CISTER
Research Center in
Real-Time & Embedded
Computing Systems

Technical Report

Dynamic Cluster Scheduling for Cluster-tree WSNs

Ricardo Severino
Nuno Pereira
Eduardo Tovar

CISTER-TR-130205
Version:
Date: 2/15/2013

Dynamic Cluster Scheduling for Cluster-tree WSNs

Ricardo Severino, Nuno Pereira, Eduardo Tovar

CISTER Research Unit

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

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

Abstract

While Cluster-Tree network topologies look promising for WSN applications with timeliness and energy-efficiency requirements, we are yet to witness its adoption in commercial and academic solutions. One of the arguments that hinder the use of these topologies concerns the lack of flexibility in adapting to changes in the network, such as in traffic flows. This paper presents a solution to provide these networks with the ability to self-adapt to different bandwidth and latency requirements, imposed by traffic flows, by changing the cluster's duty-cycle and scheduling. Importantly, our approach enables a network to change its cluster scheduling without requiring long inaccessibility times or the re-association of the nodes. Importantly, we show how to apply our methodology to the case of IEEE 802.15.4/ZigBee cluster-tree WSNs without significant changes to the protocol. Finally, we analyze and demonstrate the validity of our methodology through a comprehensive simulation and experimental validation using commercially available technology on a Structural Health Monitoring application scenario.

Dynamic Cluster Scheduling for Cluster-tree based WSNs

Ricardo Severino, Nuno Pereira, Eduardo Tovar
CISTER/INESC-TEC, ISEP
Polytechnic Institute of Porto
Porto, Portugal
{rar,nap,emt}@isep.ipp.pt

***Abstract*—While Cluster-Tree network topologies look promising for WSN applications with timeliness and energy-efficiency requirements, we have witnessed a lack of commercial and academic solutions based on this kind of topology. One of the most important arguments that hinder the use of these topologies concerns the lack of flexibility in adapting to changes in the network, such as in traffic flows. This paper presents a solution to provide these kind of networks with the ability to adapt in real-time to different bandwidth and end-to-end delay requirements imposed by incoming traffic streams, by changing the cluster's duty-cycle and scheduling. Importantly, our approach enables a network to change its cluster scheduling without requiring long inaccessibility times or the re-association of the nodes. We show how to apply our methodology to the case of IEEE 802.15.4/ZigBee cluster-tree WSNs. Finally, we analyze and demonstrate the validity of our methodology through a comprehensive simulation and experimental study using commercially available technology on a real-world Structural Health Monitoring application scenario.**

Keywords-Cluster-Tree; ZigBee; Scheduling;

I. INTRODUCTION

The increasing tendency for the integration of computations with physical processes at large scale has been pushing research on new paradigms for networked embedded systems design [1]. In this line, Wireless Sensor Networks (WSNs) have naturally emerged as enabling infrastructures for these cyber-physical applications due to their potential to closely interact with external stimulus. Applications such as homeland security, physical infrastructures monitoring, health care, building or factory automation are just a few elucidative examples of how these emerging technologies will impact our daily life and society at large.

Given the large number of these WSN applications each with an individual set of requirements [2], it is important that some of these WSN resources (e.g. bandwidth and buffer size), are predicted in advance, in order to support the prospective applications with a pre-defined Quality-of-Service (QoS).

For instance, an environmental monitoring application that simply gathers temperature readings has less stringent requirements than a real-time tracking or a structural health monitoring application.

To achieve this, it is mandatory to rely on structured logical topologies such as cluster-trees (e.g. [3], [4], [5]), which provide deterministic behavior in terms of routing and the ability to perform end-to-end resource reservation, against flat mesh-like topologies, where probabilistic routing protocols are commonly used.

Although these network topologies look promising for these WSN applications we have witnessed a lack of commercial and academic solutions based on this kind of topology. Perhaps one of the most important arguments that hinder the use of these topologies, so adequate given the degree of predictability that can be achieved, is their lack of flexibility in adapting to changes in the traffic or bandwidth requirements in real-time, for instance, how to allocate more bandwidth to a set of nodes sensing a particular phenomena, or how to reduce the latency of a data stream, without having to restart the network again. In fact, although there is already some literature on how to compute these network resources [6], [7], it is not clear how they could be re-allocated without greatly interfering with the network functionality, and specially without imposing high inaccessibility times.

This paper presents a solution to this problem, enabling networks to adapt in real-time to different bandwidth and end-to-end delay requirements imposed by incoming traffic streams, by changing the cluster's scheduling. Computing this would normally result in a complex linear programming problem which would be unfeasible to be computed by WSN nodes which typically have scarce computing power. Our re-scheduling algorithm relies in a heuristic that can be easily computed in these platforms.

In this paper, we show how to apply our methodology to the particular case of IEEE 802.15.4/ZigBee cluster-tree WSNs. Finally, we analyze and demonstrate the validity of our methodology through a comprehensive simulation and experimental study using a real-world Structural Health Monitoring application scenario, showing that our proposal can reduce the end-to-end latency in 93% and the overall data transmit duration in 49 %.

II. RELATED WORK

Synchronized Cluster-Tree topologies tend to suffer from two technical issues that tend to inhibit their use: (1) how to schedule the transmissions of different neighboring clusters, (2) how to predict the performance limits to correctly

allocate resources, and (3) how to change the resource allocation of the CT on the fly.

This is specially true for the particular case of the IEEE 802.15.4/ZigBee set of protocols, in which although the Cluster-Tree network topology is supported, no clear description on how to implement it is given, namely in what concerns the beacon collision problem.

In [8], the Task Group 15.4b proposed some basic approaches to solve this: the beacon-only period approach and the time division approach, only to be removed in the 2006 revision. In this line, in [9], the TDCS algorithm was proposed and implemented in the Open-ZB stack [10] enabling for the first time the use of this topology over IEEE 802.15.4/ZigBee based networks guaranteeing no beacon collisions. This technique used the time-division approach and worked by assigning a different time offset to each cluster. Other proposals followed a similar approach like [11] for mesh networks, or [12]. In [13] the authors choose to use different radio channels instead to tackle the problem, assuming they are available.

Recently, research was done [6], to addresses the problem of predicting resource needs by modeling the performance limits of a ZigBee CT network. In another work, [7] extended the previous work by computing the optimal TDCS schedule for several GTS data flows. Here, the assignment of the resulting parent-child offsets followed the same strategy as in [9], where the offset assignment is done at network setup time and remains fixed.

Although, some literature is already available on solving the first two problems, none of the proposals so far, in the general case of synchronized Cluster-Trees, addresses the third. This greatly limits the flexibility of the network which must keep the same cluster schedule and bandwidth reservation, independently of the flow of data in the network and of its particular requirements, which depending on the application may certainly change.

In this paper, we address this issue, and propose a set of techniques, in which the base schedule is temporary changed to encompass transient networking necessities such as end-to-end delay and bandwidth allocation.

III. Dynamic Cluster Scheduling

A. System Model

1) Cluster-Tree topology

Consider a synchronized cluster-tree WSNs featuring a tree-based logical topology where nodes are organized in different groups, called clusters. Each node is connected to a maximum of one node at the lower depth, called parent node, and can be connected to multiple nodes at the upper depth, called child nodes (by convention, trees grow down). Each node only interacts with its pre-defined parent and child nodes.

A cluster-tree topology contains two main types of nodes: routers and end-nodes (refer to Figure 1). The nodes that can associate to previously associated nodes and can participate in multi-hop routing are referred to as routers. The leaf nodes that do not allow association of other nodes and do not participate in routing are referred to as end-nodes.

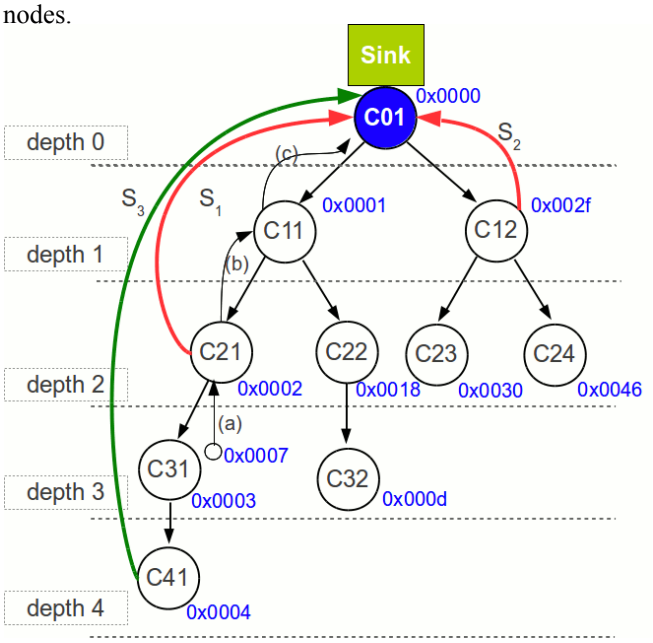

Figure 1

The router that has no parent is called root and it might hold special functions such as identification, formation and control of the entire network. Note that the root is at depth zero. Both routers and end-nodes can have sensing capabilities, therefore they are generally referred to as sensor nodes. Each router forms its cluster and is referred to as cluster-head of this cluster (e.g. router C11 is the cluster-head of cluster 11). Each cluster-head is also responsible for synchronization in its cluster and periodically sends synchronization frames.

All child nodes (i.e. end-nodes and routers) of a cluster-head are associated to its cluster, and the cluster-head handles all their data transmissions. It results that each router (except the root) belongs to two clusters, once as a child and once as a parent (i.e. a cluster-head).

