Technical Report

Scalable and Efficient Data Processing in Networked Control Systems

Aida Ehyaei

Eduardo Tovar

Nuno Pereira

HURRAY-TR-101004

Version:

Date: 10-08-2010

Scalable and Efficient Data Processing in Networked Control Systems

Aida Ehyaei, Eduardo Tovar, Nuno Pereira

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:

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

Abstract

Network control systems (NCSs) are spatially distributed systems in which the communication between sensors, actuators and controllers occurs through a shared band-limited digital communication network. However, the use of a shared communication network, in contrast to using several dedicated independent connections, introduces new challenges which are even more acute in large scale and dense networked control systems. In this paper we investigate a recently introduced technique of gathering information from a dense sensor network to be used in networked control applications. Obtaining efficiently an approximate interpolation of the sensed data is exploited as offering a good trade-off between accuracy in the measurement of the input signals and the delay to the actuation. These are important aspects to take into account for the quality of control. We introduce a variation to the state-of-the-art algorithms which we prove to perform relatively better because it takes into account the changes over time of the input signal within the process of obtaining an approximate interpolation.

Scalable and Efficient Data Processing in Networked Control Systems

Aida Ehyaei, Eduardo Tovar, Nuno Pereira
IPP-HURRAY!
Polytechnic Institute of Porto (ISEP-IPP)
{aaei,emt,nap,baa}@isep.ipp.pt

Abstract

Network control systems (NCSs) are spatially distributed systems in which the communication between sensors, actuators and controllers occurs through a shared band-limited digital communication network. However, the use of a shared communication network, in contrast to using several dedicated independent connections, introduces new challenges which are even more acute in large scale and dense networked control systems. In this paper we investigate a recently introduced technique of gathering information from a dense sensor network to be used in networked control applications. Obtaining efficiently an approximate interpolation of the sensed data is exploited as offering a good trade-off between accuracy in the measurement of the input signals and the delay to the actuation. These are important aspects to take into account for the quality of control. We introduce a variation to the state-of-the-art algorithms which we prove to perform relatively better because it takes into account the changes over time of the input signal within the process of obtaining an approximate interpolation.

1. Introduction

Modern control theory is mostly based on the abstraction that information (signals) is transmitted through perfect communication channels and that computation is either instantaneous (continuous time) or periodic (discrete time) [2]. This abstraction has served the field well for over 50 years, and has led to many success stories in wide variety of applications.

However, emerging applications of control will be much more information-rich than those of the past and will involve massively networked communications, distributed computing, and higher levels of logic and decision-making. New theory, algorithms, and technology must therefore be developed, and the design of Networked Control Systems (NCS) needs to combine information theory, computer science, physics, control and other disciplines in a much tighter way than ever before for progressing in this field.

A typical NCS is composed of four basic elements: Sensors, Controllers, Actuators, and Communication networks. In an NCS, control loops are closed through a real-time network. Control and feedback signals are exchanged among the system's components in the form of messages through the communication network.

Wireless communication is starting to play an increasingly important role in NCS. Transmitting sensor measurements and control commands over wireless links allows rapid deployment, flexible installation and fully mobile operation. Also prevents the cable wear and tear problem. Building a networked control system over a wireless medium is however a challenging task. The scarce spectrum imposes a fundamental limit on the performance of the wireless channel. Random delays and packet losses are inevitable. Even though these challenges exist for any communication network, they are much more significant in wireless networks due to limited spectrum and power, time-varying channel gains and interference [3].

The other important concern in distributed wireless networks is gathering data from nodes, especially in dense networks. Data aggregation methods can be used to combine data of several nodes into a single message, reducing the number of transmitted messages within the network and, accordingly, the communications' energy consumption. This is achieved at the expense of message delays, since each node must wait to receive messages from all (or some) of its neighbors for aggregating. Thus, a main concern in data aggregation protocols is finding a proper balance between the communication (energy) and delay costs [4].

The time-complexity of data gathering protocols is heavily dependent on the number of nodes in the overall network. Multiple broadcast domains offer the opportunity for parallel transmissions and may reduce the time-complexity, depending upon the scale and topology of the network. This is however not the case of densely instrumented systems where even a very small area may contain several hundreds of nodes. To face these challenges, recent research efforts [15] have been proposing novel approaches for quantity aggregation in very dense networks.

These approaches are based on the intelligent exploitation of Dominance / Binary-Countdown Medium Access Control (MAC) protocols [8]. By associating the priorities of messages to physical quantities (such as temperature or acceleration), several high performance

algorithms for data processing can be devised in which time-complexity is independent of the number of nodes.

In this paper we will evaluate the quality of these quantity aggregation methods within networked control applications with densely deployed input nodes. In this paper, we will also propose an improved version of the distributed algorithm able to better track densely sensed systems in networked control systems.

