems design and implementation (OSDI'02)*, 2002.

[3] W. R. Heinzelman, A. Chandrakasan, and H. Balakrishnan. Energy-Efficient Communication Protocol for Wireless Microsensor Networks. In *Proceedings of the 33rd Hawaii International Conference on System Sciences (HICSS'00)*, volume 8, pages 3005–3014, Maui, U.S.A., 2000.

[4] R. Zheng, L. Sha, and W. Feng. Mac layer support for group communication in wireless sensor networks. In *Proceedings of the second Mobile Adhoc and Sensor Systems Conference*, page 8. IEEE, 2005.

[5] AVRORA - the AVR simulation and analysis framework, 2005.

[6] B. Andersson and E. Tovar. Static-priority scheduling of sporadic messages on a wireless channel. In *Proceedings of the 9th International Conference on Principles of Distributed Systems (OPODIS'05)*, Pisa, Italy, 2005.

[7] N. Pereira, B. Andersson, and E. Tovar. Implementation of a dominance protocol for wireless medium access. In *Proceedings of the 12th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA'06)*, Sydney, Australia, 2006.

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

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

[10] A. Arora. Exscal: Elements of an extreme scale wireless sensor network. In *Proceedings of the 11th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA'05)*, pages 102–108, Washington, DC, USA, 2005. IEEE Computer Society.

[11] B. Andersson, N. Pereira, and E. Tovar. Estimating the number of nodes in wireless sensor networks. In *IPP-HURRAY Technical Report - TR-060702*, 2006. http://www.hurray.isep.ipp.pt/widom/hurray-tr-060702.pdf.

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

[13] 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.

[14] 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.

[15] C. Guestrin, P. Bodik, R. Thibaux, M. Paskin, and S. Madden. Distributed regression: an efficient framework for modeling sensor network data. In *Proceedings of IPSN*, 2004.

[16] 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.

[17] K. Jamieson, H. Balakrishnan, and Y. C. Tay. Sift: a MAC protocol for event-driven wireless sensor networks. In *Proceedings of the third European Workshop on Wireless Sensor Networks (EWSN'06)*, pages 260–275. IEEE, 2006.

[18] D.S. Scherber and H.C. Papadopoulos. Distributed computation of averages over ad hoc networks. *IEEE Journal on Selected Areas in Communications*, 23(4):776–787, 2005.

[19] M. R. Garey and D. S. Johnson. *Computers and Intractability A guide to the Theory of NP-Completeness.*

[20] B. Krishnamachari, D. Estrin, and S. B. Wicker. The impact of data aggregation in wireless sensor networks. In *Proceedings of the 22nd International Conference on Distributed Computing Systems (ICDCS'02)*, pages 575 – 578. IEEE, 2002.

[21] C. Intanagonwiwat, D. Estrin, R. Govindan, and J. Heidemann. Impact of network density on data aggregation in wireless sensor networks. In *Proceedings of the 22nd International Conference on Distributed Computing Systems (ICDCS'02)*, page 457, Washington, DC, USA, 2002. IEEE Computer Society.

[22] M. Enachescu, A. Goel, R. Govindan, and R. Motwani. Scale-free aggregation in sensor networks. *Theoretical Computer Science*, 344(1):15–29, 2005.

[23] T. Abdelzaher, T. He, and J. Stankovic. Feedback control of data aggregation in sensor networks. In *Proceedings of the 43rd IEEE Conference on Decision and Control (CDC'04)*, pages 1490–1495 Vol.2. IEEE Computer Society, 2004.

[24] B. Przydatek, D. Song, and A. Perrig. SIA: Secure information aggregation in sensor networks. In *Proceedings of the 1st ACM International Conference on Embedded Networked Sensor Systems (SenSys'03)*, pages 255–265, 2003.

[25] N. Shrivastava, C. Buragohain, D. Agrawal, and S. Suri. Medians and beyond: new aggregation techniques for sensor networks. In *Proceedings of the 2nd ACM International Conference on Embedded Networked Sensor Systems (SenSys'04)*, pages 239–249, New York, NY, USA, 2004. ACM Press.

[26] M. Blum, R.W. Floyd, V. Pratt, R. Rivest, and R. Tarjan. Time bounds for selection. *J. Comput. System Sci.*, 7:448–461, 1973.

[27] M. Kutylwski and D. Letkiewicz. Computing average value in ad hoc networks. In *Proceedings of the 28th International Symposium on Mathematical Foundations of Computer Science (MFCS'03)*, pages 511–520, 2004.

[28] D. Kempe, A. Dobra, and J. Gehrke. Gossip-based computation of aggregate information. In *Proceedings of the 44th Annual IEEE Symposium on Foundations of Computer Science (FOCS'03)*, pages 482–491, Washington, DC, USA, 2003. IEEE Computer Society.