2) Time-Division Cluster Scheduling

In general, the radio channel is a shared communication medium where more than one node can transmit at the same time. In cluster-tree WSNs, messages are forwarded from cluster to cluster until reaching the sink. The time window of each cluster is periodically divided into an active portion (AP), during which the cluster-head enables data transmissions inside its cluster, and a subsequent inactive portion, during which all cluster nodes may enter low-power mode to save energy resources. Note that each router must be awake during its active portion and the active portion of its parent router. To avoid collisions between clusters, it is mandatory to schedule the clusters' active portions in an ordered sequence, that we call Time Division Cluster

Schedule (TDCS). In other words, the TDCS is equivalent to a permutation of active portions of all clusters in a WSN such that no inter-cluster collision occurs. In case of single collision domain, the TDCS must be non-overlapping, i.e. only one cluster can be active at any time. Hence, the duration of the TDCS's cycle is given by the number of clusters and the length of their active portions. On the contrary, in a network with multiple collision domains, the clusters from different non-overlapping collision domains may be active at the same time. However, finding a TDCS that avoids clusters' collisions in a large-scale WSN with multiple collision domains is a quite complex problem, hence in this paper, for simplification, we always assume a single collision domain. For more information concerning TDCS please refer to [9].

B. Dynamic Cluster Scheduling

With TDCS [9] it is possible to find the best schedule for the routers active periods in order to avoid interference, and to support most of the network bandwidth requirements. However, the schedule is done at network setup time and assumes a static network that will remain unchanged.

The choice of the TDCS schedule has a strong impact in the end-to-end delays, in fact, it is easy to observe that in a single collision domain, where there are no overlapping clusters, a TDCS schedule optimized for downstream communication, will result in the worst-case for upstream communication, and consequently in higher end-to-end delays. Moreover, the routers are assigned with a fixed bandwidth they might not always need, while other clusters might need that bandwidth. We aim at reacting to different data flow changes in real-time, while simultaneously minimizing the network inaccessibility time. Our proposal achieves this via two techniques: (1) re-ordering the clusters' active periods to favor one set of streams, reducing the end-to-end delays, which we call DCR (DCS Cluster Re-ordering); and (2) tuning the size of the clusters' duration, increasing the bandwidth of the clusters serving a specific stream, while decreasing others bandwidth, which we named DBR (DCS Bandwidth Re-allocation).

The first technique, consists of a rescheduling of the clusters order in the TDCS cycle, aiming at minimizing end-to-end delays, while the second technique consists of rearranging the bandwidth allocation for the clusters involved in a stream, to increase its bandwidths and decrease the overall time for data transmission. Both techniques can be used together, or separately.

1) DCR Cluster Reordering Technique

Consider the cluster-tree presented in Figure 1, with 10 clusters and a TDCS schedule as presented in Figure 2 Schedule A, where all Chs have the same allocated bandwidth.

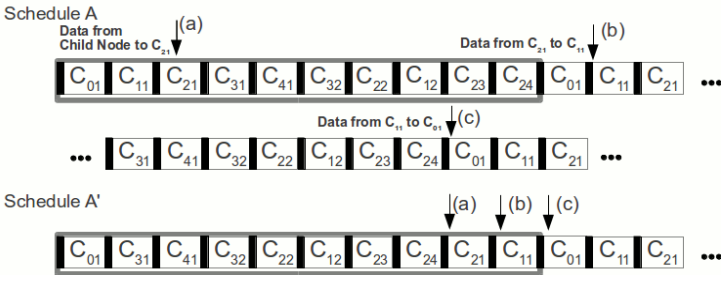

Figure 2

Notice that this schedule is set to minimize downstream traffic latency, (parents' appear earlier in the schedule than the child nodes) which is common in applications which require fast actuation actions. This way, to actuate on Cluster C_{21} for instance, one could do it in only one TDCS cycle, since those Cluster's are active immediately one after each other. If a reasonable amount of data resulting from these actuation actions is to be sent to the Root, where the Sink is located, a large delay is to be expected in receiving it, since the selected TDCS schedule is not the best for transmitting data upstream. In fact, assuming that all data could be transmitted from one CH to the next in one cycle, it would take two cycles. One from C_{21} to C_{11}, and another from C_{11} to C_{01}. This is depicted in Figure 2. (a) represents the data coming from the sensing node and being received by C_{21}. Next, (b) represents the transmission from C_{21} to C_{11}, and finally, (c) from C_{11} to C_{01}. The stream is also represented in Figure 1.

This delay will become higher as the network size and the clusters' duration increases and as the Depth of the tree of the source increases. In this scenario, the best schedule to minimize upstream latency, considering a stream from Cluster C_{21} to the Sink (S_1 in Figure 1), should be as depicted in Figure 2 Schedule A', where the next cluster to receive the packet appears next, reducing the amount of time a packet needs to be left in the queue and consequently the application end-to-end delays.

Ideally, the schedule should carry out an on-line re-scheduling of the network to favor a known set of upstream streams, minimizing the inaccessibility times.

This kind of rescheduling involves a re-ordering of the Clusters according to the streams the network must serve. This can easily grow into a complex problem if one wishes to achieve an optimum solution, due to the precedences in the tree, usually solved in the literature using linear programming [7]. However, in order to react to the network specific needs in a reasonable amount of time, one needs to guarantee that the algorithm to compute this new schedule is light and fast enough to be run in WSN platforms with scarce processing power. In this line, linear programming models might just not be the best choice for this, where we just need a better and not necessarily the optimum solution. Our approach to the problem is explained as follows.

Each stream is noted as a tuple $S_k = <R_k , P_k , T_k , D_k>$, where, R_k represents the ordered set of clusters which the

stream k must cross to reach the sink, P_k represents the priority for that stream which is an integer from 0 to 5, T_k represents the number of TDCS cycles for which stream k will remain active and D_k the depth of the stream's source.

Given the set of streams S, and the set N which contains all the cluster-heads in the tree, we must compute A which denotes the set of cluster-heads that need rescheduling, where $A = N \cap S$. Then, C_r, which denotes the priority of the r^{th} cluster-head in N can be computed through the following algorithm.

```
for all cluster head r in A do
  for all stream k in S do
    if A_r ∈ R_k
      C_r ← C_r + P_k
  end for
  C_r ← C_r + h(A_r)
end for
```

In other words, being M_r the set of streams from m to Nm which contain CH r in R, and P_m the stream's priority, we can compute C_r as:

$$C_r = \sum_{m=0}^{Nm} P_m + h(A_r)$$

Function $h(A_r)$ computes the height of Cluster-Head A_r in the tree, according to the position each Cluster holds in array R_k being the first CH in the array position 0. Thus, $h(A_r) = D_k - pos(R_k)$. This value will sum with the already computed cluster's priority to enable precedence in the schedule.

The resulting schedule will be achieved by ordering the set of all cluster-heads N according to the computed C_r for each cluster-head A_r, starting from the lower priorities to the highest. As a result, the highest priority will always be assigned to the sink, since all the streams are directed to that cluster-head. Cluster-heads which are not part of the set A keep their schedule not to interfere with the initial schedule of those and are place after the sink.

As an example, consider the network presented in Figure 1 and assume the following set of streams:

$S_1 = < \{C_{21}, C_{11}, C_{01}\}, 3, 3, 2 >$

$S_2 = < \{C_{12}, C_{01}\}, 1, 4, 1 >$

The first stream, originates at cluster C_{21} and has priority 3, while the second, originates at cluster C_{12} and has priority 1. If no reschedule was done, and assuming ideal communication without errors and delays imposed by the MAC layers, we would expect that one packet of S_1 would take approximately 18 times the duration of one active portion of a CH to reach the sink (Figure 2 Schedule A), and from S_2 three active portions. Lets now, run the presented algorithm and analyze the result.

$A = \{ C01, C11, C12, C24 \}$

$C_{01} = P_1 + P_2 + h(A_{01}) = 3 + 1 + 2 = 6$

$C_{11} = P_2 + h(A_{11}) = 3 + 1 = 4$

$C_{12} = P_1 + h(A_{12}) = 1 + 1 = 2$

$C_{21} = P_1 + h(A_{21}) = 3 + 0 = 3$

Ordering from the lowest to the highest priority, the CHs in A should be ordered as C_{11}, C_{24}, C_{12} and finally C_{01}. Considering the remaining nodes, which maintain their initial order in the schedule and lowest priority, the final schedule would be as follows:

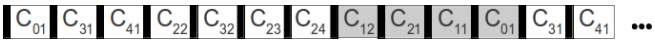

Figure 3

It is now possible a full data transaction from the origin cluster to the sink in one TDCS cycle. This reduces the transmission delay of each packet, greatly benefiting applications which demand low latencies. A packet from S_1 should now take approximately three CH active portions to reach the Sink while S_2 maintains the same three. If we wanted to decrease the latency for S_2 we could increase the priority of the stream to the same of S_1 or higher. This would result in $C_{12} = P_1 + h(A_{12}) = 3 + 1 = 4$, and now, C_{12} would have a higher priority than C_{21} thus appearing later in the schedule and decreasing the latency.

Now, comparing this schedule with the original in Figure 2 Schedule A, we observe that all the other CH also changed place in the schedule. Changing the position of all nodes must be done because there is no free room that will let us only change the streaming CHs' position and accommodate their initial positions unoccupied. However, this does not necessarily mean that all of the CHs change the offset to their parents.

For instance, in this particular case C_{41} does not change the offset. This is obvious, since the distance between C_{41} and its parent C_{31} did not change. However, C_{31} offset changed in relation to its parent C_{21} since C_{21} was re-scheduled. As a rule of thumb, a new offset will have to be computed for every children one depth bellow a re-scheduled CH. For their grand-children, this does not happen since the distance remains the same as in the original schedule. This principle will be used later in STEP 4 (Section B. 3).), to compute the network's inaccessibility time.

Finally the remaining part of the technique consists in computing all the new offsets. This is a matter of measuring the distances between parent and child in the new schedule. One way of implementing this is by creating an array of size equal to the number of Clusters in the network, where each position holds a Cluster number and its respective parent (easily set at network setup time), ordered according to the schedule. Then it is a matter of using simple counters on the array.