The rest of this paper is organized as follows. In Section 2 we briefly survey the principles behind Dominance / Binary-Countdown MAC Protocols. In Section 3 we describe quantity aggregation and approximate interpolation of data by using Dominance MAC protocols in densely deployed sensor networks. In Section 4 previous interpolation algorithms are evaluated and a novel algorithm is proposed. This algorithm performs relatively better because it takes into account the changes over time of the input signal within the process of obtaining an approximate interpolation. Finally, in Section 5 conclusions are drawn and some future works are outlined.

2. Basic Principles of Dominance MAC Protocols

Dominance-based or binary-countdown protocols [8] are an important family of MAC protocols. These protocols have good properties for supporting timeliness in systems with event-triggered messages. Moreover, they are capable of simultaneous "non-destructive" transmission of information in the same broadcast domain. This is an important characteristics for the approaches described in this paper.

The wired implementation of this protocol is widely used in the Controller Area Network (CAN) bus [9]. In CAN, messages have a unique contention field which could be their priority. When a node has a request to transmit, after waiting a predetermined time until the channel becomes idle, it starts a conflict resolution phase (arbitration phase). In this phase, the nodes send their contention field, bit-by-bit, starting from the most significant bit. The medium is devised in such a way that nodes can hear a recessive bit (a logical '1') only if no other node sends a dominant bit (a logical '0'); the bus behaves as a logical wired-AND. The nodes which hear a dominant bit while themselves send a recessive bit, refrain from arbitration. At last the only one node that reaches the end of arbitration without hearing a dominant bit (unless he was sending it as well), proceeds with transmitting the data.

The arbitration phase of dominance/binary countdown protocols is illustrated by an example in Figure 1.

The wireless implementation of a dominance MAC is dubbed WiDom [10]. During the conflict resolution phase, which is called tournament in WiDom, a node with a

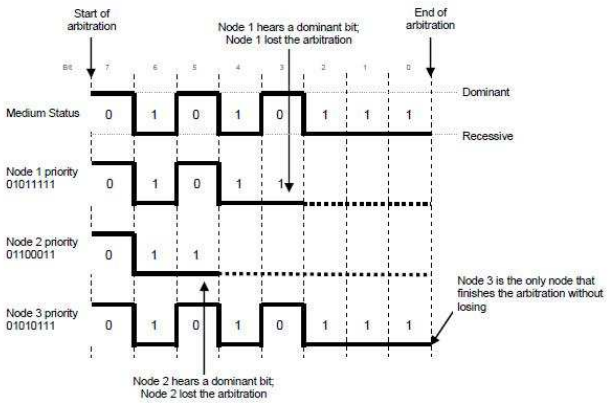


Figure 1. Arbitration in dominance/binary countdown protocols

recessive bit should listen to the medium to assess whether any dominance bit is being transmitted or not. But, wireless transceivers can hardly be transmitting and receiving at the same time. Thus, when the transmitted bit is dominant there is no need to sense the medium, whereas, when the bit to transmit is recessive, nothing has to be effectively sent, and only the medium state has to be sensed. Likewise in CAN, before any tournament, nodes have to agree on a common reference point in time to start transmitting their bits at the same time. This is called synchronization and is achieved by letting nodes to wait for a long period of silence. After detecting this period of silence, a node may signal to start the tournament by sending a synchronization carrier pulse. More details can be found in [10].

WiDom is a collision free and fully distributed protocol. It does not require synchronized clocks and supports a large number of priority levels. Such a large number of priorities can be supported by other prioritized protocols only at the cost of much higher overhead. WiDom can also be used for scheduling sporadic message streams in wireless networks with real-time requirements and provides pre-runtime guarantees. This is important because most of the emerging embedded systems are dealing with physical environments in which, stimuli are typically sporadic.

As it will be explained in the next sections various interesting features of Dominance-based protocols (CAN and WiDom are examples) can be exploited to obtain aggregate quantities in large scale dense networks, with a time-complexity that is very low and independent of the number of nodes.

3. Data Aggregation in Dense Networks

As a result of improving technology the cost of sensor nodes is decreasing towards zero. This makes it economically feasible to deploy and use a large number of

sensor nodes for monitoring the physical quantities. Also very dense networks offer a better resolution of the physical world and a better capability of detecting the occurrence of events.

There are various applications where measurements at fine spatial scales are required. Structural Health Monitoring (SHM) of buildings and propulsion systems, active flow control on the aircraft skin surfaces to reduce fuel consumption by using a very dense deployment of sensor/controller/actuator nodes embedded in the aircraft wings and fuselage [5] are some of the examples.

All these applications stress the use of dense (and large-scale) deployments of sensors/actuators to instrument physical infrastructures. Such density and scale poses huge challenges concerning both interconnectivity and the enormous quantities of sensor data to be processed.