[29] M. Jelasity, A. Montresor, and O. Babaoglu. Gossip-based aggregation in large dynamic networks. *ACM Transactions on Computer Systems*, 23(3):219–252, 2005.

[30] M. Kutylwski and D. Letkiewicz. Computing average value in ad hoc networks. In *Proceedings of the 28th International Symposium on Mathematical Foundations of Computer Science (MFCS'03)*, pages 511–520. Springer, 2003.

[31] K. Horowitz and D. Malkhi. Estimating network size from local information. *Information Processing Letters*, 88(5):237–243, 2003.

[32] M. Nekovee, A. Soppera, and T. Burbridge. An adaptive method for dynamic audience size estimation in multicast. In *Lecture Notes in Computer Science*, volume 2816, pages 23–33, 2003.

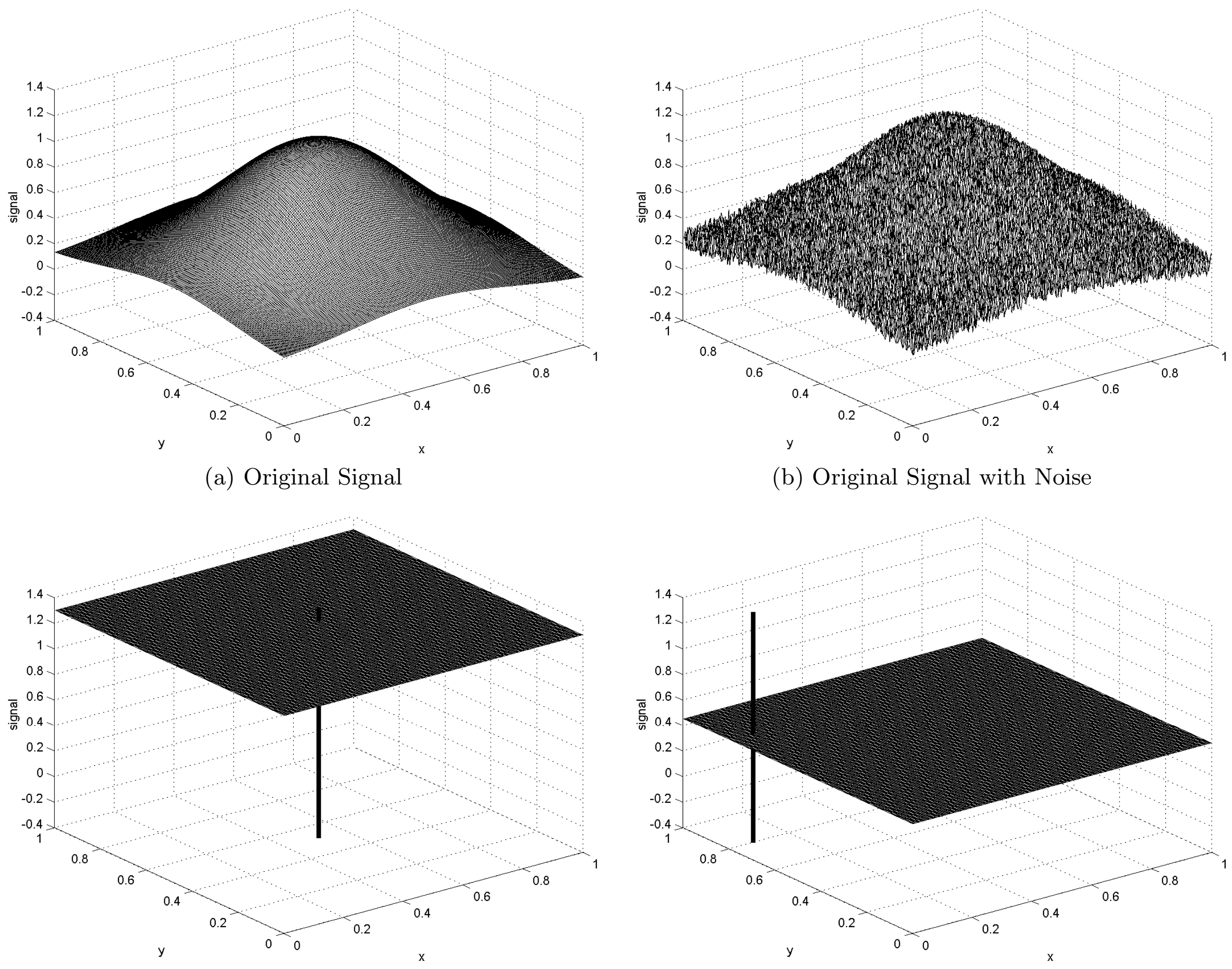
[33] ETSI (European Telecommunications Standards Institute). *Broadband Radio Access Networks(BRAN); HIPER-ACCESS; PHY protocol specification.*

[34] Crossbow. MICA2 - wireless measurement system product datasheet, 2005.