Although this approach solves the latency problem, it does not reduce the overall time it will take for a stream to be transmitted since there is no change to the available bandwidth per cycle. Hence our second proposal, DBR, which consists in increasing the bandwidth for the clusters involved in the stream.

2) DBR Bandwidth Re-Allocation Technique

For the second technique, bandwidth must be reallocated, by increasing the bandwidth for the clusters involved in the stream. The first step is to look for free space in the schedule that has not been reserved by a cluster's active portion. If there is such free space, we can distribute in an equal fashion the available space by the Clusters involved in the stream. For the particular case of Figure 2 Schedule A there is no space available. This means, we must try to reduce the amount bandwidth the clusters not related to the stream are using. Here, it is important to previously define the minimum bandwidth a Cluster can support. This is implementation dependent in many cases, since it is highly dependent of the limitations of the hardware platforms. If the SO is reduced beyond a threshold, there can be timing issues. This has been reported previously and is discussed in [14] concerning the TelosB and MicaZ platforms. The minimum bandwidth that will be available to the other clusters after the use of this technique is thus set at network setup time.

If we consider stream S_3 (Figure 1) originating a C_{41}, where $R_{41}=\{C_{41}, C_{31}, C_{21}, C_{11}, C_{01}\}$, the one we wish to increase the bandwidth of every cluster, and a system which is capable of handling a reduction of the available bandwidth by half, this technique will cut all the remaining 5 CH's duration, and redistribute this duration by the other CH's in R_{41}. This results in an increased bandwidth for that stream (Figure 4), thus reducing the transmission time. The size of the TDCS cycle is kept nonetheless, since the bandwidth was simply redistributed.

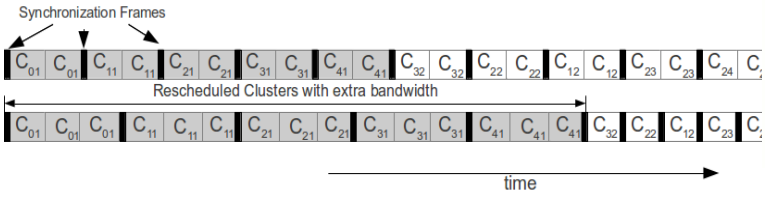

Figure 4

As depicted in Figure 4, all of the relative offsets have changed. Nevertheless, a great plus of this technique is that the network inaccessibility time is minimum if compared to the previous technique, since in only one cycle, it is possible to reschedule all the network with the new offsets, if the original schedule was setup to facilitate downstream communications. This technique is however, greatly dependent of the protocol in use, since, some protocols only allow discrete steps in the duration of the CH's active portion, like the IEEE802.15.4/ZigBee set of protocols. Because of this, a more detailed explanation about this will be given in Section IV concerning IEEE802.15.4/ZigBee protocols.

3) *The DCS Communication Protocol*

Our proposed on-line re-scheduling technique comprise six steps (Figure 5), which can easily be adapted to different network protocols. The protocol is depicted in Figure 5 in a time diagram and is described as follows.

STEP 1 - At network setup time, all Cluster-Heads are assigned with a TDCS time offset in relation to their parents according to the approach proposed in [9].

Different priorities are also assigned to different sensing actions by the nodes. Synchronization frames are sent periodically and several actuation actions on the leaf nodes can be carried out.

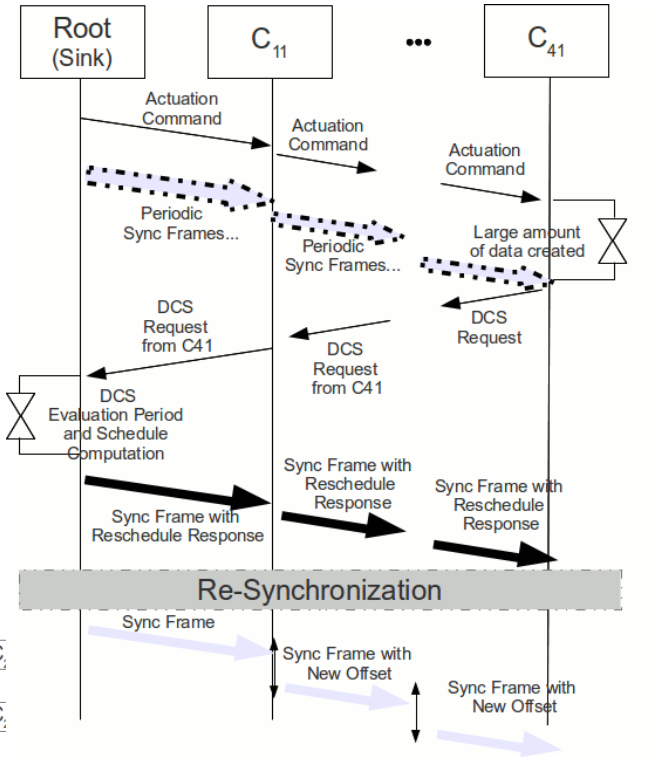

Figure 5

STEP 2 – DCS Request; If a leaf node wishes to transmit a stream of data to the Sink, its Cluster-Head must be informed. The CH will decide, according to the application which originates the request, if the most adequate strategy is a rescheduling to minimize end-to-end delays or an arrangement of the bandwidth, or both. The option of which technique to use must be defined at network setup time, since different applications impose different requirements (reduced latency or transmission time).

This request is then forwarded to the parent until it reaches the Root. On the way, each CH will add its own address to the message, to inform the Root of the clusters involved in the stream. This way, we avoid using heavy lookup tables that would have to be loaded into the Root at network setup time describing all parent child relationships.

The DCS Request is formated as follows:

DCS Message Type	Data Size	Application	Stream Priority	Number of Clusters	Set of Clusters

Figure 6

The first field (Figure 6) transports the DCS Request message code identifier. Next, the estimated amount of data to be transmitted in the stream, and the application which is requesting the DCS. The next fields identify the stream priority, for computing the new schedule, number of clusters which belong to the set, and their identification. These two last fields are updated as the DCS Request is transmitted upstream.
Upon reception, the Root will wait for a finite period of time for more requests. It will then evaluate the Stream Requests and compute a new TDCS schedule.

STEP 3 – Evaluation and Rescheduling; The evaluation process consists in checking weather or not it is worth it to reschedule the network, considering the amount of data to be transmitted and the inaccessibility time resulting from the reschedule.
Although different techniques could be used to compute this, we are interested in speed and low complexity, due to the scarce processing power of common WSN platforms. The objective is to roughly compute the benefit from scheduling, and to do it fast enough not to delay the process too much.
The result of this computation depends on the techniques chosen for the DCS (DCR or DBR).
Let us consider Stream S_3 from Figure 1, depicting a stream originating at C_{41} for C_{01}. Assume that objective is to minimize end-to-end delay.
To compute this, we start by defining a base unit to simplify the computation. The base unit represents the duration of the active portion of the CH where a stream originates. Hence, if we say that a stream has size *n=1*, this represents a stream which duration is equal to the duration of its CH active portion. All the others CH durations can be represented as multiples of this base unit, because streams move upstream, thus the Bandwidth of the parent CHs, must be equal or higher than their child's. This is imposed by the TDCS algorithm [9] .
We also introduce the concept of *µcycle* and *Macrocycle*. Here, the *µcycle* represents the amount of n units it takes for a stream of size *n=1* to reach its destination. *Macrocycle*, represents the size of the network TDCS schedule in multiples of *n*.
The amount of time to transmit an amount of data represented in multiples of *n* can be computed using the following expression, where *Ti* represents the overhead of the rescheduling which we show how to compute in Step 5.

$$t = \mu cycle + (n-1)\, Macrocycle + Ti$$

For the particular case of the network depicted in Figure 1, with schedule A, and considering a stream originating at C_{41} (S_3), we can compute it's *µcycle* as the number of base units between the different CHs in the path. The result is shown in Table 1.

	A	B
C41→C32	10	2
C32→C24	9	1
C24→C12	7	1
C12→C01	6	1
µcycle	32	5

If we use for instance a re-ordering technique (DCR), this will result in the schedule B depicted bellow, favorable to stream S3, showing a full transaction from source to destination in one TDCS cycle.

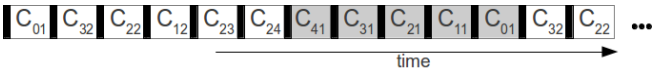

Figure 7

Its *Macrocycle* is the size of the schedule, which is of 10 base units. *Tnit* is computed according to the methodology presented in Step 5 and is equal to 3. Hence, for *n = 1,* considering Schedule A, $t_A = 32 + 0 + 0 = 32$.
For schedule B, with a DCR, $t_B = 5 + 0 + 3 = 8$.
The *Macrocycle* is equal to 10 for both cases.
This expression assumes a collision-free environment, with no contention. This is obviously a simplification, which will always output the shortest time it takes for a flow of data to reach the destination. This method, however, suffices to compute if a re-scheduling is better or not.
The root node will then compute all the offsets that result from the new cluster schedule that will serve that stream and reply to the request.