In fact, in a very dense deployment many nodes are typically placed within a same single broadcast domain (SBD). The problem is that in most of the data gathering protocols, the time-complexity depends heavily on the number of nodes. In particular, the performance of those approaches is limited by the fact that nodes in the same broadcast domain cannot transmit in parallel. This results in a very high required time for collecting the information of all nodes and obtaining the required set of measurements. As it is known, feedback control requires that the inputs are measured periodically, and low duty-cycles may imply poor control.

As already mentioned, the recent proposal to use a dominant-based MAC for quantity aggregation opened the possibility of devising faster and more efficient methods for gathering data from sensor readings. In this family of novel distributed algorithms, communications and computations are tightly coupled with the physical environment (an important feature of Cyber-Physical Systems - CPS). Notably, the aggregate quantities can be computed with a time-complexity that is independent of the number of sensor nodes [6, 7]. This is important for dense networks with many sensor nodes.

3.1. Basic Aggregate Quantities

In (wireless) sensor networks each sensor should send its reading to the sink periodically, after detecting an event or after receiving queries from the sink. In contention free medium access protocols such as Time Division Multiple Access (TDMA), the required time for accessing the channel and sending the messages will depend on the network topology and also on the number of nodes. The Dominance-based (or simply DOM) MAC protocol is a non-destructive (because there is in fact a collision) contention MAC protocol.

However, in WiDom or CAN, nodes still have to participate in the arbitration before accessing the medium. Therefore, this approach brings no timing advantage as compared to other naive solutions. On the other hand if nodes use the value of their sensor reading instead of an arbitrary priority, the node winning the contention for medium will be the one with the minimum (MIN) of the sensed values [6, 7]. By this approach, it is possible to aggregate some specific basic quantities from a single broadcast domain in a very short time as compared to any other protocol. Importantly it can be done and in a way that is not dependent on the number of nodes in that broadcast domain. The minimum value (MIN) and the maximum value (MAX) can be obtained with this method with a time-complexity of $O(npriobits)$, where *npriobits* is the number of bits used to represent the data. It is also shown in [10] that more complex aggregated quantities such as MEDIAN, COUNT (an estimation of the number of nodes), and Interpolation can also be obtained elaborating on the basic principle of obtaining MIN.

3.2. Approximate Interpolation

Interpolating the distribution of physical quantities of the physical environment is another interesting possibility of this DOM-based approach. The accuracy of the interpolation and its time-complexity are dependent on a user defined parameter, *k*, which determines the number of nodes used for estimating the approximate interpolation the value of the physical quantity.

The idea of obtaining MIN out of the readings of many sensors within a single broadcast domain with “one shot” ignites the use of this method for other more sophisticated quantities with space information as well; such as the case of approximate interpolation of sensors’ data over a geographical area.

Estimating the distribution of monitored parameters by doing interpolation on sensors’ data needs obviously much less time than receiving data from all individual sensors. Obtaining interpolation of data using DOM-based MAC protocols was proposed for the first time in [7], and some additional improvements / developments were performed later [11, 12].

The basic principle is described in [7]. The algorithm works as follows. To have an interpolation of sensor readings, it is needed that nodes know their location. Assume this is given by Cartesian coordinates (x_i, y_i) for node N_i. Let $f(x,y)$ be the function which interpolates the sensor data, s_i be the sensor reading and e_i be the magnitude of interpolation error at node N_i. Therefore:

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

and the global error would be:

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

where n is the number of nodes. For calculating the interpolation using the DOM-based MAC, nodes send their calculated e_i in the arbitration phase. The node with maximum value for e_i wins the arbitration and continues the transmission by sending its coordinates and measured value, s_i. This node is added to S and the interpolated signal is updated in all participating nodes based on the points in S.

To have a more accurate interpolation, $f(x,y)$ should minimize e. To track physical quantities that change quickly, the computational time of f in each point and also the requiring time for obtaining f from the various sensor readings should be low. Moreover, the interpolation should be updated periodically.

In previous works which used the DOM-based MAC for interpolation (e.g., [7]), weighted-average interpolation (WAI) [13, 14] is used. This function for S, a set of nodes used for interpolation, is defined as follows:

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

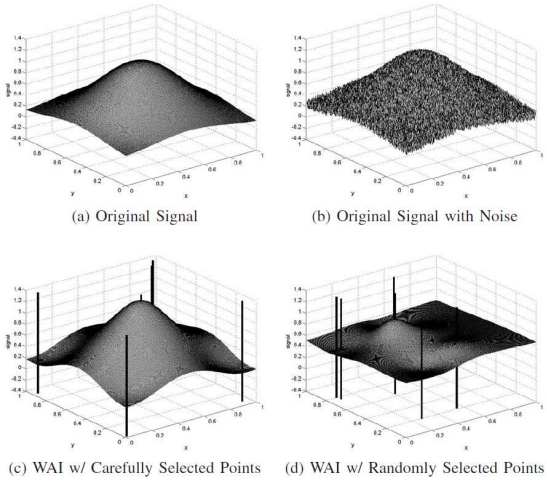
where weights $w_i(x, y)$ are given by

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

This method provides an adjustable accuracy for the user based on the smoothness of sensed parameters, changeability of the environment due to time and the tolerable delay and error for each application.