APPENDIX

A. THE INTERMEDIATE STEPS OF OUR INTERPOLATION SCHEME.

In this section, we study the operation of our interpolation scheme in more detail. We will see how our algorithm operates on the example given previously.



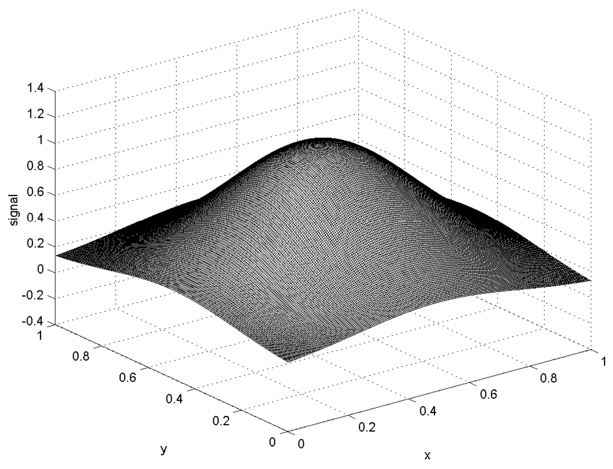
(a) Original Signal

(b) Original Signal with Noise

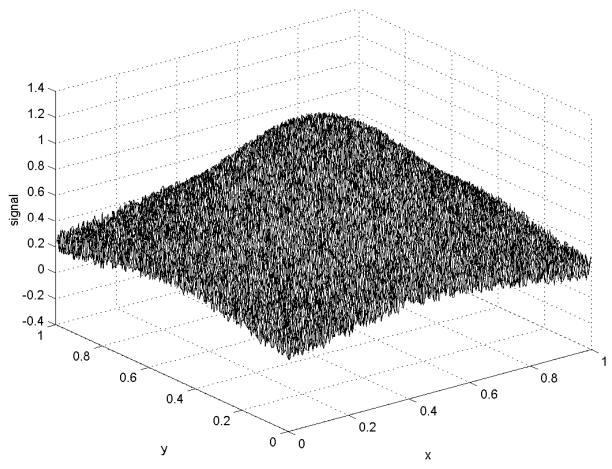
(c) WAI with Carefully Selected Data Points

(d) WAI with Randomly Selected Data Points

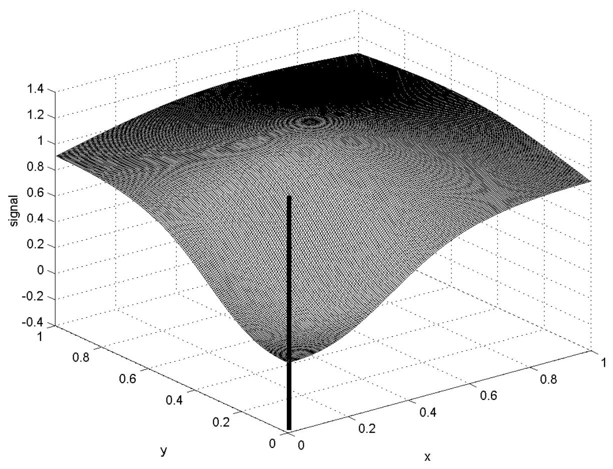
Figure 6: Interpolation k=1.