STEP 4 – Reschedule Response; After the computation of the new offsets (time offset between the beginning of the active portions of the parent and child CHs), according to the new schedule, a response is sent in the payload of the periodic synchronization frame. By using the synchronization frame to deliver this information we make sure that all CHs receive the information in a bounded amount of time, since they are not susceptible to contention and minimize the possibility of collisions.
The first part specifies the message type and the response, (request accepted or request denied). The next portion of the frame contains the expiration for that schedule, which is the amount of cycles the schedule will remain active before returning to the original network schedule. The next portion, contains a list with the new offsets and the cluster-head addresses to which these are to be applied.
Only the CHs which received a new offset are part of the content of the response frame. If the node which requested the rescheduling does not find it's address among the ones in the response, or if no response is received for more than *DCS_maxWait* cycles, it should hold the data and retry later up to a maximum of *DCS_maxRescheduleRetry* times. The

size of *DCS_CH_Address* is implementation specific as well as the *DCS_Offset*, since these variables depend of the protocol used. The frame is formated as follows:

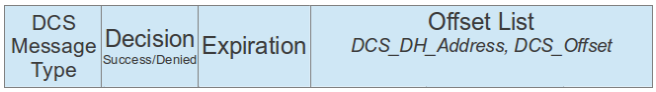

Figure 8

A DCS Messsage ID field to identify the message type, a Decision Field with the response to the DCS Request, an Expiration field with the maximum number of TDCS cycles the schedule is to remain active before returning to the original, and an Offset List, which contains the new offset expressed in a relative offset concerning the original one, and the correspondent Router address.

STEP 5 – Propagation; Each cluster-head, upon reception of the Reschedule Response payload, retrieves its newly assigned offset to their parent and propagates the remaining offset information along the network by placing it in their own synchronization frames, thus propagating the information downstream. The use of the synchronization frames for propagating this information guarantees that all CH receive the necessary information within a bounded amount of time.

The new offset information is then used by the CHs to compute the time for the next synchronization frame. At the next depth, the router joined with that cluster-head must wait for the next synchronization frame (with the new offset) from the parent, and synchronize to it.

This propagation procedure however can introduce a period over which the network is not fully accessible, with the exception of the branches that remained independent of the CHs which were rescheduled. This holds true for the Cluster Re-ordering technique only (DCR).

This is because each CH must wait for the synchronization frame of their parent so that they can align with it and also synchronize their cluster, propagate information and become active, since the offsets are always relative to the parents.

However, this delay is fixed and can be easily computed as a number of schedule cycles as described in the equation bellow.

$$T_i^{DCR} = (d_{Ar} - 1) * macrocycle$$

The inaccessibility time is equal to the Depth of the deepest rescheduled CH (d_{Ar}) in the tree in the schedule minus one, multiplied by the respective duration of one *MacroCycle*. This is the amount of time the scheduled branches of the network should be inaccessible. This is because the scheduled CHs at depth 1, transmits with a new offset immediately in the next cycle, while the scheduled nodes at depth 2 transmit after their parents' hence the delay is an extra cycle, since these schedules always favor an upstream sequence (check Figure 9). If instead of a DCR technique we use a Bandwidth Redistribution technique, this inaccessibility time is zero. Since the hierarchical order of the schedule is kept, the routers will always receive the synchronization frame of their parents immediately before (assuming an initial schedule favoring downstream transmission), and within the same *Macrocycle*.

STEP 6 – Returning to original schedule;

The schedule's change is not permanent, and the network must roll back to it's initial schedule after a defined period of time which we define as the Schedule's Expiration Period.

Because of the inaccessibility period in the DCR technique, each depth will be assigned with a different Expiration so that all depths can change the schedule back to the original in the same cycle. For this reason, Expiration in Step 5 is computed as *Expiration = ED + Ti + 1*, where *ED* is the schedule's expiration deadline that is application defined (*DCS_Exp_Deadline*) and can be computed from the amount of data to be received, *Ti* the inaccessibility time. Each CH will later compute its own Expiration by subtracting their own Depth in the tree.

For the example lets consider S_3 used before, applying a DCR technique, we can consider *ED* equal to $T_k = 4$.

Ti = d(C31) – 1 = 3 – 1 = 2 cycles
Expiration = 4 + 2 + 1 = 7 cycles

Accordingly, each CH will compute their own Expiration time, by subtracting their own depth.

ExpirationC11 = 7 – 1 = 6 cycles
ExpirationC21 = 7 – 2 = 5 cycles
ExpirationC31 = 7 – 3 = 4 cycles

By following this rule, every CH can easily compute when the current schedule expires, in order to return to the original schedule at the same time, just by counting the number of TDCS cycles since their first synchronization frame after the reschedule.

For the case of a DBR technique, expiration will be always equal to the ED, since the inaccessibility time remains equal to zero.

The CHs should activate a counter at the first synchronization frame sent with the new schedule. From this point on, each CH keeps track of the current number of synchronization frames sent by it. When this number is equal to the computer CH Expiration value the CH automatically sets its offset to the original and waits for a synchronization frame from its parent to return to the original schedule. Since the CHs at different Depths will start their counters at a different times, the correct deadline must be computed per depth at each CH.

Figure 9 describes how this process should work for this example of S_3, after a successful reschedule response. The delay of three cycles due to inaccessibility is depicted as well as the schedule expiration.

The first TDCS cycle transmits the new offsets within the DCS Response. Each router will now reset their internal

clock references and wait for a synchronization frame from their parent. C12 and C11 are the first to receive this and they transmit their synchronization frames with the new schedule, followed by their child, (C22, C23, C24, C21).

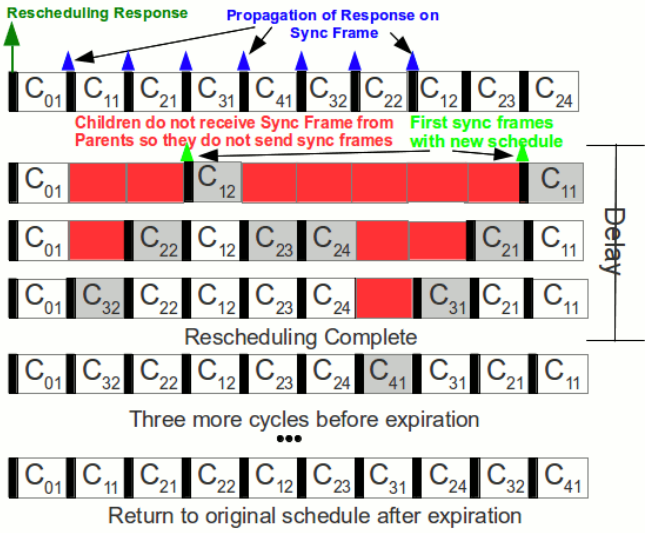

Figure 9

Next, the CHs at depth three do the same until the last CH at depth four (C41) is also rescheduled. The schedule is kept for three more TDCS cycles and it expires. All the offsets return to the original schedule in only one TDCS cycle. As observed, the network inaccessibility time is bounded and return to the original schedule is done without much complexity, as the routers are resynchronized in an hierarchical fashion.

IV. INSTANTIATING ETDCS IN IEEE 802.15.4/ZIGBEE

A. IEEE 8021.5.4/ZigBee Overview

IEEE 802.15.4 and ZigBee [15], particularly the synchronized cluster-tree network model, emerge as potential solutions for industrial WSNs, since they enable to fulfill QoS requirements such as energy-efficiency (dynamically adjustable duty-cycle in a per-cluster basis) and timeliness (best effort/guaranteed traffic differentiation and deterministic tree-routing).

The IEEE 802.15.4 MAC protocol supports two operational modes that may be selected by the ZigBee Coordinator (ZC), which identifies and manages the whole WSN: i) the non beacon-enabled mode, in which the MAC is simply ruled by nonslotted carrier sense multiple access with collision avoidance (CSMA/CA); and ii) the beacon-enabled mode, in which beacons are periodically sent by the ZC for synchronization and network management purposes.

In the beacon-enabled mode, the ZC defines a superframe structure, which is constructed based on the Beacon Interval, which defines the time between two consecutive beacon frames, and on the Superframe Duration (SD), which defines the active portion in the BI, and is divided into 16 equally-sized time slots, during which frame transmissions are allowed. Optionally, an inactive period is defined if BI > SD.

During the inactive period (if it exists), all nodes may enter in asleep mode (to save energy).

BI and SD are determined by two parameters, the Beacon Order (BO) and the Superframe Order (SO), respectively, as follows:

$$\left.\begin{aligned} BI &= aBaseSuperframeDuration \cdot 2^{BO} \\ SD &= aBaseSuperframeDuration \cdot 2^{SO} \end{aligned}\right\} \quad for\ 0 \le SO \le BO \le 14$$

where aBaseSuperframeDuration = 15.36 ms, (assuming 250 kb/s in the 2.4 GHz frequency band) denotes the minimum superframe duration, corresponding to SO = 0.

During the SD, nodes compete for medium access using slotted CSMA/CA in the CAP. For time-sensitive applications, IEEE 802.15.4 enables the definition of a contention-free period (CFP) within the SD, by the allocation of guaranteed time slots (GTSs). Low duty-cycles are achieved by setting small values of the superframe order (SO) as compared to the beacon order (BO), leading to longer sleeping (inactive) periods.

ZigBee defines network and application layer services on top of the IEEE 802.15.4 protocol. In the cluster-tree model, all nodes are organized in a parent-child relationship, network synchronization is achieved through a distributed beacon transmission mechanism and a deterministic tree routing mechanism is used.

A ZigBee network is composed of three device types: (i) the ZigBee Coordinator (ZC), which identifies the network and provides synchronization services through the transmission of beacon frames containing the identification of the PAN and other relevant information; ii) the ZigBee Router (ZR), which has the same functionalities as the ZC with the exception that it does not create its own PAN—a ZR must be associated to the ZC or to another ZR, providing local synchronization to its cluster (child) nodes via beacon frame transmissions; and (iii) the ZigBee End-Device (ZED), which neither has coordination nor routing functionalities and is associated to the ZC or to a ZR.

B. Integrating Elastic Management in a ZigBee Network

The PAN-Coordinator is responsible for receiving the new schedule request from the other cluster-heads and computing the new schedule as described before.

A new module was devised to be integrated above the network layer of ZigBee (Figure 10), at the Application Support Layer. This new module, DCS, is responsible for managing the DCS mechanism, in regards to the beacon payload creation (for propagating offset information), computing and changing the offset information for the lower layers, and computing the schedules and corresponding expiration.