The pseudo code for the approach is presented in Algorithm 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. 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 illustrates the operation of interpolation 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 6 randomly selected nodes gives poor interpolation. This is illustrated in Figure 2d.



(a) Original Signal (b) Original Signal with Noise

(c) WAI w/ Carefully Selected Points (d) WAI w/ Randomly Selected Points

Figure 2. Interpolation example [7]

Algorithm 1 Basic (Normal) Interpolation algorithm [7]

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.

```
1: function find nodes() return a set of packets
2:   S ← ∅
3:   for q ← 1 to k do
4:     Calculate f(x_i,y_i) in Equation 3 and assign it to the variable "myinterpolatedvalue"
5:     error ← abs( s_i - to integer(myinterpolatedvalue) )
6:     temp_prio ← error × (MAXNNODES + 1) + i
7:     prio ← (MAXP+1) - temp prio
8:     snd pack ←< s_i,x_i,y_i>
9:     <winning_prio, rcv pack> ← send_and_rcv( prio, snd_pack)
10:    S ← S ∪ { rcv pack }
11:  end for
12: return S
13: end function
```

4. Quantity aggregation in Control Loops

One important feature of a NCS is that it efficiently tights communications and computations with the physical world. By featuring efficient data sharing among the various controllers, NCS are able to easily fuse global information to make intelligent decisions over large physical spaces.

However, the insertion of the communication network in the feedback control loop makes the analysis and the design of a NCS a complex issue. Shared networking imposes additional time delays in control loops and increased possibility of data loss. Depending on the application, time-delays can impose severe degradation on the system performance.

To reduce the required time for gathering information from nodes in dense networked control systems, we propose to use quantity aggregation methods and approximate interpolation algorithms.

Interpolating physical parameters by receiving

information from only few sensing nodes produces an overall image of the network in a fast and efficient way.

In the following sub-sections we will discuss the effect of various design / operational options on the quality of control. In particular, we will assess the suitability of the currently available approaches to be used in control loops where the input signal changes very fast. Additionally, in Section 4.3 we will introduce a novel algorithm, which we show is able to perform better by the changes over time of the input signal within the process of obtaining an approximate interpolation.

4.1. Evaluating the Basic Interpolation Algorithm

The basic interpolation algorithm, which was described in Section 3.2, does not take into account the changes over time of the input signal for obtaining an interpolation. The quality of this method is evaluated in this section in terms of computation delay and interpolation error. The goal is to find out the effect of changing parameters such as WAI function and *k* on the accuracy of the interpolation. *k* is the number of points which is used in each round for constructing the interpolated image.

Average Interpolation Error (*AIE*) and Maximum Interpolation Error (*MIE*) are defined as follow:

$$AIE = \frac{\sum_i^n VS_i - \sum_i^n IS_i}{n} \quad (5)$$

$$MIE = \max_{i=1 \,..\, n} (VS_i - IS_i) \quad (6)$$

where VS_i is the measured value of sensor *i*, IS_i is the calculated value in the geographical position of sensor node *i* by the interpolation method and *n* is the number of sensor nodes.

To compare the computation time of the various algorithms, we use a timing function available from the C language, since we are interested in the relative, not the absolute value, of the computation time. Those absolute values will obviously depend on the actual real sensor platforms in which the algorithms may run.

We studied the effect of changing the WAI function on reducing the interpolation error. Changing the power of the denominator in the weighting function (Eq. (4)) shows that power value of '1.1' results in the lowest error which is slightly better (0.04%) than the result of power value '1' (Figure 4). Therefore, power '1' (as in Eq. 4) is used in simulation results due to simpler computation.

For a sample signal as illustrated in Figure 3, the simulation results for Algorithm 1 are presented in Figures 5 to 7. The amplitude of Signal in Figure 3 has values between zero and one across the domain. The results of the simulation show that, by increasing *k*, the computation time of interpolation is linearly increased. This is more or less a obvious result since the computation time of each round is proportional to the number of interpolation points.