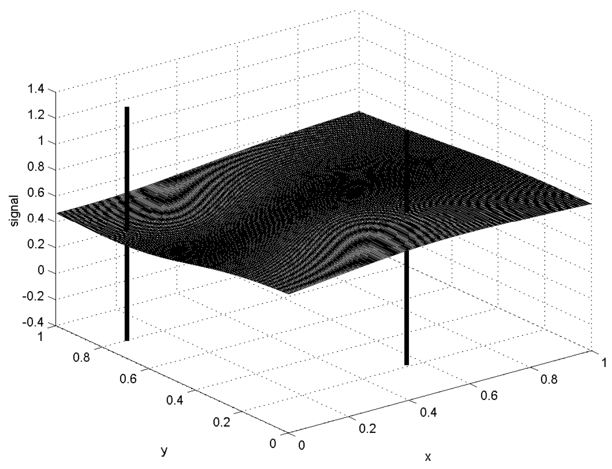
(a) Original Signal



(b) Original Signal with Noise

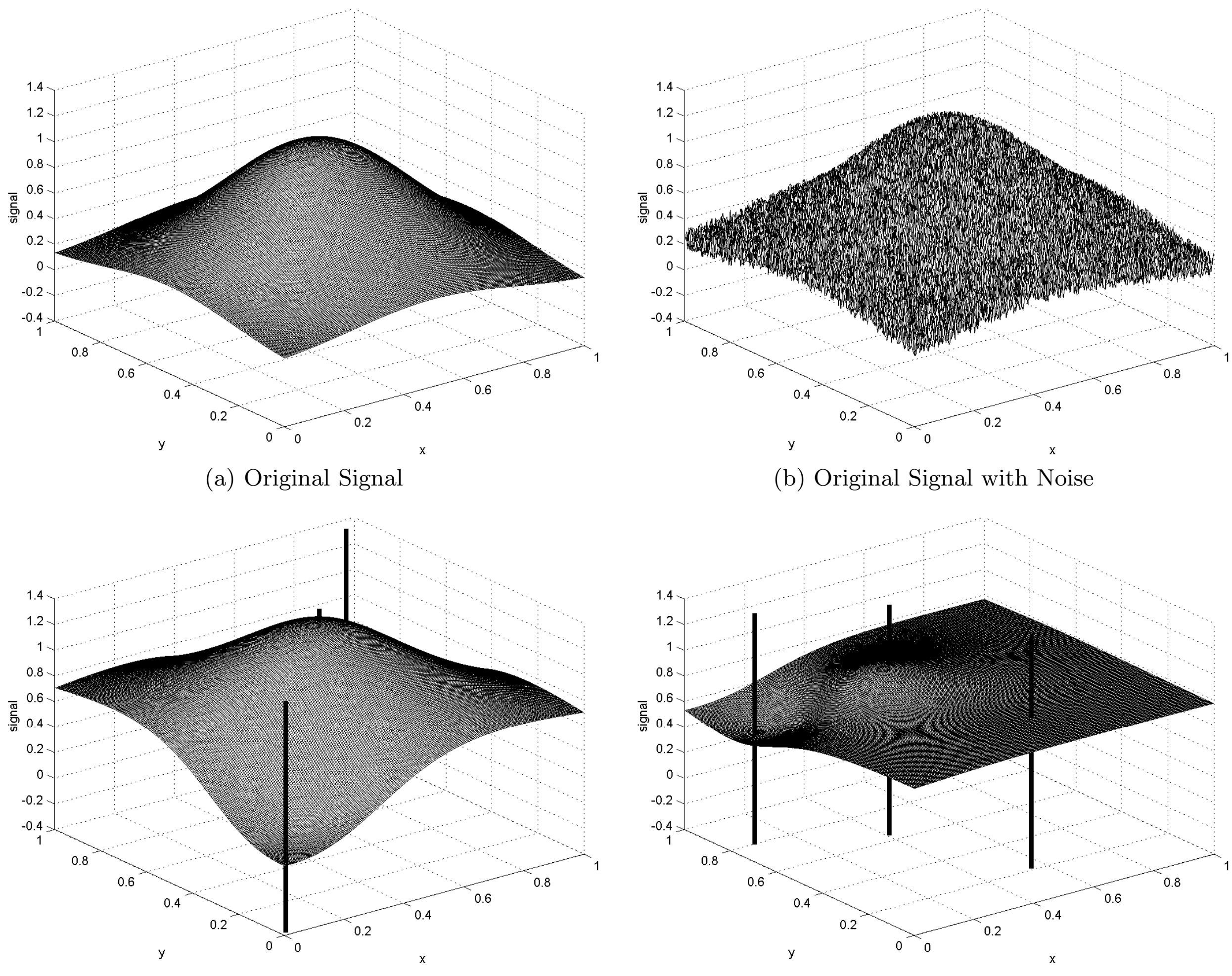


(c) WAI with Carefully Selected Data Points



(d) WAI with Randomly Selected Data Points

Figure 7: Interpolation k=2.