At network setup time, the TDCS algorithm is applied to the the tree, setting up the base schedule. This schedule depends on the application, but generally favors downstream traffic,

for faster actuation upon the leaf nodes and setting up other application related parameters. In this way, the resulting schedule should be one which establishes precedence between parent-child, in a way that in one BI all clusters can be reached. This is a necessary condition for the DCS to be successful.

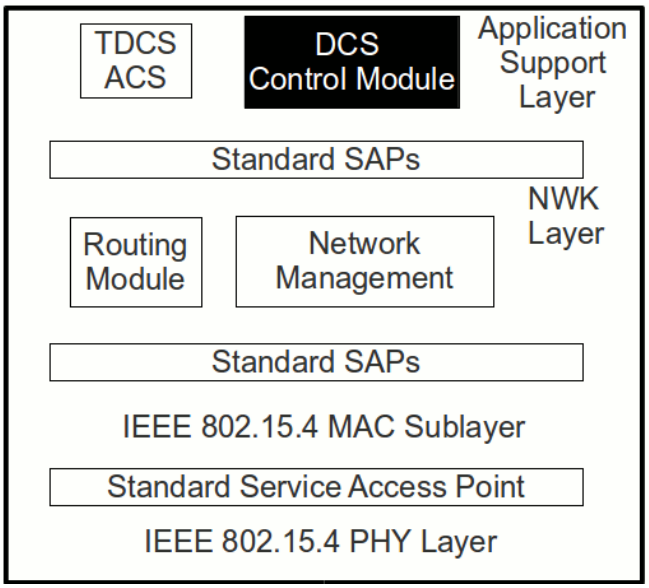

Figure 10

As the nodes gather data, they can direct streaming requests at the PAN Coordinator. The PAN-Coordinator will evaluate these requests according to what is described in Step 2 of Section III B 3. If the result is positive, it will compute the new schedule and setup the Rescheduling Response to be placed in the IEEE 802.15.4 Beacon Payload. The next Beacon frame will carry this information.

As Beacons are transmitted between the several clusters, the Rescheduling information is propagated among the tree. As the Zigbee Routers receive the Rescheduling Response and the new offsets to their parent, all the nodes will know a DCS Rescheduling in occurring just by parsing the received Beacon. This is important since in the next BI, many nodes will fail to receive a Beacon from their parent, due to the inaccessibility time described in Step 4. This will be specially visible in the deepest nodes of the rescheduled branch. If no information concerning the status of the process was propagated, the nodes could assume they had lost their parent, receiving a *SYNC-LOSS.indication* from the respective MAC layer, and would try an Association procedure to another potential parent. By knowing this in advance, they can disable this process for *(Depth-1)*BI* amount of time, which is the maximum time the rescheduling should take per Depth, after which, the Device will re-enable the re-association procedure after the *SYNC-LOSS*.

Upon reception of their parent's Beacon, the ZigBee Cluster-Heads, will search for their address among the Rescheduling information at the Beacon Payload to learn the new offset. Then, they will trigger the DCS Module generating a *DCS-NEW-SCHEDULE.indication*, and set their own Beacon Payload with the remaining information of the Rescheduling Response to propagate the information to the children down the tree. Having done this, the DCS Module, will issue a *SYNC.reques*t to the Network Layer to resynchronize with the corresponding parent, and after a synchronization an *MLME-START.request.*

The *MLME-START.request* primitive, depends of the rescheduling technique to be used. If a Re-ordering technique is to be used, then the CH will used a *DCS-RESTART-ROUTER.request*, with the new offset information. This new interface is similar the standard *NLME-RESTART-ROUTER.request*, except no change is done to the other parameters of the stack. The objective is to simply turn the routing functionality on.

If a Bandwidth reallocation is to be done, then the request will also change the Superframe Order parameter of the stack to reflect the bandwidth change. The system timers at the MAC layer, upon reception of this request are automatically updated with the new Superframe Order. Upon the reception of a Beacon from the parent, the ZigBee Router will automatically resynchronize and resume its work.

When the DCS Module is triggered, the Schedule Expiration is also computed according to what is described in Step 6 of Section III B 3, and a counter (*DCS_Expiration_timer*) is triggered with that value. When this counter expires, the DCS Module automatically repeats the *DCS-RESTART* process with the old offset values, returning to the initial values. These are stored in a database, DCS_Init_db, which contains the initial offset and Superframe Order values.

As described, the implementation of the DCS mechanism does not involve major changes to the protocol. In fact, only a couple of new primitives are to be added to the ZigBee NWK stack to enable the DCS functionalities.

V. Performance Evaluation

The DCS mechanism was evaluated through simulation and experimentally using a real world Structural Health Monitoring application as a testbed. This application, previously designed in [16] and [17], was chosen considering its requirements of tight node synchronization and control, and the large amount of sensing data that must be handled by the network.

Its system architecture was designed to sample in a synchronized fashion multiple accelerometers placed at different locations in a physical structure and forward this data to a central station (PAN-Coordinator) for later processing using a IEEE 802.15.4/ZigBee Cluster-Tree network topology. Each Sensing Node is composed by a TelosB node [18] with a signal acquisition board, with a 24 bits DAC, attached to a MEMS 3-axis acceleration sensor (Figure 11).

The Coordinator Node supervises the network and nodes' activities (e.g. node configuration, data acquisition rate, start/stop sampling) and guarantees a tight synchronization between all nodes which is of the utmost importance for this kind of applications; it also forwards the configuration parameters and dispatches the acquired data to the Command & Configuration Application (C&C App) which provides the system user with a human-machine interface (HMI) to configure the system and also an application programming interface (API) to integrate the WSN system with the data processing/analysis applications. The latter enable to infer about the reaction of the monitored structure to natural vibration or impacts. For more detail about the SHM system please refer to [16].

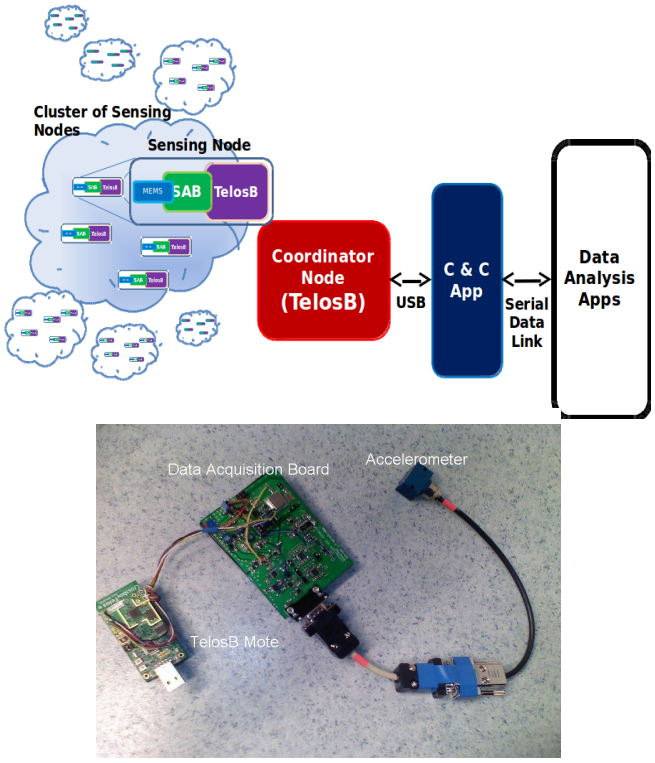

Figure 11

1) System setup

The network is setup according to Figure 1 network topology and the Sensing Nodes are spread into different clusters. In Figure 1, the addresses next to the nodes represent the Cluster-Heads' ZigBee NWK addresses.

In this example application, the initial schedule favors downstream communications and is setup as Figure 12 Schedule 1. This is made so that the PAN-Coordinator, after setting up all the nodes in the network, is able to start and stop the data acquisition on all the nodes simultaneously. This is mandatory for the application so that the results are coherent.

Notice the free spaces in comparison with the schedules previously presented. This has to do with the discrete steps which are allowed for the BO and SO settings in the IEEE 802.15.4 protocol. Because BO = 8 and SO = 4 was chosen for all ten routers, there is space in the TDCS cycle which remains free. Again, the blue lines around the schedule mark one complete TDCS cycle. They gray filled spaces mark the routers involved in the stream transmission.

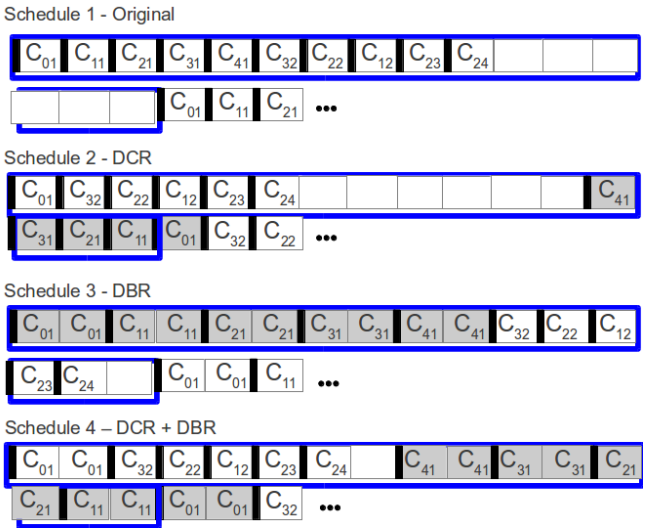

Figure 12

When the data acquisition finishes, the Sensing Nodes are pooled in turn for the sensing data. Each reading for the accelerometer has a size of 3 axis times 24 bits resulting in 72 bits. Depending on the sampling rate, a large volume of data is going to be generated and transmitted to the PAN-Coordinator, which will forward it to a PC for processing. Although there is no real-time requirement in this part of the operation, meaning receiving data in a bounded amount of time, since no real-time analysis is performed, engineers which to receive all the data in the minimum possible time since they need several runs to be carried out. If the initial schedule is kept, this operation will take a large amount of time to complete, and the assessment can last several minutes and even hours, depending on the origin of the stream. We wish to change the schedule to accelerate the data transfer from the Sensing Nodes at that Cluster to the PAN-Coordinator using the DCS mechanism using the Bandwidth Reallocation technique, resulting in a schedule depicted in Figure 12 Schedule 3. However, the DCR technique (Figure 12 Schedule 2 and 4) was also implemented and analyzed.