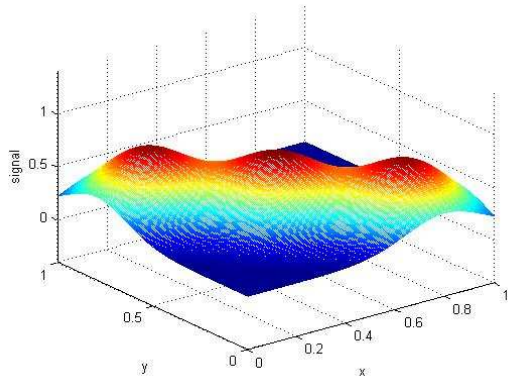


Figure 3. An example signal

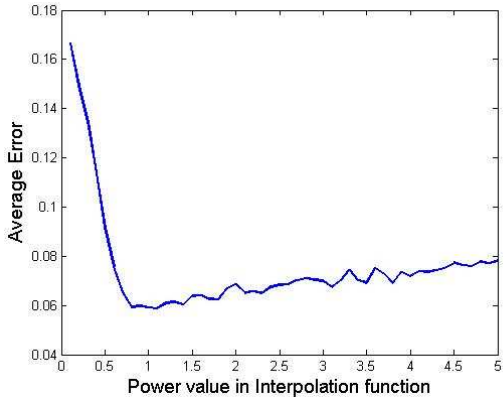


Figure 4. Average error for different WAI function

However, the maximum and average interpolation errors are not reduced considerably for large values of *k*. The number of nodes with interpolation error more than a threshold (18 percent in Figure 7) keeps decreasing. But, after a specific value of *k* (for example 25) this number remains approximately constant (Figure 7). Therefore, increasing *k* would not necessarily improve the accuracy of interpolation, while the time complexity would increase unnecessarily.

One improvement to the basic interpolation algorithm was proposed in [11]. In that variation, for interpolating the value of each point, only the closest control points to that point are considered. The simulation results show that based on the shape of the signal, that version of the algorithm may or may not decrease the interpolation error. However, the computation delay is increased due to search for closest control points to each point.

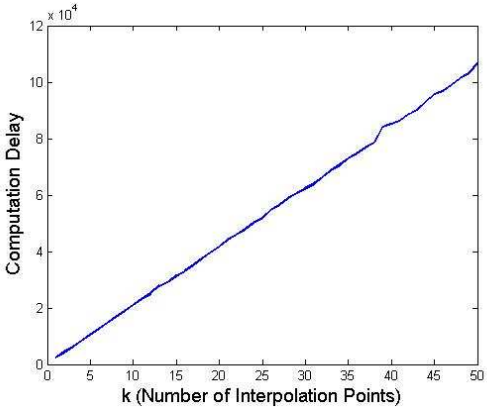


Figure 5. Computation Delay for a static signal

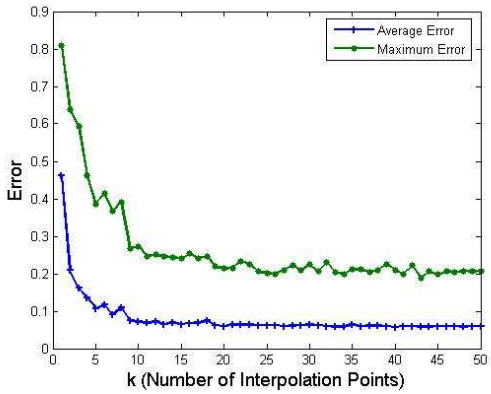


Figure 6. Interpolation Error for a static signal

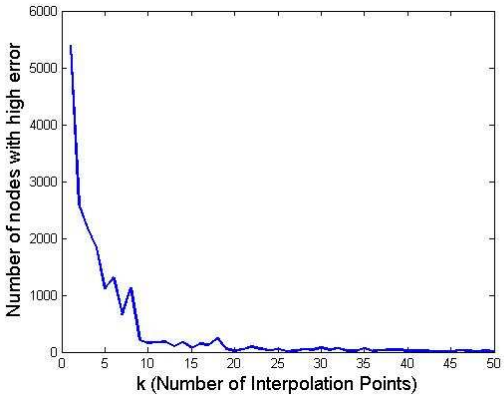


Figure 7. Number of Nodes with High Error

4.2. Evaluating the Incremental Interpolation

In [7, 11] it is assumed that sensor readings do not change much during an interpolation round. For these static signals even by considering noise, the average interpolation error would be less than 10 percent for $k > 10$ and it keeps decreasing slightly by increasing k.

But, the reality is that signals change over time. Even with two percent change in signal per interpolation iteration (each k) and without considering noise, the average error is more than 10 percent and keeps rising by incrementing k (Figure 8). Higher computation delay for bigger ks causes the interpolation algorithm to not follow the changes in physical quantities appropriately.