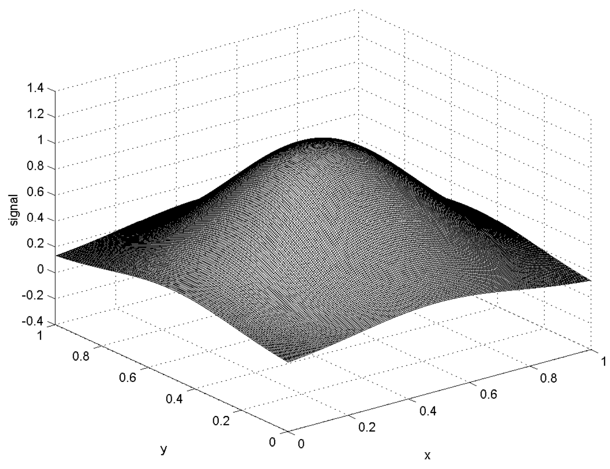
(a) Original Signal

(b) Original Signal with Noise

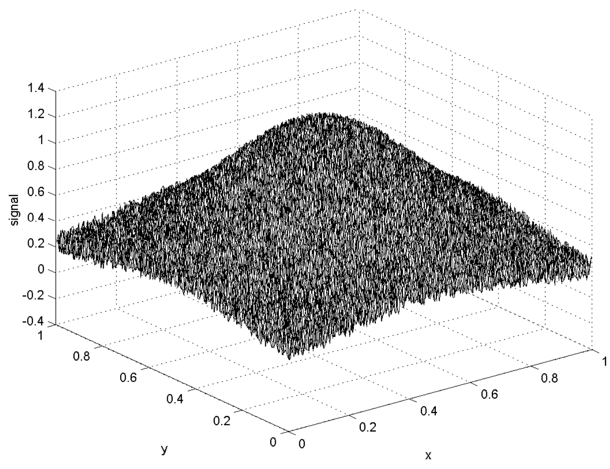
(c) WAI with Carefully Selected Data Points

(d) WAI with Randomly Selected Data Points

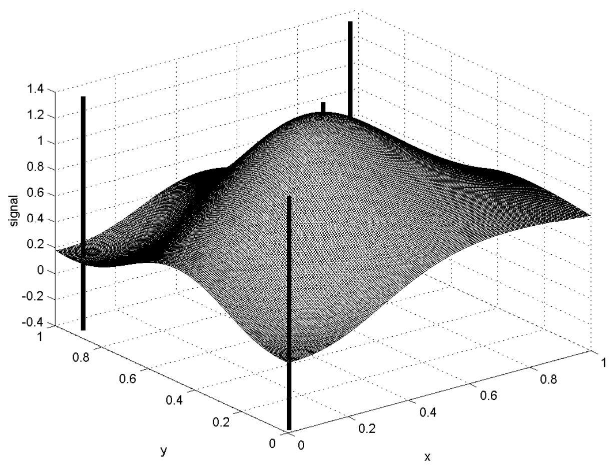
Figure 8: Interpolation k=3.