2) Simulation Results

The DCS mechanism was implemented over the Open-ZB Zigbee Model [15], and simulated with the OPNET Modeler simulation software. A network topology like the one shown in Figure 1 with nwkMaxChildren (Cm) = 3, nwkMaxDepth (Dm) = 5, and nwkMaxRouters (Rm) = 2, was setup and the application layer of the node was set to generate traffic at a rate correspondent to a sampling rate of

100Hz which is recommended for fine-grained structural health monitoring [16]. For maintaining uniformity along this paper, in the analysis we always consider stream S_3, which originates at router C_{41} in Figure 1. Figure 13 shows one of the simulation scenarios.

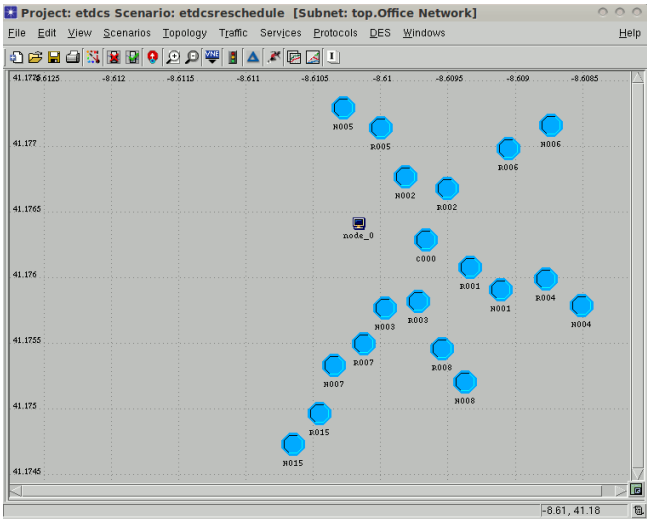

Figure 13

Several analysis to evaluate the performance of the two techniques were carried out, with a special attention to two metrics: end-to-end delays and overall stream transmit duration.

End-to-end Delay Analysis

To understand the impact of the first technique we did several runs of the network with different BO settings (from BO = 8 up to BO = 12), simulating a larger network, with the initial scheduling and using the re-ordering technique.

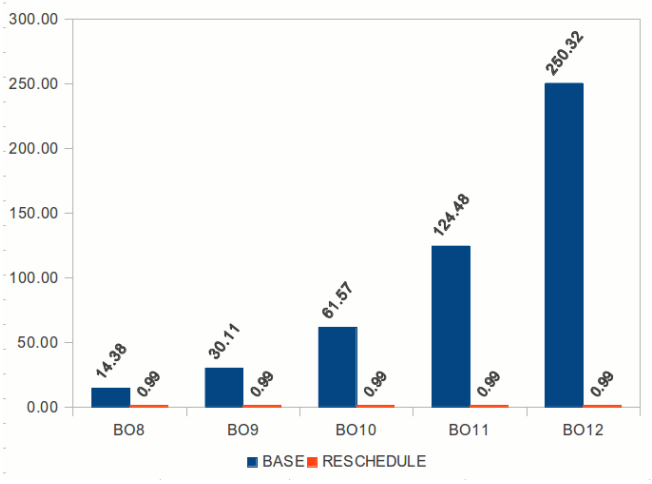

Figure 14

The end-to-end delays were measured for packets transmitted from Sensing Node with address 0x0007 (Figure 1), which was associated to Router 0x0004, at Depth 4, to the PAN-Coordinator, with no extra traffic on the network. Frame size was set to 800 bits, and Packet Inter-arrival Time was set to 0,06 seconds to emulate the arrival of Sensing Data at the Sensing Node's serial port (this was verified experimentally).

Figure 14, shows the end-to-end delay results for the different BO. Superframe order is fixed to SO = 4. Notice the decrease on the delay achieved by simply re-ordering the schedule. We can achieve a reduction in the end-to-end delays in the order of 13 seconds for BO = 8 and even several minutes as the BO increases with the size of the network, reaching 4 minutes for the case of BO = 12, to approximately one second.

The end-to-end delays with DCR remain constant despite the different BO settings. This is expected since although the network increases, the transmission of a packet is completed in only one BO cycle. Since the Bandwidth of the routers is also the same, the end-to-end delay should remain constant and thus independent of the network size.

To understand the impact of the second DCS technique, Bandwidth Reallocation, on the end-to-end delay, the initial schedule's order was maintained and the available bandwidth of the Superframe was distributed among the Routers involved on the stream. Different BO/SO configurations were tried (SO = 4 up to SO = 9) and as more space became available with the increasing BO, the DBR technique was used to distribute it through the routers. Figure 15 presents the results for the different SO settings using BO = 10 and both DCS techniques. Results are similar to all other BO settings.

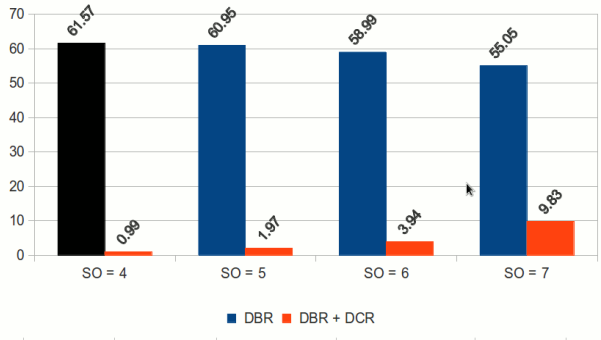

Figure 15

There is a slight but not significant decrease of the end-to-end delay as the SO are increased. Since the Routers increase their SO, the unused part of the Superframe was reduced and thus there is a better use of the Superframe bandwidth. This reduces the time the packet must remain in the queue at each router, waiting for the next Superframe to be transmitted to the parent, thus slightly reducing the overall end-to-end delay. This is visible in Figure 16 (top blue squares), showing how the average queuing delay decreases as the SO increases.

In comparison, the DCR technique presents a much higher impact on the end-to-end delay as expected, decreasing for

the case of BO/SO = 10/5, the delay from 60,95 to 1,97 seconds, a decrease of 96,7%. In fact, for its worst case of BO/SO = 10/7, it still represents a decrease of 82,14% concerning the DBR technique, as show in Figure 15.

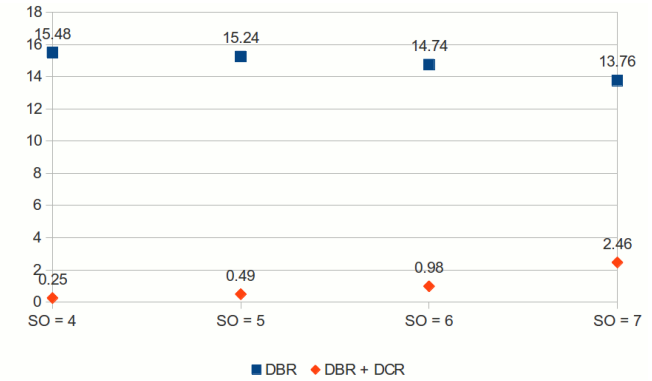

Figure 16

There is however, a slight increase in the delay when using the DRT technique as the SO increases. Although, there is a re-ordering of the schedule according favoring upstream traffic, and a redistribution of the unused bandwidth, the increase in SO implies a larger time a packet must wait in queue at each router, waiting to be transmitted to the parent, in comparison to the cases with lower SO.

Using the DBR technique is thus not recommended when one wishes to significantly reduce the end-to-end delay in the application.

Stream Overall Transmission Time

Like previously mentioned, minimizing the overall transmission time is quite important, in our SHM application, where large amounts of data must be transmitted in the less amount of time possible. To analyze the second metric, the overall stream transmission time, we generated different amounts of data at the application layer to simulate the different sampling durations of the SHM application. We generated scenarios with different volumes of sensing data, corresponding to short 10 and 30 seconds runs and runs with 1, 5, 10 and 30 minutes, in the SHM system. Data frame size remained fixed to 100 KB.

We measured how much time it took for the data transfer to complete. During this time, there was no more traffic in the network, so that collisions were not possible, not to interfere with the experiment.

Figure 17 shows the results for sampling durations of 10, 30 and 60 seconds. As shown, the DBR technique presents the best result in decreasing the overall transmission time, representing a decrease close to 50%, as expected when the available bandwidth is doubled on the Routers, to SO = 5.

For the case of 1 minute of sampling time, using the DBR technique alone reduced the overall transmission time from nine minutes to 4 and a half minutes, a decrease of 49%.