The explanation for this behavior is intuitive as well. When a point is added into the set of interpolation points, S, the one that was added previously may be already measuring a very different value. Accordingly, the basic Interpolation algorithm cannot track the changes in signal appropriately. For the interpolation method to be applicable for control applications, changes in physical quantities should be taken into account. In other words, the interpolation algorithm should be able to interpolate signals that change with respect to time during one interpolation round. This need is more acute in very dynamic physical quantities.

A modified algorithm which was proposed in [12] provides an Incremental interpolation. This algorithm was designed especially for dynamic signals. The aim of this algorithm was to react fast to changes in the physical quantity being tracked. The normal interpolation algorithm obtains an interpolation from scratch every time it is executed. Conversely, the Incremental algorithm uses the information of the previous rounds to improve the interpolated signal step by step (incrementally). After the completion of the startup phase, old control points are replaced by new ones and the interpolation is updated iteratively. Removing and adding control points is done with the rational that the least recent nodes in S contribute the least to a faithful representation of the physical world. The pseudo-code of this algorithm is presented in Algorithm 2.

The algorithm works as follows. First, Algorithm 1 is called and this gives us a set S with the elected 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 incremental algorithm was not implemented before and its performance was not evaluated. The simulation results show that removing old nodes from S has a worse effect on the interpolation results since those nodes have the most contribution in constructing the interpolation. Removing them can thoroughly distort the interpolated signal. Not removing the old nodes from S causes more computation complexity and not necessarily better accuracy, similarly to the effect in the basic algorithm

Algorithm 2 Incremental Interpolation [12]

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.
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 ← S \ 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: error ← abs(s_i - to integer(myinterpolatedvalue))
13: temp_prio ← error * (MAXNNODES + 1) + i
14: prio ← (MAXP+1) – temp_prio
15: snd_pack ←< s_i,x_i,y_i>
16: <winning_prio, rcv_pack> ← send_and_rcv(prio, snd_pack)
17: S ← S ∪ { rcv_pack }
18: **end while**

when increasing unnecessarily the value of k.

4.3. A New Interpolation Algorithm

For better coping with fast changing physical signals, a new interpolation algorithm is proposed in this paper. This algorithm uses some information from the system about the type of changes in the physical quantities. For the sake of simplicity (other variants can be further elaborated), we consider that the changes in a signal are monotonous, and therefore by calculating the Differential once at the considered points and updating the value at the referred point at each iteration k of the interpolation, the approximate interpolation results much better.

For a monotonous increment / decrement change, Algorithm 3 describes the proposed approach. All the nodes execute the same algorithm in which they are aware that after receiving each new control point, the previously taken one will resend its new sensed value. In the other words, in each iteration (except for the first one), after receiving the information of new Control point, the previous considered node sends its value again (line 15). Then, it is possible for all the nodes to measure the approximate Differential of changes in that control point (line 16). This information will be applied in the next iterations for obtaining the interpolation as follows:

$$f(x,y) = \begin{cases} 0 \; if \; S = \emptyset \\ s_i \; if \; \exists \; N_i \in S : \; x_i = x \wedge y_i = y \\ \frac{\sum_{i \in S} (s_i + g_i).w_i(x,y)}{\sum_{i \in S} w_i(x,y)} \; otherwise \end{cases} \quad (7)$$

where g_i is the Differential of i^{th} interpolation point and the other parameters are as described previously for Eq. 3.

Simulation results show a great improvement in interpolation of the signal by using Algorithm 3 (Differential algorithm) instead of Algorithm 1 (Basic Algorithm). The results are presented in two categories. Figure 8 and 9 show average error of both algorithms when the rate of change in signal is limited (up to 4%) in each interpolation round. With random changes in the signal, the basic algorithm is unable to follow the signal by elapsing time. Increasing the interpolation points means increasing the time of interpolation. However, the Differential algorithm has less than 10% error in interpolating the signal (Figure 8). When the rate of changes is constant, the Differential algorithm has slightly better result while the basic algorithm gets worst result (Figure 9).

The second category is presenting the change in error percentage for different rates of change in signal for a constant number of interpolation rounds (Figure 10, 11). Increasing the rate of change leads to small rise in the percentage of error for Differential algorithm whereas for the basic algorithm average error keeps increasing. For the random scenarios, the presented results are the average of 100 runs of the algorithms.

Algorithm 3 Improved Interpolation Algorithm

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.