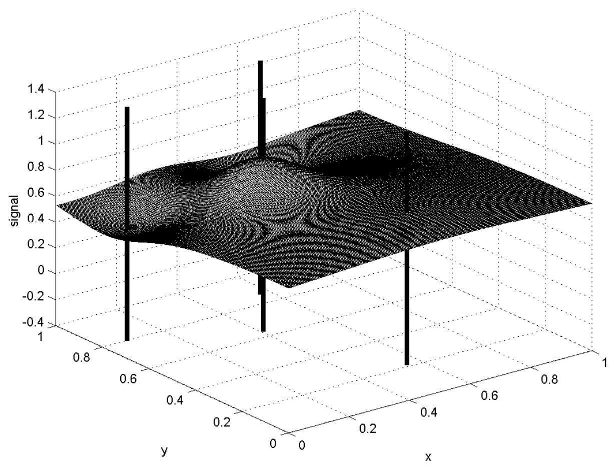
(a) Original Signal



(b) Original Signal with Noise

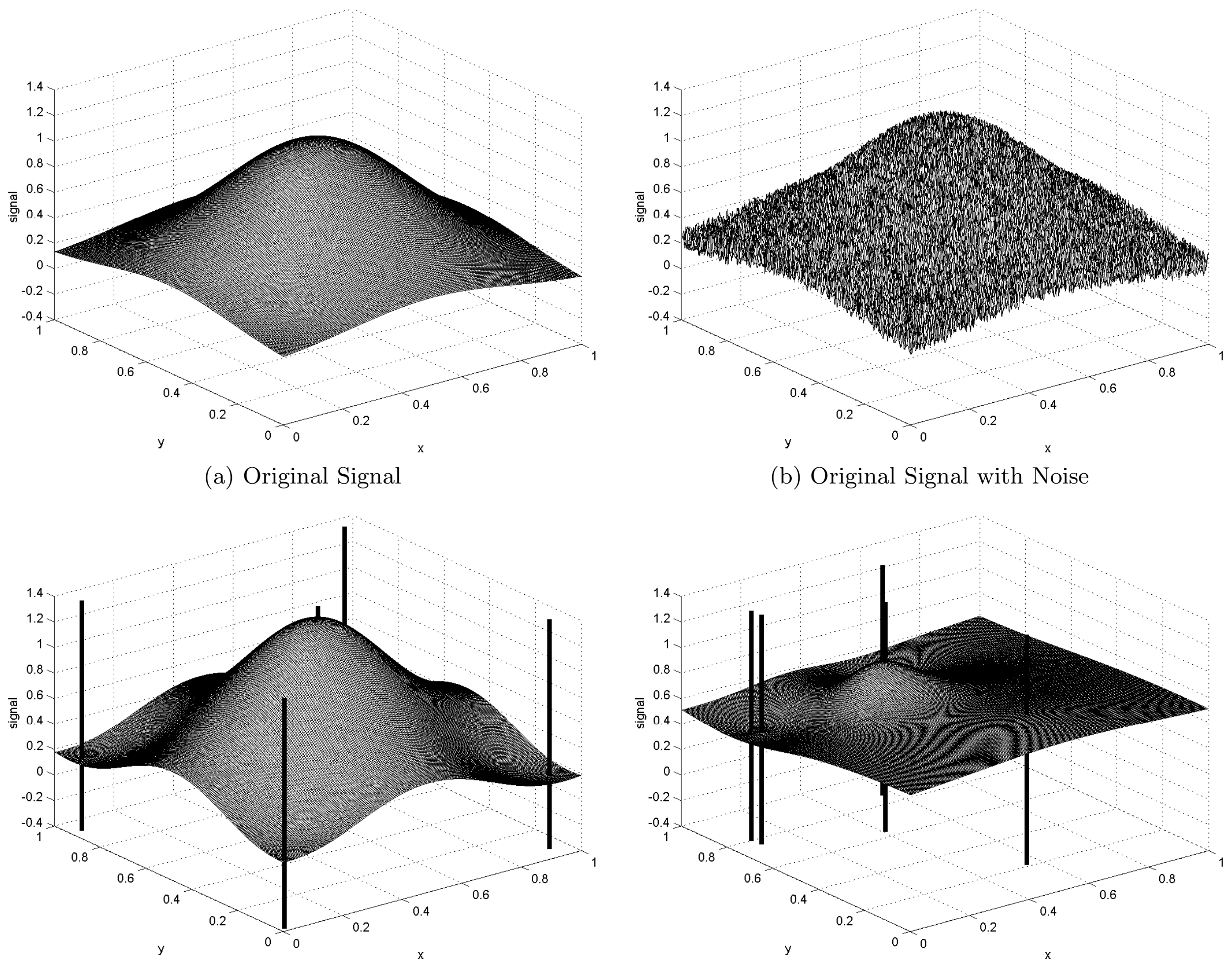


(c) WAI with Carefully Selected Data Points



(d) WAI with Randomly Selected Data Points

Figure 9: Interpolation k=4.



(a) Original Signal

(b) Original Signal with Noise

(c) WAI with Carefully Selected Data Points

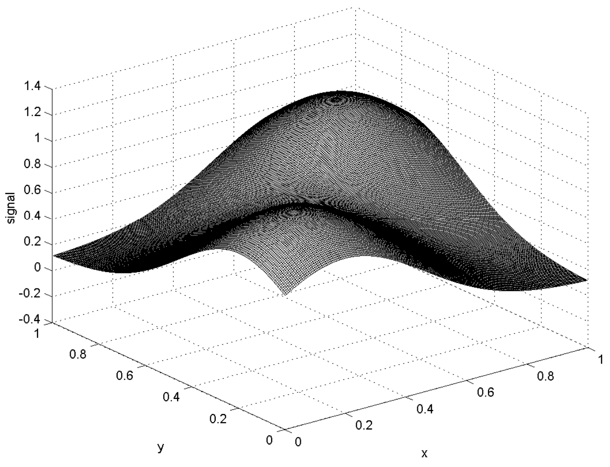
(d) WAI with Randomly Selected Data Points

Figure 10: Interpolation k=5.

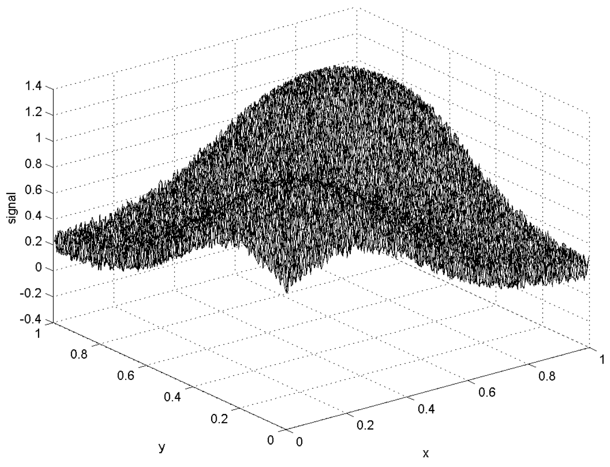
B. APPENDIX B.

In this section, we study the operation of our interpolation scheme in more detail. We will explore other signals forms and see that our scheme performs well even for them as long as the signal does not change too abrupt when the location changes.