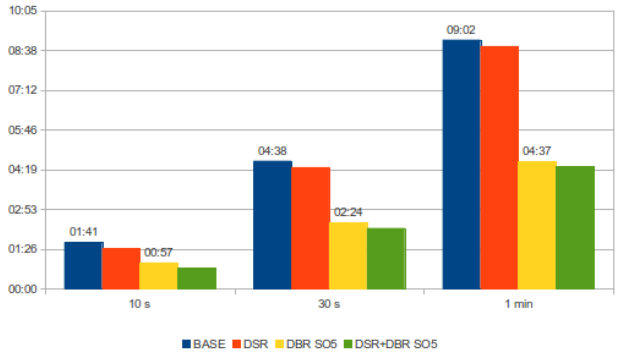

Figure 17

Interestingly, the DCR technique also decreases the overall transmission time, but not in a significant way. It decreases it about 14 seconds for this particular case of BO=8, and it is constant for every SO setting, independently of the amount of data to be transmitted. This small difference, however should not be neglected. For larger BO, the impact of this increases as shown in Figure 18, reaching 8 minutes for BO=13. This happens because of the impact of the reduced end-to-end delay at the beginning of the transmission. With a re-ordering of the clusters' schedule, the first packets are delivered in a shorter amount of time, in only one TDCS cycle, contrary to what happens when this technique is not used, taking several cycles to complete the transmission of the first packets. Because of this, the transmission will end sooner. As the BO increases, the impact of this is higher since the duration of the TDCS cycle also increases.

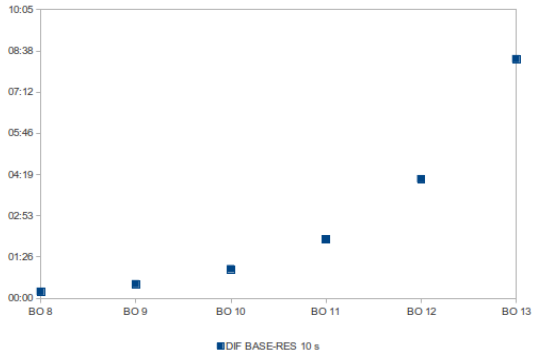

Figure 18

Figure 19, shows the overall stream transmission times for both techniques and different sampling durations, using BO=9.

For a sampling duration of 30 minutes, we achieve a 75% reduction in the overall transmission time. This means, that instead of waiting for almost 9 minutes to receive the SHM application sensing data, we just need approximately 2 minutes.

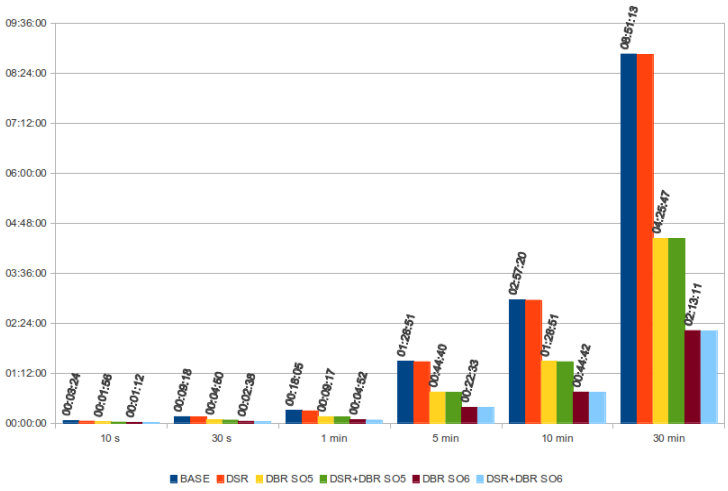

Figure 19

Notice, this is to receive from one node only. This must be multiplied by the number of sensing nodes on that cluster, since they are pooled in turn in the SHM application. If we were to have 6 sensing nodes on the cluster, this would take 12 minutes to complete against 54 minutes with the base schedule, a reduction close to 78% on the waiting period for the sensing data from cluster C41 (0x0004).

3) *Experimental Evaluation*

The DCS module was implemented in TinyOS over the Open-ZB IEEE 802.15.4/ZigBee stack [10]. A ZigBee network with 12 TelosB [18] motes was setup in a configuration replicating the one depicted in Figure 1, using BO=8 and SO=4, with one PAN-Coordinator connected to a PC through a USB connection, and nine Routers each forming their own cluster. Two Sensing Nodes (End Devices) were associated to the Router at Depth 4 (address 0x0004) to generate sensing data for later retrieval. To reduce costs, the Sensing Nodes were used without the accelerometer modules. Instead, timers at the application layer were used to generate traffic at different sampling rates. The control of the application, concerning the data acquisition period and rate, was done using the Command & Configuration Application (C&C App), running in the PC, attached to the PAN-Coordinator. Figure xxxx shows the setup.

Both DCS techniques were implemented and tested to validate our work, although the most important technique for this specific SHM application is the DBR, which as shown before can greatly reduce the overall stream transmission time. A base scenario, without any schedule improvement, was also setup to measure the improvement.

A Daintree Networks 2400E Sensor Network Analyzer [19] was used to log all the communications during the experimental evaluation.

Starting with the DBR technique, the most important parts of the log are highlighted in Figure 20 and commented bellow.

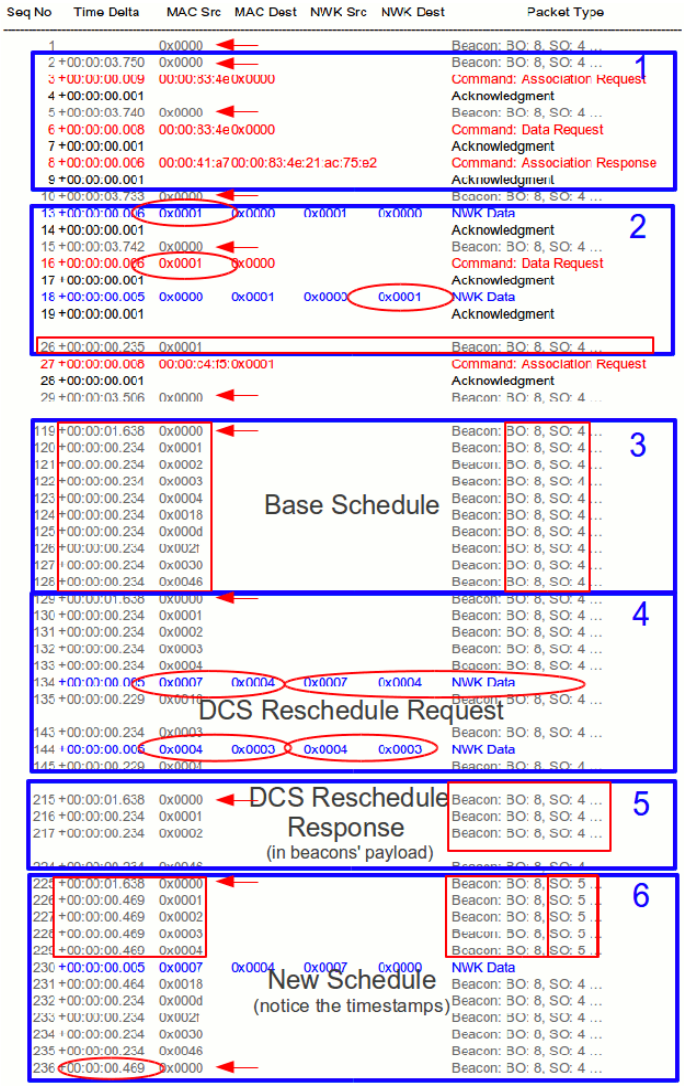

Figure 20

A few packets were omitted for space reasons to simplify the reading.

This figure shows the use of the DBR technique to reduce the overall stream transmission time. Beacons from the PAN Coordinator are signaled with a red arrow.

At network setup time, the nodes associate (1) and the TDCS algorithm assigns each cluster an offset (2), assuming a the initial schedule favoring downstream communication, thus improving the control over the application during the sampling period. This information is sent in a Data Message (blue Data Message inside rectangle 2). The application is configured and started (3), and the base schedule can be seen. Upon completion of the data acquisition task, the application pools the nodes for data, using the protocol described in [16]. The first Sensing Node to be pooled, wishes to initiate upstream data communication and triggers

the DCS mechanism (4) with a DCS Request. This request is forward by the routers until it is delivered to the PAN-Coordinator. Two Data Messages with DCS Request can be seen being forwarded. The relationship between addresses and the logical topology is shown in Figure 1.

Upon arrival, the PAN Coordinator computes the new schedule and sends to the network a DCS Reschedule Message which is disseminated within the payload of the beacon frame throughout all the network (5). The new schedule is immediately adopted as shown in (6), and a change on the SO is noticeable on the Beacon frame description and on the sniffer timestamps.

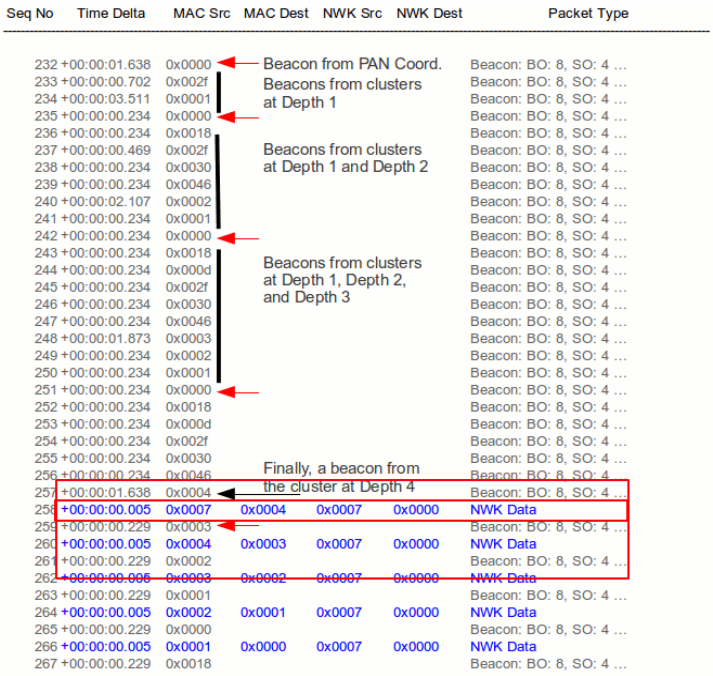

Figure 21

The DCR technique was also evaluated on this experimental setup and Figure 21, shows the output from the Packet Analyzer. Part of the output related to the network setup and DCS communication was omitted since is is already shown on the previous figure and there are no significant changes for this technique. Again, the beacons from the PAN Coordinator are signaled with a red arrow.