1: **function** find nodes() **return** a set of packets
2: S ← ∅
3: **for** q ← 1 to k **do**
4: g_q ← 0
5: **end for**
6: **for** q ← 1 to k **do**
7: Calculate f(x_i,y_i) in Equation 7 and assign it to the variable "myinterpolatedvalue"
8: error ← abs(s_i - to integer(myinterpolatedvalue))
9: temp_prio ← error × (MAXNNODES + 1) + i
10: prio ← (MAXP+1) – temp_prio
11: snd_pack ←< s_i,x_i,y_i>
12: <winning_prio, rcv_pack> ← send_and_rcv(prio, snd_pack)
13: S ← S ∪ { rcv_pack }
14: **if** q ≠ 1 **then**
15: the new sensed data of $(q-1)^{th}$ control point is received.
16: g_q ← the change in value of the control point
17: **end if**
18: **end for**
19: **return** S
20: **end function**

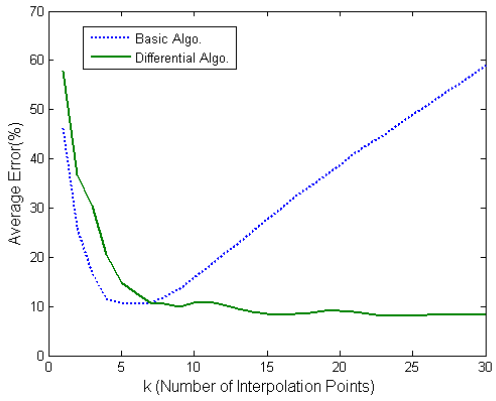


Figure 8. Average Error of Basic and Differential algorithms with random (up to 4%) change in signal per interpolation round

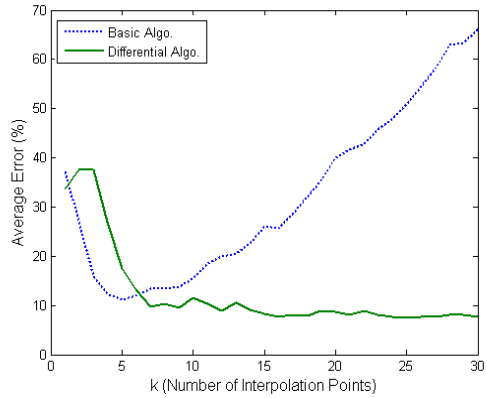


Figure 9. Average Error of Basic and Differential algorithms with constant (4%) change in signal per interpolation round

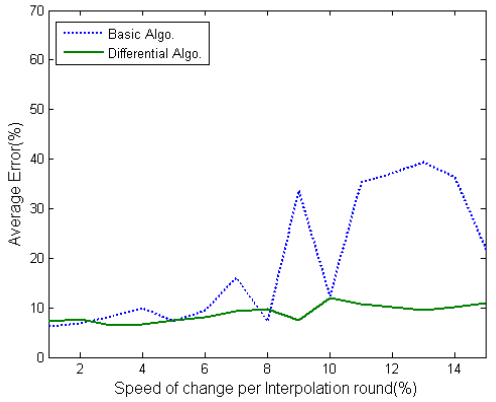


Figure 10. Average Error of Basic and Differential algorithms versus Random rate of change in signal for k= 12

Note however that each round of Interpolation in Algorithm 3 is longer than the round of interpolation in Algorithm 1 since there is a re-sending of data during each iteration. If the arbitration takes x time units and sending data takes y time units, the communication time of each

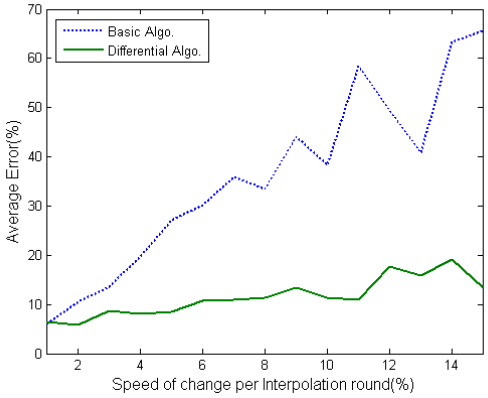


Figure 11. Average Error of Basic and Differential algorithms versus constant rate of change in signal for k= 12

iteration in Algorithm 3 last (x + 2 * y) time units compared to (x + y) in Algorithm 1.

The computation time is also longer in Algorithm 3 due to recalculating the interpolated values at each iteration k. In the "normal" approach for the interpolation, by adding new control points, some terms were added to nominator and denominator of interpolated values. Conversely, in Algorithm 3 in each iteration the Eq. 7 should be recalculated from scratch since the values in set S of previous iteration may have changed.