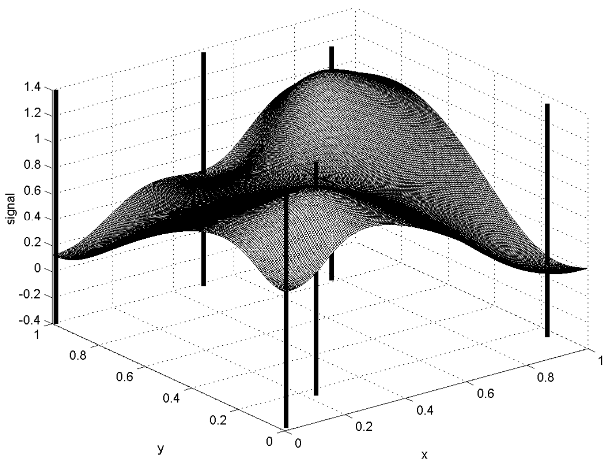
B.1 Twin peaks



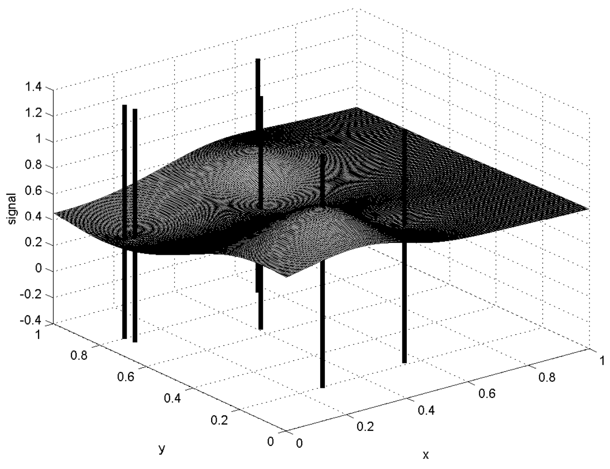
(a) Original Signal



(b) Original Signal with Noise



(c) WAI with Carefully Selected Data Points

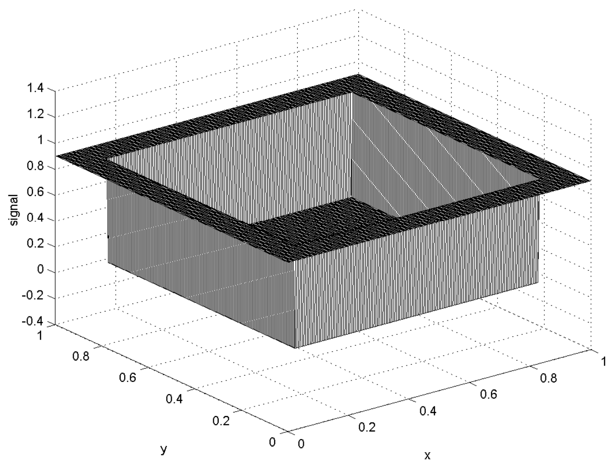


(d) WAI with Randomly Selected Data Points

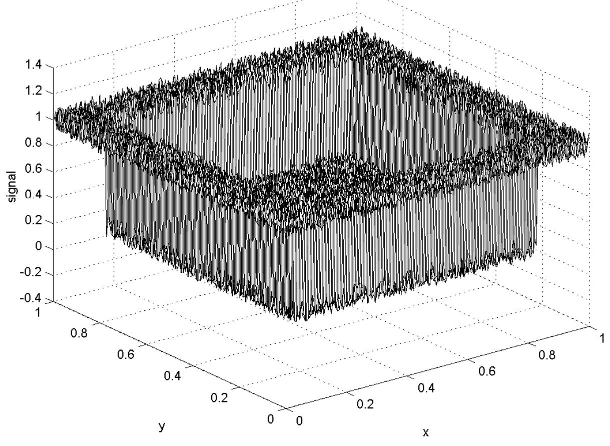
Figure 11: Interpolation.