When all the Routers receive their new offset information in the DCS Reschedule Response message, they immediately stop sending beacons and wait for their parent's beacon to synchronize to it. The first beacon comes from the PAN Coordinator which maintains its period. Next, Routers at Depth one are the firsts to synchronize to it using the new offsets. Notice the Packet Analyzer time stamp, showing the new relative offsets. Now that the Depth one Routers transmitted their beacons, the next level ones (Depth two) can also synchronize. The process continues until the all the Routers are synchronized. At this point, the Sensing Nodes (0x0007 in the example) start transmitting data which will forward until it reaches the sink.

Next figure shows the comparison between simulation and experimental results. As observed, the behavior previously observed in simulation is replicated in the experimental evaluation with minor differences.

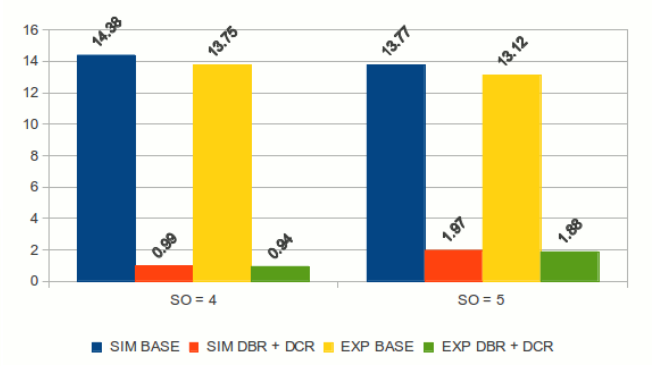

Figure 22

A reduction of 93% on the end-to-end delay is achieved with the DCR technique for BO = 8 in our application. Again, the DBR technique, with an increase of Bandwidths to SO = 5, does not present a significant change to this metric. Results are quite close to simulation on the end-to-end delay result with DCR, however for the base schedule, experimental delay was slightly different. This has to do with the different duration of the Beacon Order on the experimental platforms, due to timing constraints, which is of 3,75 seconds instead of the theoretical 3,932 seconds.

Concerning the DBR technique, results show a reduction on the overall transmission time in the order of 49%, again quite close to simulation results.

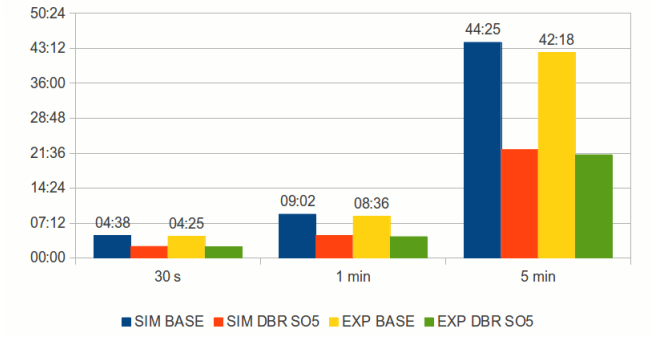

Figure 23

Concerning the network inaccessibility time, as predicted, it was bounded to three TDCS cycles, which is the time it takes for the whole network to resynchronize with the new schedule. This can be observed in the Packet Analyzer output files.

VI. CONCLUSIONS

Although Synchronized Cluster-Tree network topologies look promising to enable WSN applications with stringent QoS requirements, due to the predictability which can be achieved, we have witnessed a lack of commercial and academic solutions based on this kind of topology, in part due to the technical challenges their engineering imposes.

Among these challenges, changing the resource allocation of the CT on the fly, without imposing long inaccessibility times still remained open to research. This possibility would dramatically improve these networks' flexibility in adapting to changes in the traffic or bandwidth requirements in real-time.

In this paper we presented a solution to this problem, enabling networks to adapt in real-time to different bandwidth and end-to-end delay requirements imposed by incoming traffic streams, by changing the clusters' scheduling. We presented two DCS techniques which can reduce the end-to-end latency of a stream up to 93%, and the overall data transmit duration from a leaf node to the sink up 50% although more can be achieved with other network settings.

Importantly, our methodology was applied to a real-world WSN-based Structural Health Monitoring system, showing that it can be easily implemented under the IEEE802.15.4/ZigBee set of protocols with minor add-ons and can run in general purpose WSN platforms such as the TelosB motes.

REFERENCES

[1] Stankovic, J., Lee, I., Mok, A., and Rajkumar, R. 2005. Opportunities and obligations for physical computing systems. IEEE Computer 38, 11 (Nov.), 25–33.

[2] Raman, B. and Chebrolu, K. 2008. Censor networks: a critique of "sensor networks" from a systems perspective. ACM SIGCOMM Computer Communication Review 38, 3 (July), 75–78.

[3] Abdelzaher, T., Prabh, S., and Kiran, R. 2004. On real-time capacity limits of multihop wireless sensor network. In Proceedings of the 25th IEEE International Real-Time Systems Symposium (RTSS). IEEE Computer Society Press, Washington, DC, USA, 359–370.

[4] Gibson, J., Xie, G., and Xiao, Y. 2007. Performance limits of fair-access in sensor networks with linear and selected grid topologies. In Proceedings of the 50th IEEE Global Communications Conf. (GLOBECOM). IEEE Computer Society Press, Washington, DC, USA, 688–693.

[5] Prabh, S. and Abdelzaher, T. 2007. On scheduling and real-time capacity of hexagonal wireless sensor networks. In Proceedings of the 19th Euromicro Conf. on Real-Time Systems (ECRTS). IEEE Computer Society Press, Washington, DC, USA, 136–145.

[6] Petr Jurcik, Anis Koubâa, Ricardo Severino, Mário Alves, and Eduardo Tovar. 2010. Dimensioning and worst-case analysis of cluster-tree sensor networks. *ACM Trans. Sen. Netw.* 7, 2, Article 14 (September 2010), 47 pages.

[7] Hanzálek, Z.; Jurčík, P.; , "Energy Efficient Scheduling for Cluster-Tree Wireless Sensor Networks With Time-Bounded Data Flows: Application to IEEE 802.15.4/ZigBee," *Industrial Informatics, IEEE Transactions on* , vol.6, no.3, pp.438-450, Aug. 2010.

[8] IEEE-TG15.4. 2006. Part 15.4: wireless medium access control (MAC) and physical layer (PHY) specifications for low-rate wireless personal area networks (LR-WPANs). IEEE Computer Society.

[9] Koubaa, A., Cunha, A., and Alves, M. A time division beacon scheduling mechanism for IEEE 802.15.4/ZigBee cluster-tree wireless sensor networks. In Proceedings of the 19th Euromicro Conf. on Real-Time Systems (ECRTS). IEEE Computer Society Press, Washington, DC, USA, 125–135.

[10] Cunha, A., Koubaa, A., Severino, R., and Alves, M. Open-ZB: an open source implementation of the IEEE 802.15.4/ZigBee protocol stack on TinyOS. In Proceedings of the 4th IEEE International Conf. on Mobile Ad-hoc and Sensor Systems (MASS).

[11] R. Burda and C. Wietfeld "A Distributed and Autonomous Beacon Scheduling Algorithm for IEEE802.15.4/ZigBee Networks", in Proc. of IEEE MASS 2007, Pisa, Italy, Oct. 2007.

[12] Muthukumaran, P.; de Paz Alberola, R.; Spinar, R. & Pesch, D. MeshMAC: Enabling Mesh Networking over IEEE 802.15.4 through Distributed Beacon Scheduling., in Jun Zheng; Shiwen Mao; Scott F. Midkiff & Hua Zhu, ed., 'ADHOCNETS' , Springer, , pp. 561-575 .

[13] Toscano, E.; Lo Bello, L.; , "A multichannel approach to avoid beacon collisions in IEEE 802.15.4 cluster-tree industrial networks," *Emerging Technologies & Factory Automation, 2009. ETFA 2009. IEEE Conference on* , vol., no., pp.1-9, 22-25 Sept. 2009

[14] Cunha, A., Severino, R., Pereira, N., Koubaa, A., and Alves, M. 2008. ZigBee over TinyOS: implementation and experimental challenges. In Proceedings of the 8th Portuguese Conf. On Automatic Control (CONTROLO). UTAD, Portugal, 911–916.

[15] ZigBee. 2005. ZigBee Specification, Version 1.0. ZigBee Standards Organization.

[16] Ricardo Severino, Ricardo Gomes, Mário Alves, Eduardo Tovar, Rafael Aguilar, Paulo Lourenço, Paulo Gandra de Sousa. A Wireless Sensor Network Platform for Structural Health Monitoring: enabling accurate and synchronized measurements through COTS+custom-based design. In workshop "Applications of Wireless Sensor Networks", co-located with the 5th IFAC International Conference on Management and Control of Production and Logistics, University of Coimbra, Portugal, September 8-10, 2010.

[17] Rafael Aguilar, Luis F. Ramos, Paulo B. Lourenço, Ricardo Severino, Ricardo Gomes, Paulo Gandra, Mario Alves, Eduardo Tovar. Prototype WSN Platform for Performing Dynamic Monitoring of Civil Engineering Structures . Operational Modal Monitoring of Ancient Structures using Wireless Technology . In Sensors, Instrumentation and Special Topics, Volume 6 , Conference Proceedings of the Society for Experimental Mechanics Series Volume 9, 2011, pp 81-89

[18] Crossbow. 2013. TelosB mote datasheet. [Online]. Available: http://www.xbow.com.

[19] Daintree Networks. 2012 Sensor Network Analyzer (SNA). [Online] Available: http://www.daintree.com.