5. Implementation Issues

We performed a brief analysis of the time to compute Basic Algorithm and Differential Algorithm in real-world sensor network platforms. This was done by implementing Differential Algorithm for the MicaZ platform. The code was implemented making use of basic hardware abstraction code such that the code running on the platform was reduced to a minimum and we had total control over the code being executed. The code was compiled with no compiler optimizations and the execution time was measured using a microcontroller real-time clock. The results are presented in Figure 12. For Differential Algorithm, we considered that each node had to compute the differential and also computed the interpolated value at all iterations, which is the worst-case computation scenario. As we can see, the execution time of Differential Algorithm increases much faster. This is because it needs to recompute Equation 7 at each iteration. These are important results to bear in mind when developing new interpolation schemes and models of the physical phenomena. However, they are seen as preliminary results, since we believe the implementation of the algorithm can be improved. One possible approach is to change the implementation such that we avoid computing all terms of Equation 7 by maintaining the partial sums in the numerator and denominator of

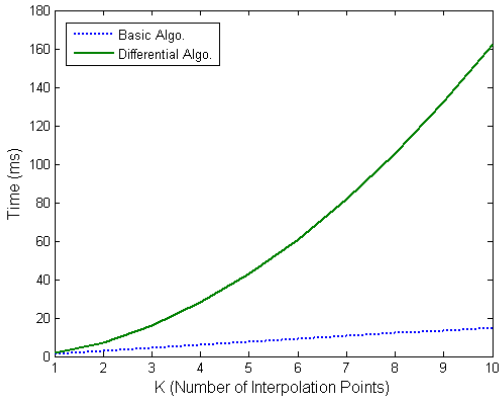


Figure 12. Execution time as a function of *k* for Basic and Differential algorithms in a real-world platform.

Equation 7. This is possible assuming the Differential is constant after inserted in *S*.

Other improvements such as addressing specific limitations of the platform can be considered and we will be working on developing better implementations.

6. Conclusion and Future works

In this paper we discuss a few aspects of using the basic building block of MIN (MAX) in DOM-based quantity aggregation in feedback control in networked control systems. Based on information of few points, an interpolated signal is constructed which enables estimating the information (about some physical quantity) of all points in a distributed geographical area. This approximate interpolation can be used as the input data in feedback control systems. Previous interpolation algorithms could not perform well in terms of both accuracy and latency. In this paper we propose a novel algorithm that considers simple predictors on how the physical signal changes during computation of the input (an approximate interpolation function) of the physical signal.

This is the beginning of a set of research efforts which attempt to minimize the error of the DOM-based interpolation approach while maintaining low time-complexity. We are starting to consider more sophisticated approaches, such as using Kalman filters, to further improve the overall performance of the algorithm. Next step will be implementing Algorithm 3 and having a set of benchmarking tests able to prove the usefulness of such type of improvements.

References

[1] J.P. Hespanha, P. Naghshtabrizi, Y. Xu, "A Survey of Recent Results in Networked Control Systems," Proceedings of the IEEE, Vol. 95, No. 1, January 2007.

[2] G. C. Goodwin, D. E. Quevedo, and E. I. Silva, "An introduction to networked control systems," in Proc. Asian Control Conference, Bali, Indonesia, 2006.

[3] X. Liu and A. Goldsmith, "Wireless Medium Access Control in Networked Control Systems," Proc. IEEE American Control Conference, June 2004.

[4] L. Becchetti, P. Korteweg, A. Marchetti-Spaccamela, M. Skutella, L. Stougie, and A. Vitaletti. Latency Constrained Aggregation in Sensor Networks. In Proc. European Symposium on Algorithms (ESA), pages 88–99, 2006.

[5] "Fly-by-Wireless: A Revolution in Aerospace. Architectures for instrumentation and Control," NASA/CANEUS Workshop. NASA/JSC/ES6/George Studor.

[6] N. Pereira, R. Gomes, B. Andersson, and E. Tovar, "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, 2009.

[7] B. Andersson, N. Pereira, W. Elmenreich, E. Tovar, F. Pacheco, and N. Cruz, "A scalable and efficient approach to obtain measurements in CAN-based control systems," in IEEE Transactions on Industrial Informatics, vol. 4, May, 2008, pp. 80–91.

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

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

[10] N. Pereira, B. Andersson, and E. Tovar, "WiDom: A dominance protocol for wireless medium access," IEEE Transactions on Industrial Informatics, vol. 3(2), May 2007.

[11] E. Tovar, B. Andersson, N. Pereira, M. Alves, S.Prabh and F. Pacheco, "Highly Scalable Aggregate Computations in Cyber-Physical Systems: Physical Environment Meets Communication Protocols," Proceedings of the 7th International Workshop on Real-Time Networks (RTN'08), Prague, Czech Republic, July 1, 2008.

[12] B. Andersson, N. Pereira, E. Tovar and R. Gomes, "Using a Prioritized Medium Access Control Protocol for Incrementally Obtaining an Interpolation of Sensor Readings," Intelligent solutions in Embedded Systems, Seventh Workshop on 25-26 June 2009, Page(s): 29-36

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

[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, 2004, pp. 166 – 175.

[15] N. Pereira, R. Gomes, B. Andersson, E. Tovar, "Efficient Aggregate Computations in Large-Scale Dense WSN," Proceedings of the 2009 15th IEEE Symposium on Real-Time and Embedded Technology and Applications, 2009. pp: 317-326.