B.2 Swimming pool

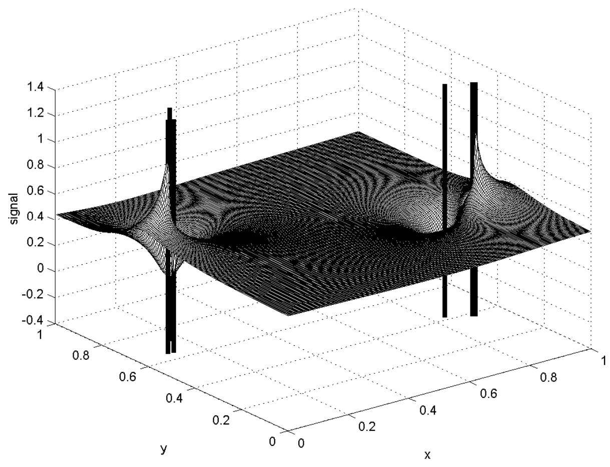
B.3 2-step Swimming pool



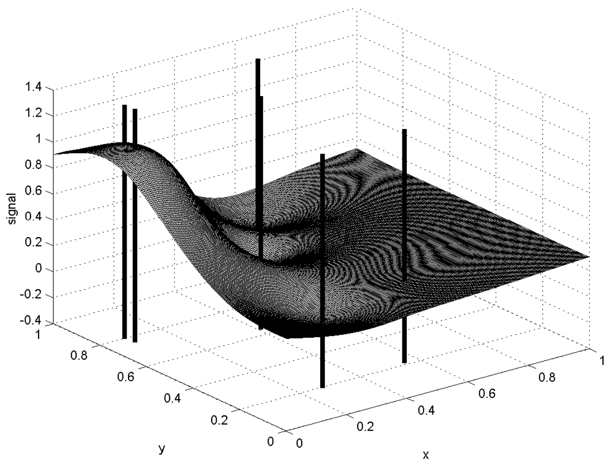
(a) Original Signal



(b) Original Signal with Noise

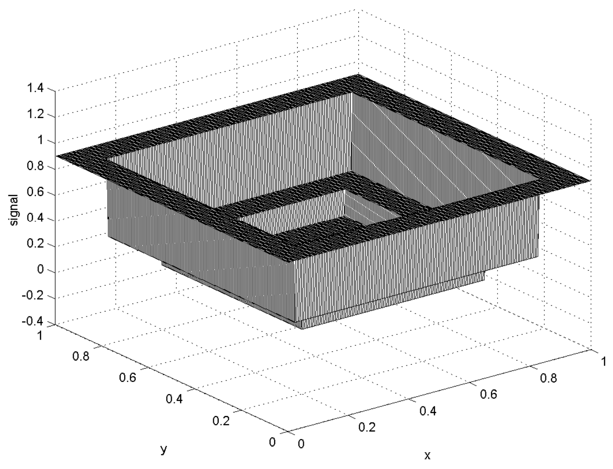


(c) WAI with Carefully Selected Data Points

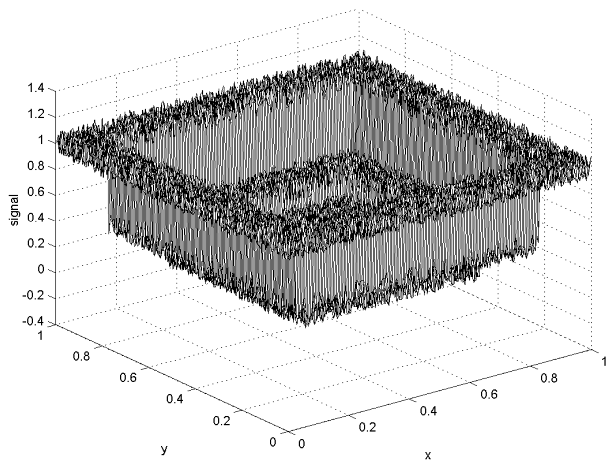


(d) WAI with Randomly Selected Data Points

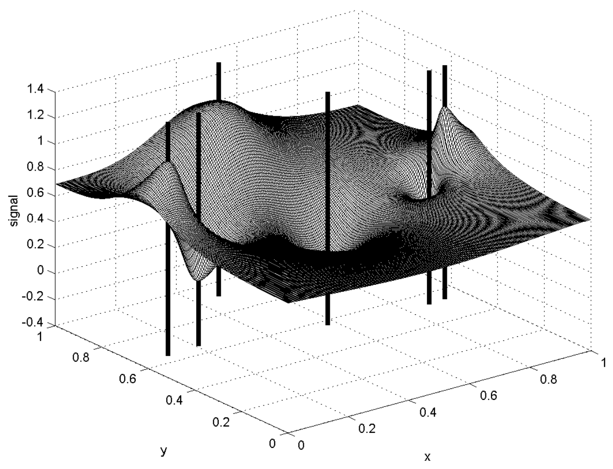
Figure 12: Interpolation.



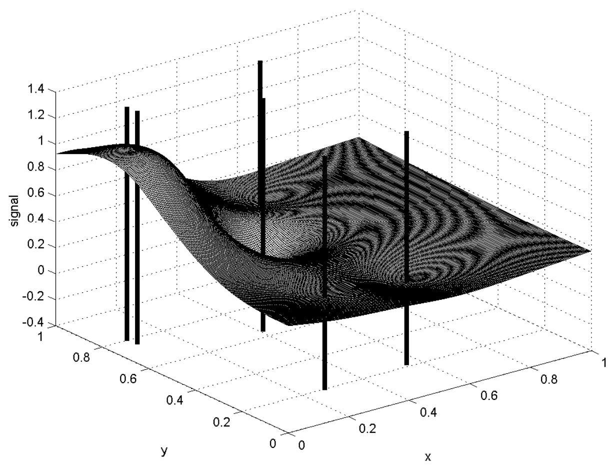
(a) Original Signal



(b) Original Signal with Noise



(c) WAI with Carefully Selected Data Points



(d) WAI with Randomly Selected Data Points

Figure 13: Interpolation.